



Описание функциональных
характеристик
версия 0.6.0

ООО «Веб-Сервер»

дек. 09, 2025

Оглавление

1	Аннотация	1
2	Общие сведения	2
3	Балансировщик нагрузки	3
3.1	Методы балансировки	4
3.1.1	HTTP-балансировка	4
3.1.2	Stream-балансировка	4
3.1.3	Алгоритмы балансировки	5
3.1.4	Примеры HTTP-балансировки нагрузки	5
3.1.5	Примеры TCP/UDP-балансировки нагрузки	9
3.2	Проверки работоспособности серверов	12
3.2.1	Пассивные проверки	12
3.2.2	Активные проверки	13
3.3	Резервирование проксируемых серверов	15
3.4	Просмотр и редактирование конфигурации	16
3.4.1	Просмотр и редактирование конфигурации балансировщика нагрузки	16
3.4.2	Использование портов	17
3.5	Резервное копирование и восстановление конфигурации	17
3.5.1	Просмотр списка сохраненных конфигураций	17
3.5.2	Изменение рабочего статуса конфигурации	17
3.5.3	Восстановление последней рабочей версии конфигурации	18
3.5.4	Восстановление конфигурации из резервной копии	18
3.6	Справочная информация	18
3.6.1	Общие сведения	18
3.6.2	Основной модуль	18
3.6.3	HTTP-модули	19
3.6.4	Потоковые модули	19
3.6.5	Почтовые модули	19
3.6.6	Сторонние модули	19
3.6.7	Инструкции	19
4	Глобальная балансировка	484
4.1	Методы балансировки	485
4.1.1	Алгоритмы балансировки	485
4.1.2	Проверки работоспособности	485
4.1.3	Примеры	486
4.2	Настройка GSLB	490
4.2.1	Настройка конфигурации GSLB	491
4.3	Параметры конфигурации	492
4.3.1	groups	492

4.3.2	zones	492
4.3.3	options	493
4.3.4	rules	494
4.3.5	servers	495
5	IP-маршрутизация	497
5.1	BGP-маршрутизация	498
5.1.1	Резервирование с помощью протокола BGP (Active-Standby)	498
5.1.2	Распределение нагрузки с помощью протокола BGP (Active-Active)	507
5.2	OSPF-маршрутизация	519
5.2.1	Резервирование с помощью протокола OSPF (Active-Standby)	519
5.2.2	Распределение нагрузки с помощью протокола OSPF (Active-Active)	527
5.3	VRRP-маршрутизация	534
5.3.1	Предварительная настройка для VMware ESXi	535
5.4	Использование протокола BFD для уменьшения времени реакции	554
5.4.1	Введение	554
5.4.2	Настройка с OSPF	555
5.4.3	Настройка с BGP	556
5.4.4	Настройка с VRRP	558
5.4.5	Настройка со статическими маршрутами	558
5.4.6	Настройка независимого пира	559
5.4.7	Профиль	560
5.5	Настройка RHI	560
5.5.1	Настройка BGP	561
5.5.2	Настройка OSPF	562
5.5.3	Настройка балансировщика нагрузки	564
5.5.4	Настройка конфигурации RHI	565
6	Высокая доступность	567
6.1	Обзор работы	568
6.1.1	Развертывание	568
6.1.2	Режим Active / Standby	568
6.1.3	Статус пары высокой доступности	569
6.1.4	Проверки доступности узлов пары	569
6.1.5	Синхронизация	569
6.2	Создание пары высокой доступности	570
6.2.1	Требования	570
6.2.2	Предварительные действия	570
6.2.3	Создание пары высокой доступности в режиме Active / Standby	570
6.3	Настройка высокой доступности в паре	571
6.3.1	VRRP	571
6.3.2	OSPF	574
6.3.3	BGP	577
6.4	Управление парой	581
6.4.1	Просмотр информации о паре	581
6.4.2	Синхронизация узлов	582
6.4.3	Настройка протокола опроса доступности узлов	583
6.4.4	Удаление пары	583
7	Управление	584
7.1	Управление пользователями	584
7.1.1	Требования	584
7.1.2	Добавление нового пользователя	585
7.1.3	Изменение учетных данных существующего пользователя	585
7.1.4	Изменение статуса существующего пользователя	585
7.1.5	Удаление пользователя	585
7.2	Выбор репозитория для резервного копирования	586
7.2.1	Локальный репозиторий (filesystem)	586
7.2.2	Внешний репозиторий (S3)	586

7.2.3	Управление репозиториями	586
7.3	Запуск командной строки (CLI)	587
7.3.1	Запуск новой версии CLI (на порту 2222)	588
7.3.2	Запуск старой версии CLI (на порту 2022)	588
7.4	Справочник команд (CLI)	588
7.4.1	Поддерживаемые команды	589
7.4.2	Синтаксис	589
7.4.3	cert-config	590
7.4.4	configuration	594
7.4.5	diagnostics	597
7.4.6	firewall	604
7.4.7	ip	606
7.4.8	logs	606
7.4.9	modules	610
7.4.10	ps	611
7.4.11	settings	612
7.4.12	system	615
7.4.13	transfer	616
7.4.14	vttysh	617
8	Права на интеллектуальную собственность	618
	Алфавитный указатель	619

ГЛАВА 1

Аннотация

Настоящий документ содержит описание функциональных характеристик Angie ADC.

Angie ADC — программное обеспечение класса "контроллер доставки приложений", которое представляет собой систему балансировки, включающее DNS-балансировку, а также позволяющее маршрутизировать и балансировать сетевые запросы, используя протоколы маршрутизации внешнего и внутреннего шлюза.

глава 2

Общие сведения

Angie ADC — комплексное программное обеспечение для балансировки нагрузки и управления сетевым трафиком для создания гибкой, производительной и безопасной инфраструктуры.

Особенности:

- Балансировщик нагрузки на уровнях L4-L7.
- Глобальная DNS-балансировка (GSLB).
- IP-маршрутизация.
- Решения для обеспечения высокой доступности.
- Присутствие в реестре российского ПО.

Angie ADC имеет удобный веб-интерфейс, командную строку (CLI) и API для интеграции с внешними системами, что обеспечивает понятный и надежный мониторинг и управление функциями.

Angie ADC поставляется как виртуальное устройство (Virtual Appliance).

ГЛАВА 3

Балансировщик нагрузки

Балансировка

Балансировщик нагрузки позволяет распределять клиентские запросы по серверам на основании выбранного алгоритма балансировки, данных статистики или обратной связи от сервера (см. *Методы балансировки*). В идеале клиентские запросы равномерно распределяются по всем серверам, что увеличивает совокупную производительность системы. Доступна балансировка на двух уровнях: L7 (протокол HTTP) и L4 (TCP и UDP).

Высокая доступность

Высокая доступность системы обеспечивается за счет *проверок работоспособности* апстрим-серверов и создания *дополнительных резервных групп серверов*.

Дополнительная настройка

Помимо алгоритмов распределения запросов, в HTTP- и TCP/UDP-балансировке могут использоваться смежные модули:

- Модуль `queue` создает очередь запросов. Если балансировщик не смог выбрать апстрим-сервер для обработки, то запрос попадает в очередь ожидания.
- Модуль `keepalive` позволяет держать открытые неактивные соединения с апстрим-серверами, т. е. не закрывать соединения при завершении обработки запроса.
- Директива `bind_conn`, реализованная в модуле `keepalive`, позволяет привязать клиентское соединение к серверному соединению.
- Модуль `sticky` также позволяет привязывать клиентское соединение к апстрим-серверу, но на основе информации, содержащейся в самом запросе.
- Модуль `zone` обеспечивает доступ с использованием REST API. API позволяет получать набор метрик, в том числе относящихся к балансировке.
- Модуль `upstream_probe` позволяет обеспечить активные проверки апстрим-серверов.

Подробнее см. в *справочнике HTTP-модуля* и в *справочнике потокового модуля*.

Хранение версий конфигурации

Все версии конфигурации балансировщика нагрузки сохраняются. Вы можете просмотреть историю изменений, а также *восстановить конфигурацию* из резервной копии.

Справочная информация

В разделе *Справочная информация* представлены сведения о встроенных и сторонних модулях, примеры их настройки, а также поддерживаемые ими директивы и переменные.

В этом разделе:

- *Методы балансировки*
- *Проверки работоспособности серверов*
- *Резервирование проксируемых серверов*
- *Просмотр и редактирование конфигурации*
- *Резервное копирование и восстановление конфигурации*
- *Справочная информация*

3.1 Методы балансировки

3.1.1 HTTP-балансировка

Для приложений и веб-сервисов доступна балансировка на уровне приложений L7 (HTTP) с возможностями донастройки с помощью *keepalive*, *sticky*, *zone*, *upstream_probe*, *queue* и *groupn backup-серверов*. Настройка производится в секции **http** в конфигурации балансировщика.

3.1.2 Stream-балансировка

Для работы на уровне L4 (TCP, UDP) доступна stream-балансировка с возможностями донастройки с помощью *sticky*, *zone*, *upstream_probe* и *groupn backup-серверов*. Настройка производится в секции **stream** в конфигурации балансировщика.

3.1.3 Алгоритмы балансировки

Алгоритм	Описание	HTTP/TCP/UDP
round-robin	Запросы распределяются по серверам последовательно (используется по умолчанию). Поддерживается учет веса серверов (weight), медленный старт (slow_start), backup-группы серверов.	
hash	Задаёт метод балансировки нагрузки для группы, при котором соответствие клиента серверу определяется при помощи хэшированного значения ключа. Поддерживается учет веса серверов (weight). Подробнее см. TCP/UDP , HTTP .	
ip_hash	Сервер выбирается с помощью хэш-функции на основе IP-адреса клиента, что обеспечивает "липкость сессии". В итоге запрос конкретного клиента всегда попадает на один и тот же сервер, если он работоспособен. Поддерживается учет веса серверов (weight). Подробнее см. HTTP .	
least_conn	Новый запрос направляется на сервер с минимальным количеством активных соединений (наименее загруженный). Поддерживается учет веса серверов (weight), медленный старт (slow_start), backup-группы серверов. Подробнее см. TCP/UDP , HTTP .	
random	Запросы распределяются по серверам случайным образом. При большом числе запросов они будут равномерно распределяться по upstream-серверам. Поддерживается учет веса серверов (weight). Подробнее см. TCP/UDP , HTTP .	
least_time	Вероятность передачи соединения активному серверу обратно пропорциональна среднему времени его ответа. Поддерживаются backup-группы серверов. Подробнее см. TCP/UDP , HTTP .	
feedback	Балансировка нагрузки по обратной связи (по произвольному параметру из ответа на основной или проверочный запрос). Поддерживается учет веса серверов (weight), медленный старт (slow_start), backup-группы серверов. Подробнее см. TCP/UDP , HTTP .	
least_bandwidth	Балансировка на основе наименьшего потребления полосы пропускания. Поддерживается учет веса серверов (weight), медленный старт (slow_start), backup-группы серверов. Подробнее см. TCP/UDP , HTTP .	
least_packages	Балансировка на основе наименьшего числа пакетов в единицу времени. Может использоваться в случаях, когда оборудование, расположенное за балансировщиком, не способно "переварить" слишком большой поток пакетов. Поддерживается учет веса серверов (weight), медленный старт (slow_start), backup-группы серверов. Подробнее см. TCP/UDP .	

3.1.4 Примеры HTTP-балансировки нагрузки

Базовая настройка

Самая простая конфигурация для балансировки нагрузки с помощью Angie ADC может выглядеть следующим образом:

```
http {
    upstream myapp1 {
        server srv1.example.com;
        server srv2.example.com;
        server srv3.example.com;
    }
    server {
```

```
listen 80;
location / {

    proxy_pass http://myapp1;
}
}
```

В приведенном примере:

- Группа серверов `myapp1` содержит три сервера `srv1`, `srv2` и `srv3`.
- На этих серверах работают три экземпляра одного и того же приложения.
- Используется метод балансировки `round-robin` (по умолчанию).
- Все запросы на порт 80 перенаправляются через группу серверов `myapp1`.

`least_conn`

Балансировка нагрузки по наименее загруженному серверу.

Модуль балансировки нагрузки `least_conn` в модуле HTTP позволяет распределять нагрузку более равномерно, направляя новые запросы на серверы с наименьшим количеством активных соединений. Этот метод рекомендуется, если время обработки запросов может сильно варьироваться.

Пример конфигурации:

```
http {

    upstream backend {

        least_conn;

        server backend1.example.com;
        server backend2.example.com;
        server backend3.example.com;
    }

    server {

        listen 80;
        location / {

            proxy_pass http://backend;
        }
    }
}
```

В этом примере:

- Директива `least_conn` включает метод балансировки нагрузки по наименьшему числу соединений.
- Для балансировки нагрузки определены три проксируемых сервера.
- Входящие запросы к серверу проксируются в группу `backend`.

Преимущества метода:

- Динамическая балансировка: непрерывно отслеживает количество активных соединений для адаптации к текущей нагрузке на каждом сервере.

- Простота: работает без необходимости дополнительных параметров или сложных настроек.
- Совместимость: совместим с другими конфигурациями проксируемых серверов, включая веса, максимальное количество сбоев и время ожидания после сбоя.

ip_hash

Балансировка нагрузки с сохранением сессий.

Методы `round-robin` и `least-conn` не гарантируют, что запросы от одного клиента будут обрабатываться одним и тем же сервером. Если важно, чтобы клиент всегда направлялся на один сервер (например, при использовании сессионных данных), можно использовать метод `ip_hash`.

Пример конфигурации:

```
upstream myapp1 {
    ip_hash;
    server srv1.example.com;
    server srv2.example.com;
    server srv3.example.com;
}
```

Метод `ip_hash` гарантирует, что запросы от одного клиента будут попадать на один сервер, если этот сервер доступен.

weight

Взвешенная балансировка нагрузки.

С помощью директивы `weight` можно указать, какой сервер должен обрабатывать больше запросов. Этот способ рекомендуется, если серверы имеют разную производительность. По умолчанию вес сервера равен 1.

Пример:

```
upstream myapp1 {
    server srv1.example.com weight=3;
    server srv2.example.com;
    server srv3.example.com;
}
```

В этой конфигурации каждые пять запросов будут распределены следующим образом: три запроса попадут на `srv1`, один запрос — на `srv2` и еще один — на `srv3`.

Метод `weight` может использоваться в комбинации с методами `round-robin` и `least-conn`.

least_time

Балансировка нагрузки на основе наименьшего времени ответа.

Модуль балансировки нагрузки `least_time` задает для группы метод балансировки нагрузки, при котором вероятность передачи соединения активному серверу обратно пропорциональна среднему времени его ответа; чем оно меньше, тем больше соединений будет получать сервер.

Когда использовать `least_time`?

- Переменная нагрузка: запросы распределяются на серверы, которые обрабатывают их быстрее.

- Высокая доступность: уменьшается время ожидания для пользователей.
- Гетерогенные серверы: если серверы имеют разное оборудование или настройки, `least_time` помогает уравновесить нагрузку.

Пример конфигурации:

```
http {
    upstream backend {
        least_time header;
        server backend1.example.com;
        server backend2.example.com;
        server backend3.example.com;
    }
    server {
        listen 80;
        location / {
            proxy_pass http://backend;
        }
    }
}
```

В этом примере `least_time header`: балансировка основана на времени отклика до получения заголовка ответа от сервера.

feedback

Балансировка по произвольному параметру из ответа на основной или проверочный запрос.

Модуль балансировки нагрузки **feedback** Задаёт в `upstream` механизм балансировки нагрузки по обратной связи. Он динамически корректирует решения при балансировке, умножая вес каждого проксируемого сервера на среднее значение обратной связи, которое меняется с течением времени в зависимости от значения переменной и подчиняется необязательному условию. По умолчанию параметр `factor` равен 90.

Пример конфигурации:

```
upstream backend {
    zone backend 1m;

    feedback $feedback_value factor=80 account=$condition_value;

    server backend1.example.com;
    server backend2.example.com;
}

map $upstream_http_custom_score $feedback_value {
    "high" 100;
    "medium" 75;
    "low" 50;
    default 10;
}

map $upstream_probe $condition_value {
```



```
"high_priority" "1";
"low_priority" "0";
default "1";
}
```

Эта конфигурация категоризирует ответы серверов по уровням обратной связи на основе определенных оценок из полей заголовков ответа, а также добавляет условие на `$upstream_probe`, чтобы учитывать только ответы от активной проверки `high_priority` или ответы на обычные клиентские запросы.

3.1.5 Примеры TCP/UDP-балансировки нагрузки

Конфигурация для TCP

```
stream {

    upstream tcp_backend {

        server 192.168.1.10:3306 weight=2;
        server 192.168.1.11:3306;
        server 192.168.1.12:3306;
    }

    server {

        listen 3306;
        proxy_pass tcp_backend;
        proxy_timeout 10s;
        proxy_connect_timeout 5s;
    }
}
```

В этой конфигурации задана группа серверов, участвующих в балансировке (`tcp_backend`). Настроен вес для серверов, чтобы более мощный сервер получал больше запросов (`weight=2`). Сервер слушает входящие соединения на порту 3306 (используется для MySQL). Трафик перенаправляется к группе серверов `tcp_backend`. Настроены тайм-ауты:

- максимальное время ожидания ответа от сервера 10s;
- максимальное время для установления соединения с сервером 5s.

Конфигурация для UDP

Для приложений, работающих через протокол UDP, конфигурация аналогична TCP, но с настройками для UDP-трафика.

```
stream {

    upstream udp_backend {

        server 192.168.1.20:53;
        server 192.168.1.21:53;
    }

    server {
```

```
listen 53 udp;
proxy_pass udp_backend;

proxy_responses 1;
proxy_timeout 10s;
}
}
```

least_bandwidth

Балансировка на основе наименьшего потребления полосы пропускания.

Алгоритм периодически вычисляет среднее использование полосы пропускания для каждого апстрим-сервера, используя формулу скользящего среднего для сглаживания колебаний со временем. Когда начинается новая сессия, модуль балансировки нагрузки `least_bandwidth` выбирает проксируемый сервер с наименьшим средним использованием пропускной способности, скорректированным с учетом веса сервера, если он указан. Этот механизм гарантирует, что сессии направляются на серверы, которые в настоящее время используют меньше пропускной способности, что способствует равномерному распределению сетевой нагрузки.

Пример конфигурации:

```
stream {

    upstream backend {

        least_bandwidth both factor=80;
        server backend1.example.com:12345;
        server backend2.example.com:12345;
    }
    server {

        listen 12345;
        proxy_pass backend;
    }
}
```

В этом примере:

- В директиве `least_bandwidth` настроен учет как входящей, так и исходящей пропускной способности.
- Коэффициент сглаживания установлен равным 80, что влияет на скорость адаптации среднего значения к изменениям в использовании пропускной способности.
- Для балансировки нагрузки определены два проксируемых сервера.

Преимущества метода:

- Динамическая балансировка: непрерывно отслеживает использование пропускной способности для адаптации к текущей нагрузке на каждом сервере.
- Коэффициент сглаживания `factor` контролирует отзывчивость расчета скользящего среднего; более высокий коэффициент означает более медленную адаптацию к изменениям.
- Режимы: позволяет выбирать балансировку на основе входящей пропускной способности, исходящей пропускной способности или обоих вариантов, в зависимости от потребностей приложения.
- Совместим с другими конфигурациями проксируемых серверов, включая веса, максимальное количество сбоя и время ожидания после сбоя.

least_packets

Балансировка на основе наименьшего числа пакетов в единицу времени.

Средняя скорость пакетов для каждого проксируемого сервера периодически проверяется с использованием формулы скользящего среднего для сглаживания колебаний со временем. Когда инициируется новый сеанс, модуль балансировки нагрузки **least_packets** в потоковом модуле выбирает проксируемый сервер с наименьшей средней скоростью передачи пакетов, скорректированной с учетом веса сервера. Этот процесс гарантирует, что сеансы направляются на серверы, которые в настоящее время наименее загружены с точки зрения обработки пакетов, что способствует равномерному распределению сетевой нагрузки.

Пример конфигурации:

```
stream {
    upstream backend {
        least_packets both factor=80;
        server backend1.example.com:12345;
        server backend2.example.com:12345;
    }
    server {
        listen 12345;
        proxy_pass backend;
    }
}
```

В этом примере:

- В директиве **least_packets** настроен учет как входящих, так и исходящих пакетов.
- Коэффициент сглаживания установлен равным 80, что влияет на то, как быстро среднее значение адаптируется к изменениям скорости пакетов.
- Определены два проксируемых сервера для балансировки нагрузки.

Преимущества метода:

- Динамическая балансировка: непрерывно отслеживает скорость пакетов для адаптации к текущей нагрузке на каждом сервере.
- Параметр **factor** контролирует отзывчивость расчета скользящего среднего; более высокий коэффициент означает более медленную адаптацию к изменениям.
- Режимы: позволяет выбирать балансировку на основе числа входящих пакетов, исходящих пакетов или обоих значений.
- Совместим с другими конфигурациями проксируемых серверов, включая веса, максимальное количество сбоев и время ожидания после сбоя.

hash

Контекстное хэширование.

Поддерживается метод **hash** (например, по IP или произвольному значению). Если сервер из пула станет недоступным или будет добавлен новый сервер, минимальное количество клиентов перенаправится на другие серверы. Используется для "лишних сессий", когда важно, чтобы запросы от одного клиента всегда обрабатывались одним и тем же сервером.

Пример:

```
upstream tcp_backend {
    hash $remote_addr consistent;
    server 192.168.1.10:3306;
    server 192.168.1.11:3306;
}
```

Ограничение IP

Для защиты можно использовать ограничения IP через **allow** и **deny**.

Пример:

```
server {
    listen 3306;
    allow 192.168.1.0/24;
    deny all;
    proxy_pass tcp_backend;
}
```

В этом примере разрешена только локальная сеть, остальным доступ запрещен.

3.2 Проверки работоспособности серверов

Angie ADC включает пассивные и активные проверки работоспособности серверов (health probes). Если сервер начинает отвечать с ошибками, Angie ADC временно исключает его из пула доступных серверов.

3.2.1 Пассивные проверки

Пассивные проверки работают автоматически на основе реальных клиентских запросов.

Пример:

```
upstream myapp1 {
    server srv1.example.com max_fails=2 fail_timeout=10s;
    server srv2.example.com;
    server srv3.example.com;
}
```

В этой конфигурации сервер помечается как недоступный после двух неудачных запросов подряд (**max_fails=2**). Через десять секунд сервер снова проверяется на доступность (**fail_timeout=10s**). Если он восстановился, запросы снова начинают отправляться на этот сервер.

3.2.2 Активные проверки

Активные проверки позволяют оценить доступность сервера без участия клиентов. Поддерживаются активные проверки для HTTP и TCP / UDP. Проверки задаются с помощью директивы `upstream_probe`. Для одного апстрима можно определить несколько проверок. Можно гибко настраивать поведение проверок. Поддерживается отправка ICMP echo-запросов (ping). Подробнее см. справочные материалы по активным проверкам [HTTP](#) и [stream](#).

Пример для HTTP

```
upstream backend {
    zone backend 1m;

    server backend1.example.com;
    server backend2.example.com;
}

map $upstream_status $good {
    200      "1";
}

server {
    listen ...;

    location /backend {
        ...
        proxy_pass http://backend;

        upstream_probe backend_probe
            uri=/probe
            port=10004
            interval=5s
            test=$good
            essential
            persistent
            fails=3
            passes=3
            max_body=10m
            mode=idle;
    }
}
```

Детали работы:

- Изначально сервер не получает клиентские запросы, пока не пройдет *все* заданные для него проверки с параметром **essential** (пропуская помеченные как **persistent**, если конфигурация перезагружена и до этого сервер считался работающим). Если таких проверок нет, сервер считается работающим.
- Сервер считается неработающим и не получает клиентские запросы, если *какая-либо* заданная для него проверка достигает своего порога **fails** или сам сервер достигает порога **max_fails**.
- Чтобы неработающий сервер снова мог считаться работающим, *все* заданные для него проверки должны достичь своего порога **passes**; после этого учитывается порог **max_fails**.

Пример для TCP/UDP

```
upstream backend {
    zone backend 1m;

    server a.example.com;
    server b.example.com;
}

map $upstream_probe_response $good {
    ~200    "1";
    default  "";
}

server {
    listen ...;

    # ...
    proxy_pass backend;
    upstream_probe_timeout 1s;

    upstream_probe backend_probe
        port=12345
        interval=5s
        test=$good
        essential
        persistent
        fails=3
        passes=3
        max_response=512k
        mode=onfail
        "send=data:GET / HTTP/1.0\r\n\r\n";
}
```

Примечание

Согласно спецификации RFC 2616 (HTTP/1.1) и RFC 9110 (HTTP Semantics), заголовки HTTP должны разделяться последовательностью CRLF (`\r\n`), а не просто `\n`.

Детали работы:

- Изначально сервер не получает клиентские запросы, пока не пройдет *все* заданные для него проверки с параметром **essential** (пропуская помеченные как **persistent**, если конфигурация перезагружена и до этого сервер считался работающим). Если таких проверок нет, сервер считается работающим.
- Сервер считается неработающим и не получает клиентские запросы, если *какая-либо* заданная для него проверка достигает своего порога **fails** или сам сервер достигает порога **max_fails**.
- Чтобы неработающий сервер снова мог считаться работающим, *все* заданные для него проверки должны достичь своего порога **passes**; после этого учитывается порог **max_fails**.

Пример ICMP echo-запроса для HTTP

```
http {
    upstream u {
        zone z 1k;

        server 127.0.0.1:8081;
        server 127.0.0.1:8082;
    }

    server {
        listen ...;

        location / {
            proxy_pass http://u/;

            upstream_probe simple1
                interval=3s
                ping;

            upstream_probe simple2
                interval=3s
                ping ping_timeout=3s;
        }
    }
}
```

Детали работы:

- Используется проверка методом ICMP Echo вместо HTTP-запроса.
- ICMP-проверка проверяет доступность IP-адреса 127.0.0.1. Порты `upstream` игнорируются.
- Сервер считается неработающим и не получает клиентские запросы, если не отвечает на запрос ICMP Echo в течение заданного таймаута. Первая проверка использует таймаут ожидания по умолчанию (1 секунда). Вторая проверка задает таймаут ожидания ответа явно (3 секунды).
- Единственным критерием прохождения проверки является успешное получение ICMP-ответа в пределах таймаута.

3.3 Резервирование проксируемых серверов

В Angie ADC поддерживается использование нескольких backup-групп проксируемых серверов .

Уровень резервирования backup-группы можно задать с помощью директивы `backup=n` в блоках `stream` и `http`. Например `backup=1` (или просто `backup`) — группа резервных серверов первого уровня, (в API отображается как `true`), `backup=2` — группа резервных серверов второго уровня, (в API отображается как соответствующее число).

Для SRV-записей уровень резервирования группы можно задать двумя способами:

- С помощью директивы `backup=prio` как абсолютное значение.
В этом случае значение приоритета `priority=n` из SRV-записи будет использовано в качестве уровня резервной группы: `backup=n`.
- Через относительное значение приоритета `priority=n` из SRV-записи.
В этом случае директива `backup=n` задает базовый уровень отсчета (например `backup=3`), а относительные уровни резервирования отсчитываются от этого уровня. Сервер с наименьшим

значением `priority` получит уровень `backup=3`, следующий за ним — `backup=4` и так далее. Если уровень отсчета `backup=n` не задан, отсчет будет идти от нуля: сервер с наименьшим значением `priority` получит `backup=0` и станет основным, за ним — `backup=1`, резервный, и так далее.

Директива `backup_switch permanent` дает возможность `backup`-группе проксируемых HTTP-серверов оставаться активной, когда серверы из основной группы стали снова доступными. Например, вы можете указать `permanent=2m`, чтобы балансировщик нагрузки не пытался вернуться на основную группу чаще, чем раз в две минуты. Если указать параметр `permanent` без значения, то балансировщик нагрузки не будет пытаться вернуться на основную группу после перехода на резервную. Функциональность `backup_switch permanent` реализована для `http` и `stream`.

Пример:

```
upstream my_backend {
    server primary1.example.com;
    server primary2.example.com;

    server backend.example.com service=http resolve backup=prio;

    server backup1a.example.com backup;
    server backup1b.example.com backup;

    server backup2a.example.com backup=2;
    server backup2b.example.com backup=2;

    backup_switch permanent=2m;
}
```

3.4 Просмотр и редактирование конфигурации

3.4.1 Просмотр и редактирование конфигурации балансировщика нагрузки

По умолчанию в конфигурационном файле заданы общие настройки, настройки журналов и формат записей; количество соединений, которые может одновременно обрабатывать один рабочий процесс, настройки HTTP, типы файлов, настройки основного сервера и дополнительные маршруты.

Чтобы просмотреть или отредактировать конфигурацию, выполните следующие действия:

1. Откройте консоль Angie ADC.
2. На вкладке **Панель мониторинга** выберите виджет балансировщика нагрузки и нажмите на него.
Откроется экран мониторинга с детальной информацией.
3. Нажмите кнопку **Конфигурация** в верхней правой части экрана.
Откроется конфигурационный файл балансировщика нагрузки.
Вы можете внести правки в текущую конфигурацию или выбрать одну из предыдущих версий, щелкнув ссылку **История** в верхней части экрана.
4. Сохраните файл после внесения изменений, нажав кнопку **Сохранить**.
Изменения будут применены сразу.

3.4.2 Использование портов

Если вы планируете использовать в конфигурации другие порты помимо 80/443, вы можете открыть их с помощью команды *open* через CLI Angie ADC.

Важно

При настройке конфигурации не занимайте следующие порты (они используются внутренними сервисами Angie ADC):

TCP: 22, 25, 53, 2022, 2222, 2601, 2602, 2603, 2604, 2605, 2606, 2615, 2616, 2617, 2619, 2623, 3000, 3050, 3051, 3053, 3081, 3101, 3111, 5432, 8080, 9090, 9898

UDP: 53, 323, 3784, 3785, 4784

3.5 Резервное копирование и восстановление конфигурации

В Angie ADC реализовано резервное копирование и хранение всех версий конфигурационного файла балансировщика нагрузки. При сохранении изменений в текущей конфигурации балансировщика Angie ADC автоматически применяет ее и сохраняет предыдущую версию в репозитории. Вы можете использовать *внутренний или внешний репозиторий*. Можно просмотреть список сохраненных версий конфигурационных файлов и применить нужную версию.

3.5.1 Просмотр списка сохраненных конфигураций

Чтобы просмотреть список сохраненных конфигураций:

1. Откройте консоль Angie ADC.
2. Выберите **Панель мониторинга** → <Имя_балансировщика_нагрузки> → **Конфигурация** → **История**.

Откроется таблица со списком сохраненных конфигурационных файлов. Для каждого файла указано время его создания, версия Angie ADC, статус (рабочая или нерабочая конфигурация), пользовательский комментарий, теги (**angie-lb** и **UUID**) и дата последнего изменения.

3.5.2 Изменение рабочего статуса конфигурации

Чтобы пометить конфигурацию как рабочую или нерабочую:

1. Откройте консоль Angie ADC.
2. Выберите **Панель мониторинга** → <Имя_балансировщика_нагрузки> → **Конфигурация** → **История**.

Откроется таблица со списком сохраненных конфигурационных файлов.

3. Щелкните многоточие (...) напротив конфигурации, статус которой вы хотите изменить, и нажмите **Пометить как нерабочую** или **Пометить как рабочую**.
4. Если вы устанавливаете нерабочий статус, укажите причину поломки и нажмите **Сохранить**. Статус конфигурации будет изменен.

3.5.3 Восстановление последней рабочей версии конфигурации

Чтобы применить последнюю рабочую версию конфигурации:

1. Откройте консоль Angie ADC.
2. Выберите Панель мониторинга → <Имя_балансировщика_нагрузки> → Конфигурация → История.
3. В верхней части экрана нажмите кнопку Применить последнюю рабочую версию.
Будет применена последняя рабочая конфигурация.

3.5.4 Восстановление конфигурации из резервной копии

Чтобы применить произвольную конфигурацию из резервной копии:

1. Откройте консоль Angie ADC.
2. Выберите Панель мониторинга → <Имя_балансировщика_нагрузки> → Конфигурация → История.
Откроется таблица со списком сохраненных конфигурационных файлов.
3. Щелкните многоточие (...) напротив конфигурации, которую вы хотите использовать, и нажмите кнопку Применить.
Конфигурация будет применена сразу.

3.6 Справочная информация

Справочные материалы по настройке балансировщика Angie ADC.

3.6.1 Общие сведения

В этих статьях рассказывается о различных аспектах обработки запросов и взаимодействия с другими серверами.

- *Конфигурационные файлы*
- *Соединения, сессии, запросы, логи*

3.6.2 Основной модуль

Основной модуль	Управление служебными файлами, процессами и другими модулями Angie ADC.

3.6.3 HTTP-модули

<i>HTTP-модули</i>	Обработка HTTP-запросов и ответов, управление HTTP-сервером, соединениями и статическими файлами.

3.6.4 Потокосые модули

<i>Потокосые модули</i>	Обработка HTTP-запросов и ответов, управление HTTP-сервером, соединениями и статическими файлами.

3.6.5 Почтовые модули

<i>Почтовые модули</i>	Функциональность почтового прокси-сервера.

3.6.6 Сторонние модули

<i>GeoIP2</i>	Добавляет поиск по геоданным в базах MaxMind GeoIP2.

3.6.7 Инструкции

Настройка ACME

Конфигурационные файлы

Angie ADC использует текстовый конфигурационный файл. По умолчанию этот файл называется `angie.conf` и обычно находится в директории `/etc/angie`.

Конфигурационный файл обычно состоит из следующих контекстов:

- *events* — общая обработка соединений;
- *http* — HTTP-трафик;
- *stream* — TCP- и UDP-трафик.

Директивы, размещенные вне этих контекстов, считаются находящимися в контексте `main`:

```
user angie; # директива в контексте 'main'

events {
    # конфигурация обработки соединений
}

http {
    # Конфигурация трафика HTTP, для всех вложенных виртуальных серверов

    server {
```

```
# конфигурация виртуального HTTP сервера 1
location /one {

    # конфигурация обработки HTTP запросов с URI, начинающимися с '/one'
}
location /two {

    # конфигурация обработки HTTP запросов с URI, начинающимися с '/two'
}

server {

    # конфигурация виртуального HTTP сервера 2
}

stream {

    # Конфигурация трафика TCP/UDP, для всех вложенных виртуальных серверов
    server {

        # конфигурация виртуального TCP сервера 1
    }
}
```

Для упрощения управления конфигурацией рекомендуется использовать директиву *include* в основном файле `angie.conf`, чтобы ссылаться на содержимое файлов, специфичных для функций:

```
include /etc/angie/http.d/*.conf;
include /etc/angie/stream.d/*.conf;
```

Наследование

В общем случае дочерний контекст (тот, который содержится в другом контексте, который считается родительским) унаследует настройки директив, определенных на уровне родителя. Некоторые директивы могут появляться в нескольких контекстах; в таких случаях вы можете переопределить настройки, унаследованные от родителя, включив директиву в дочерний контекст.

Синтаксис

Единицы измерения

Размеры можно указывать в следующих единицах:

Без суффикса	Байты
k, K	Килобайты
m, M	Мегабайты
g, G	Гигабайты

Например: 1024, 8k, 1m, 16g.

Интервалы времени можно указывать в миллисекундах, секундах, минутах, часах, днях и так далее, с использованием следующих суффиксов:

ms	Миллисекунды
s	Секунды
m	Минуты
h	Часы
d	Дни
w	Недели
M	Месяцы (принято считать равными 30 дням)
y	Годы (принято считать равными 365 дням)

Несколько единиц могут быть объединены в одном значении, указывая их в порядке от наиболее значимого к наименее значимому, при необходимости разделяя пробелами. Например, "1h 30m" обозначает тот же промежуток времени, что и "90m" или "5400s". Значение без суффикса интерпретируется как секунды. Рекомендуется всегда указывать суффикс.

Некоторые интервалы времени могут быть указаны только с разрешением в секундах.

Директивы

Каждая директива состоит из имени и набора параметров. Если какая-либо часть директивы должна содержать пробелы, она должна быть заключена в кавычки или экранирована:

```
add_header X-MyHeader "foo bar";
add_header X-MyHeader foo\ bar;
```

Если именованный параметр требует пробелов и вы используете кавычки, его имя также должно быть заключено в кавычки:

```
server example.com "sid=server 1";
```

Настройка хэшей

Для эффективной обработки статических наборов данных, таких как имена серверов, значения директивы *map*, MIME-типы и имена заголовков запросов, Angie ADC использует хэш-таблицы. При запуске и каждом переопределении конфигурации Angie ADC определяет оптимальный размер для этих хэш-таблиц, чтобы размер корзины, которая хранит ключи с одинаковыми хэш-значениями, не превышал заданный параметр (*hash bucket size*). Размер таблицы измеряется в корзинах и корректируется до тех пор, пока не превысит параметр *hash max size*. Большинство хэш-таблиц имеют соответствующие директивы для настройки этих параметров, такие как *server_names_hash_max_size* и *server_names_hash_bucket_size* для имен серверов.

Параметр *hash bucket size* выравнивается по кратности размера линии кэша процессора. Такое выравнивание улучшает эффективность поиска ключей на современных процессорах, уменьшая количество обращений к памяти. Если *hash bucket size* равен размеру одной линии кэша, максимальное количество обращений к памяти во время поиска ключа будет два: одно для вычисления адреса корзины и второе для поиска внутри корзины. Поэтому, если Angie ADC сообщает, что следует увеличить либо *hash max size*, либо *hash bucket size*, начните с увеличения *hash max size*.

Перезагрузка конфигурации

Перезагрузка конфигурационного файла происходит автоматически при сохранении изменений в конфигурации балансировщика нагрузки. Подробнее см. [Просмотр и редактирование конфигурации](#).

Соединения, сессии, запросы, логи

Механизмы обработки соединений

Angie поддерживает различные методы обработки соединений. Доступность конкретного метода зависит от используемой платформы. На платформах, поддерживающих несколько методов, Angie обычно автоматически выбирает наиболее эффективный метод. Однако, при необходимости, метод обработки соединений можно явно выбрать с помощью директивы `use`.

Доступны следующие методы обработки соединений:

Метод	Описание
<code>select</code>	Стандартный метод. Соответствующий модуль собирается автоматически на платформах, не имеющих более эффективных методов.
<code>poll</code>	Стандартный метод. Соответствующий модуль собирается автоматически на платформах, не имеющих более эффективных методов.
<code>kqueue</code>	Эффективный метод, доступный на FreeBSD 4.1+, OpenBSD 2.9+, NetBSD 2.0 и macOS.
<code>epoll</code>	Эффективный метод, доступный на Linux 2.6+.
<code>/dev/poll</code>	Эффективный метод, доступный на Solaris 7 11/99+, HP/UX 11.22+ (eventport), IRIX 6.5.15+ и Tru64 UNIX 5.1A+.
<code>eventport</code>	Метод <code>event ports</code> доступен на Solaris 10+. (Из-за известных проблем рекомендуется использовать метод <code>/dev/poll</code> .)

Обработка HTTP-запросов

Каждый HTTP-запрос проходит через ряд фаз, на каждой из которых выполняется определенный тип обработки.

Post-read	Начальная фаза. Модуль <i>RealIP</i> вызывается на этой фазе.
Server-rewrite	Фаза, на которой обрабатываются директивы из модуля <i>Rewrite</i> , определенные в блоке <code>server</code> (но вне блока <code>location</code>).
Find-config	Специальная фаза, на которой выбирается <i>location</i> на основе URI запроса.
Rewrite	Похожа на фазу <i>Server-rewrite</i> , но применяется к правилам <i>rewrite</i> , определенным в блоке <code>location</code> , выбранном на предыдущей фазе.
Post-rewrite	Специальная фаза, на которой запрос перенаправляется в новое место, как на фазе <i>Find-config</i> , если его URI был изменен во время фазы <i>Rewrite</i> .
Preaccess	На этой фазе стандартные модули Angie, такие как <i>Limit Req</i> , регистрируют свои обработчики.
Access	Фаза, на которой проверяется право клиента на выполнение запроса, обычно с помощью стандартных модулей Angie, таких как <i>Auth Basic</i> .
Post-access	Специальная фаза, на которой обрабатывается директива <i>satisfy any</i> .
Precontent	На этой фазе стандартные модули, такие как директивы <i>try_files</i> и <i>mirror</i> , регистрируют свои обработчики.
Content	Фаза, на которой обычно генерируется ответ. На этом этапе несколько стандартных модулей Angie регистрируют свои обработчики, включая <i>Index</i> ; также сюда относятся директивы <i>proxy_pass</i> , <i>fastcgi_pass</i> , <i>uwsgi_pass</i> , <i>scgi_pass</i> и <i>grpc_pass</i> . Обработчики вызываются последовательно, пока один из них не произведет вывод.
Log	Финальная фаза, на которой выполняется логирование запросов. В настоящее время только модуль <i>Log</i> регистрирует свой обработчик на этом этапе для ведения журнала доступа.

Обработка TCP/UDP-сессий

TCP/UDP-сессия от клиента проходит через ряд фаз, на каждой из которых выполняется определенный тип обработки:

Post-accept	Начальная фаза после принятия соединения от клиента. На этой фазе вызывается модуль <i>RealIP</i> .
Pre-access	Предварительная фаза для проверки доступа. Модули <i>Set</i> вызываются на этой фазе.
Access	Фаза для ограничения доступа клиента перед фактической обработкой данных. На этом этапе вызывается модуль <i>Access</i> .
SSL	Фаза, на которой происходит терминация TLS/SSL. На этой фазе вызывается модуль <i>SSL</i> .
Preread	Фаза для чтения начальных байт данных в <i>preread buffer</i> , чтобы позволить таким модулям, как <i>SSL Preread</i> , проанализировать данные до их обработки.
Content	Обязательная фаза, на которой данные фактически обрабатываются, обычно с участием модуля <i>Return</i> , который отправляет ответ клиенту. Директива <i>proxy_pass</i> также относится сюда.
Log	Финальная фаза, на которой фиксируется результат обработки сессии клиента. На этой фазе вызывается модуль <i>Log</i> .

Обработка запросов

Выбор виртуального сервера

Изначально соединение создается в контексте сервера по умолчанию. Имя сервера может быть определено на следующих этапах обработки запроса, каждый из которых участвует в выборе конфигурации сервера:

- Во время SSL-рукопожатия, заранее, в соответствии с SNI.
- После обработки строки запроса.
- После обработки поля заголовка `Host`.

Если имя сервера не определено после обработки строки запроса или поля заголовка `Host`, Angie использует пустое имя в качестве имени сервера.

На каждом из этих этапов могут применяться различные конфигурации сервера. Поэтому некоторые директивы следует указывать с осторожностью:

- В случае директивы `ssl_protocols` список протоколов устанавливается библиотекой OpenSSL до применения конфигурации сервера в соответствии с именем, запрошенным через SNI. В результате протоколы следует указывать только для сервера по умолчанию.
- Директивы `client_header_buffer_size` и `merge_slashes` применяются до чтения строки запроса. Поэтому эти директивы используют либо конфигурацию сервера по умолчанию, либо конфигурацию сервера, выбранную через SNI.
- В случае директив `ignore_invalid_headers`, `large_client_header_buffers` и `underscores_in_headers`, которые участвуют в обработке полей заголовков запроса, конфигурация сервера дополнительно зависит от того, была ли она обновлена в соответствии со строкой запроса или полем заголовка `Host`.
- Ответ с ошибкой обрабатывается с использованием директивы `error_page` на сервере, который в данный момент обрабатывает запрос.

Виртуальные серверы на основе имен

Сначала Angie ADC определяет, какой сервер должен обрабатывать запрос. Рассмотрим простую конфигурацию, где все три виртуальных сервера слушают на порту 80:

```
server {
    listen 80;
    server_name example.org www.example.org;
    # ...
}

server {
    listen 80;
    server_name example.net www.example.net;
    # ...
}

server {
    listen 80;
    server_name example.com www.example.com;
    # ...
}
```


В этой конфигурации Angie ADC определяет, какой сервер должен обработать запрос, основываясь исключительно на поле заголовка `Host`. Если значение этого заголовка не совпадает ни с одним из имен серверов или если запрос не содержит этого заголовка, Angie ADC направит запрос к серверу по умолчанию для этого порта. В приведенной выше конфигурации сервером по умолчанию является первый — что является стандартным поведением Angie ADC. Также можно явно указать, какой сервер должен быть сервером по умолчанию, используя параметр `default_server` в директиве `listen`:

```
server {
    listen 80 default_server;
    server_name example.net www.example.net;
    # ...
}
```

Примечание

Обратите внимание, что сервер по умолчанию является свойством слушающего сокета, а не имени сервера.

Международные имена

Международные доменные имена (IDNs) должны указываться в директиве `server_name` с использованием представления ASCII (Punycode):

```
server {
    listen 80;
    server_name xn--e1afmkfd.xn--80akhbyknj4f; # пример.испытание
    # ...
}
```

Запрет запросов с неопределенными именами серверов

Если запросы без заголовка `Host` нежелательны, можно определить сервер, который просто отклоняет такие запросы:

```
server {
    listen 80;
    server_name "";
    return 444;
}
```

В этой конфигурации имя сервера задано пустой строкой, что соответствует запросам без заголовка `Host`. Затем возвращается специальный нестандартный код 444, который закрывает соединение.

Сочетание виртуальных серверов, основанных на именах и IP-адресах

Рассмотрим более сложную конфигурацию, где некоторые виртуальные серверы слушают на разных адресах:

```
server {

    listen 192.168.1.1:80;
    server_name example.org www.example.org;
    # ...
}

server {

    listen 192.168.1.1:80;
    server_name example.net www.example.net;
    # ...
}

server {

    listen 192.168.1.2:80;
    server_name example.com www.example.com;
    # ...
}
```

В этой конфигурации Angie ADC сначала проверяет IP-адрес и порт запроса по директивам *listen* блоков *server*. Затем он проверяет поле заголовка *Host* запроса по записям *server_name* блоков *server*, которые совпали с IP-адресом и портом. Если имя сервера не найдено, запрос будет обработан сервером по умолчанию. Например, запрос к **www.example.com**, полученный на порту 192.168.1.1:80, будет обработан сервером по умолчанию для этого порта — т.е. первым сервером — поскольку **www.example.com** не определен для этого порта.

Как упоминалось ранее, сервер по умолчанию является свойством слушающего сокета, и можно определить разные серверы по умолчанию для разных портов:

```
server {

    listen 192.168.1.1:80;
    server_name example.org www.example.org;
    # ...
}

server {

    listen 192.168.1.1:80 default_server;
    server_name example.net www.example.net;
    # ...
}

server {

    listen 192.168.1.2:80 default_server;
    server_name example.com www.example.com;
    # ...
}
```

Выбор локаций

Рассмотрим простую конфигурацию PHP-сайта:

```
server {

    listen 80;
    server_name example.org www.example.org;
    root /data/www;

    location / {

        index index.html index.php;
    }

    location ~* \.(gif|jpg|png)$ {

        expires 30d;
    }

    location ~ \.php$ {

        fastcgi_pass localhost:9000;
        fastcgi_param SCRIPT_FILENAME
        $document_root$fastcgi_script_name;
        include fastcgi_params;
    }

}
```

Angie сначала ищет наиболее конкретный префикс `location`, заданный буквальными строками, независимо от их порядка в списке. В приведенной выше конфигурации единственный префикс локации — это `location /`, который соответствует любому запросу и будет использоваться в качестве последнего средства. Затем Angie проверяет локации, определенные с помощью регулярных выражений, в порядке их появления в конфигурационном файле. Первое совпадающее выражение завершает поиск, и Angie использует эту `location`. Если ни одно регулярное выражение не совпадает с запросом, Angie использует наиболее конкретную префиксную `location`, найденную ранее.

Примечание

Локации всех типов проверяют только URI часть строки запроса, исключая аргументы. Это связано с тем, что аргументы в строке запроса могут быть указаны разными способами, например:

- `/index.php?user=john&page=1`
- `/index.php?page=1&user=john`

Кроме того, строка запроса может содержать любое количество параметров:

- `/index.php?page=1&something+else&user=john`

Теперь давайте рассмотрим, как запросы будут обрабатываться в приведенной выше конфигурации:

- Запрос `/logo.gif` сначала соответствует префиксу `location /`, а затем регулярному выражению `\.(gif|jpg|png)$`. Поэтому он обрабатывается последней локацией. Используя директиву `root /data/www`, запрос сопоставляется с файлом `/data/www/logo.gif`, и файл отправляется клиенту.

- Запрос `/index.php` также изначально соответствует префиксу `location /`, а затем регулярному выражению `.(php)$`. Следовательно, он обрабатывается последней локацией, и запрос передается серверу FastCGI, слушающему на `localhost:9000`. Директива `fastcgi_param` устанавливает параметр FastCGI `SCRIPT_FILENAME` в `/data/www/index.php`, и сервер FastCGI выполняет файл. Переменная `$document_root` устанавливается в значение директивы `root`, а переменная `$fastcgi_script_name` устанавливается в URI запроса, т.е. `/index.php`.
- Запрос `/about.html` соответствует только префиксу `location /`, поэтому он обрабатывается в этой локации. Используя директиву `root /data/www`, запрос сопоставляется с файлом `/data/www/about.html`, и файл отправляется клиенту.

Обработка запроса `/` более сложна. Он соответствует только префиксу `location /`, поэтому он обрабатывается в этой локации. Директива `index` затем проверяет наличие индексных файлов в соответствии с ее параметрами и директивой `root /data/www`. Если файл `/data/www/index.html` не существует, но файл `/data/www/index.php` существует, директива выполняет внутреннюю переадресацию на `/index.php`, и Angie снова ищет локацию, как если бы запрос был отправлен клиентом. Как упоминалось ранее, переадресованный запрос в конечном итоге будет обработан сервером FastCGI.

Проксирование и балансировка нагрузки

Одним из распространенных способов использования Angie ADC является настройка в качестве прокси-сервера. В этой роли Angie ADC принимает запросы, перенаправляет их на проксируемые серверы, получает ответы от этих серверов и отправляет ответы обратно клиентам.

Простой прокси-сервер:

```
server {
    location / {
        proxy_pass http://backend:8080;
    }
}
```

Директива `proxy_pass` заставит Angie ADC передавать запросы клиентов на сервер `backend:8080` (прокси-сервер). Существует множество дополнительных *директив*, которые можно использовать для дальнейшей настройки прокси-соединения.

Проксирование FastCGI

Angie ADC может использоваться для маршрутизации запросов на FastCGI-серверы, которые выполняют приложения, построенные с использованием различных фреймворков и языков программирования, таких как PHP.

Простейшая конфигурация Angie ADC для работы с FastCGI-сервером включает использование директивы `fastcgi_pass` вместо директивы `proxy_pass`, а также директив `fastcgi_param` для установки параметров, передаваемых FastCGI-серверу. Предположим, что FastCGI-сервер доступен по адресу `localhost:9000`. В PHP параметр `SCRIPT_FILENAME` используется для определения имени скрипта, а параметр `QUERY_STRING` используется для передачи параметров запроса. Результирующая конфигурация будет следующей:

```
server {
    location / {
        fastcgi_pass localhost:9000;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        fastcgi_param QUERY_STRING $query_string;
    }
}
```

```

    }

    location ~ \.(gif|jpg|png)$ {

        root /data/images;
    }
}

```

Эта конфигурация настраивает сервер, который направляет все запросы, кроме запросов к статическим изображениям, на проксируемый сервер, работающий на `localhost:9000` через протокол FastCGI.

Проксирование WebSocket

Для обновления соединения с HTTP/1.1 до WebSocket используется механизм [протокольного переключения](#), доступный в HTTP/1.1.

Однако есть тонкость: поскольку заголовок `Upgrade` является [заголовком перехода](#), он не передается от клиента к проксируемому серверу. При использовании прямого проксирования клиенты могут использовать метод `CONNECT` для обхода этой проблемы. Этот подход не работает при обратном проксировании, так как клиенты не осведомлены о каких-либо прокси-серверах, и требуется специальная обработка на прокси-сервере.

Angie ADC реализует специальный режим работы, который позволяет установить туннель между клиентом и проксируемым сервером, если проксируемый сервер возвращает ответ с кодом 101 (Switching Protocols), а клиент запрашивает переключение протокола через заголовок `Upgrade` в запросе.

Как уже упоминалось, заголовки перехода, включая `Upgrade` и `Connection`, не передаются от клиента к проксируемому серверу. Поэтому, чтобы проксируемый сервер был осведомлен о намерении клиента переключиться на протокол WebSocket, эти заголовки должны быть переданы явно:

```

location /chat/ {

    proxy_pass http://backend;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
}

```

Более сложный пример демонстрирует, как значение поля заголовка `Connection` в запросе к проксируемому серверу зависит от наличия поля `Upgrade` в заголовке запроса клиента:

```

http {

    map $http_upgrade $connection_upgrade {

        default upgrade;
        '' close;
    }

    server {

        ...

        location /chat/ {

            proxy_pass http://backend;

```

```

        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection $connection_upgrade;
    }
}

```

По умолчанию соединение будет закрыто, если проксируемый сервер не передает никаких данных в течение 60 секунд. Этот тайм-аут можно увеличить с помощью директивы `proxy_read_timeout`. В качестве альтернативы: на проксируемом сервере можно настроить периодическую отправку кадров WebSocket ping для сброса тайм-аута и проверки того, активно ли соединение.

Балансировка нагрузки

Балансировка нагрузки между несколькими экземплярами приложения — это широко используемая техника для оптимизации использования ресурсов, максимизации пропускной способности, уменьшения задержек и обеспечения отказоустойчивых конфигураций.

Angie ADC можно использовать в качестве высокоэффективного HTTP-балансировщика нагрузки для распределения трафика между несколькими серверами приложений, тем самым улучшая производительность, масштабируемость и надежность веб-приложений.

Самая простая конфигурация для балансировки нагрузки с помощью Angie ADC может выглядеть следующим образом:

```

http {

    upstream myapp1 {

        server srv1.example.com;
        server srv2.example.com;
        server srv3.example.com;
    }

    server {

        listen 80;

        location / {

            proxy_pass http://myapp1;
        }
    }
}

```

В приведенном примере три экземпляра одного и того же приложения работают на серверах с `srv1` по `srv3`. Когда метод балансировки нагрузки не настроен явно, по умолчанию используется круговой метод (`round-robin`). Другие поддерживаемые механизмы балансировки нагрузки включают: `weight`, `least_conn` и `ip_hash`. Реализация обратного прокси в Angie также поддерживает встроенные (или пассивные) проверки состояния серверов. Эти проверки настраиваются с помощью директив `max_fails` и `fail_timeout` внутри блока `server` в контексте `upstream`.

Логирование

Syslog

Директивы `error_log` и `access_log` поддерживают логирование в `syslog`. Для настройки логирования в `syslog` используются следующие параметры:

<code>server=address</code>	Указывает адрес сервера <code>syslog</code> . Адрес может быть доменным именем или IP-адресом с необязательным портом, либо путем к UNIX-доменному сокету, указанным после префикса <code>"unix:"</code> . Если порт не указан, используется UDP-порт 514. Если доменное имя разрешается в несколько IP-адресов, используется первый разрешенный адрес.
<code>facility=string</code>	Устанавливает уровень для сообщений <code>syslog</code> , как определено в RFC 3164 . Возможные уровни включают: <code>"kern"</code> , <code>"user"</code> , <code>"mail"</code> , <code>"daemon"</code> , <code>"auth"</code> , <code>"intern"</code> , <code>"lpr"</code> , <code>"news"</code> , <code>"uucp"</code> , <code>"clock"</code> , <code>"authpriv"</code> , <code>"ftp"</code> , <code>"ntp"</code> , <code>"audit"</code> , <code>"alert"</code> , <code>"cron"</code> , <code>"local0".."local7"</code> . По умолчанию используется <code>"local7"</code> .
<code>severity=string</code>	Определяет уровень серьезности сообщений <code>syslog</code> для <code>access_log</code> , как указано в RFC 3164 . Возможные значения те же, что и для второго параметра (уровень) директивы <code>error_log</code> . По умолчанию используется <code>"info"</code> . Серьезность сообщений об ошибках определяется Angie, поэтому этот параметр игнорируется в директиве <code>error_log</code> .
<code>tag=string</code>	Устанавливает тег для сообщений <code>syslog</code> . По умолчанию используется тег <code>"angie"</code> .
<code>nohostname</code>	Отключает добавление поля <code>hostname</code> в заголовок сообщения <code>syslog</code> .

Пример конфигурации `syslog`:

```
error_log syslog:server=192.168.1.1 debug;
access_log syslog:server=[2001:db8::1]:12345,facility=local7,tag=angie,severity=info,
↪combined;
```

Примечание

Сообщения в `syslog` записываются не чаще раза в секунду, чтобы предотвратить переполнение.

Основной модуль

Модуль предоставляет основную функциональность и директивы конфигурации, необходимые для базовой работы сервера, а также решает важные задачи, такие как управление рабочими процессами, настройка событийно-ориентированных моделей и обработка входящих соединений и запросов. Он включает ключевые директивы для настройки основного процесса, ведения журналов ошибок и контроля поведения сервера на низком уровне.

Пример конфигурации

```
user www www;
worker_processes 2;

error_log /var/log/error.log info;

events {
    use kqueue; worker_connections 2048;
}
```

Директивы

accept_mutex

<i>Синтаксис</i>	accept_mutex on off;
По умолчанию	accept_mutex off;
<i>Контекст</i>	events

Когда `accept_mutex` включен, рабочие процессы будут принимать новые соединения поочередно. Иначе уведомления о новых соединениях получают все рабочие процессы, что может привести к неэффективному использованию системных ресурсов, если количество новых соединений невелико.

Примечание

Нет необходимости включать `accept_mutex` на системах, которые поддерживают флаг `EPOLLEXCLUSIVE`, или при использовании директивы `reuseport`.

accept_mutex_delay

<i>Синтаксис</i>	accept_mutex_delay время;
По умолчанию	accept_mutex_delay 500ms;
<i>Контекст</i>	events

Если `accept_mutex` включен, эта директива указывает максимальное время, в течение которого рабочий процесс ждет, чтобы продолжить принимать новые соединения, если другой рабочий процесс уже обрабатывает новые соединения.

daemon

<i>Синтаксис</i>	<code>daemon on off;</code>
По умолчанию	<code>daemon on;</code>
<i>Контекст</i>	<code>main</code>

Определяет, будет ли Angie запускаться в режиме демона. Используется в основном для разработки.

debug_connection

<i>Синтаксис</i>	<code>debug_connection адрес CIDR unix;;</code>
По умолчанию	<code>—</code>
<i>Контекст</i>	<code>events</code>

Включает отладочный лог для отдельных клиентских соединений. Для остальных соединений используется уровень лога, заданный директивой *error_log*. Указывать соединения можно по IPv4-или IPv6-адресу, сети или имени хоста. Используйте параметр `unix:`, чтобы включить отладочный лог для соединений через UNIX-сокеты.

```
events {
    debug_connection 127.0.0.1;
    debug_connection localhost;
    debug_connection 192.0.2.0/24;
    debug_connection ::1;
    debug_connection 2001:0db8::/32;
    debug_connection unix;;
    # ...
}
```

Примечание

Чтобы эта директива работала, в сборке Angie должен быть включен отладочный лог.

debug_points

<i>Синтаксис</i>	<code>debug_points abort stop;</code>
По умолчанию	<code>—</code>
<i>Контекст</i>	<code>main</code>

Эта директива используется для отладки.

В случае обнаружения внутренней ошибки, например, утечки сокетов в момент перезапуска рабочих процессов, включение `debug_points` либо создаст файл дампа памяти (`abort`), либо остановит процесс (`stop`) для анализа с помощью отладчика.

env

<i>Синтаксис</i>	<code>env переменная [=значение];</code>
По умолчанию	<code>env TZ;</code>
<i>Контекст</i>	<code>main</code>

По умолчанию Angie удаляет все переменные окружения, унаследованные от родительского процесса, кроме переменной `TZ`. Эта директива позволяет сохранить часть унаследованных переменных, поменять их значения или создать новые переменные окружения.

Эти переменные затем доступны рабочими процессами.

Обратите внимание, что управление системными библиотеками таким образом может быть не всегда эффективным, поскольку библиотеки часто проверяют переменные только во время инициализации, которая происходит до срабатывания этой директивы.

Пример:

```
env MALLOC_OPTIONS;
env PERL5LIB=/data/site/modules;
env OPENSSSL_ALLOW_PROXY_CERTS=1;
```

Примечание

Переменная окружения `ANGIE` используется внутри Angie и не должна задаваться напрямую пользователем.

error_log

<i>Синтаксис</i>	<code>error_log файл [уровень];</code>
По умолчанию	<code>error_log logs/error.log error;</code>
<i>Контекст</i>	<code>main, http, mail, stream, server, location</code>

Настраивает логирование, позволяя указывать несколько логов на одном уровне конфигурации. Если файл лога не указан явно на уровне конфигурации `main`, будет использоваться файл по умолчанию.

Первый параметр указывает файл для хранения лога. Специальное значение `stderr` задает стандартный поток ошибок. Для настройки логирования в *syslog* используйте префикс `"syslog:"`. Для логирования в циклический буфер памяти используйте префикс `"memory:"`, за которым следует размер буфера; обычно он используется для отладки.

Второй параметр задает уровень логирования одним из следующих значений: `debug`, `info`, `notice`, `warn`, `error`, `crit`, `alert` или `emerg`. Уровни перечислены в порядке возрастания серьезности. При задании уровня будут логироваться сообщения равного и более высокого уровня:

Настройка	Уровни записи
debug	debug, info, notice, warn, error, crit, alert, emerg
info	info, notice, warn, error, crit, alert, emerg
notice	notice, warn, error, crit, alert, emerg
warn	warn, error, crit, alert, emerg
error	error, crit, alert, emerg
crit	crit, alert, emerg
alert	alert, emerg
emerg	emerg

Если этот параметр не задан, по умолчанию используется уровень логирования **error**.

Примечание

Чтобы работал уровень логирования **debug**, в сборке Angie должен быть включен отладочный лог.

Каждая запись в журнале ошибок имеет следующий формат:

```
временная_метка [уровень] PID#TID: *id_запроса сообщение
```

Где:

- **временная_метка** — дата и время события;
- **уровень** — уровень логирования события;
- **PID#TID** — идентификаторы процесса и потока;
- ***id_запроса** — уникальный идентификатор запроса (если применимо);
- **сообщение** — текст сообщения об ошибке или событии.

events

<i>Синтаксис</i>	events { ... };
По умолчанию	—
<i>Контекст</i>	main

Предоставляет контекст конфигурационного файла для директив, влияющих на обработку соединений.

include

<i>Синтаксис</i>	include файл маска;
По умолчанию	—
<i>Контекст</i>	любой

Включает в конфигурацию другой файл или файлы, подходящие под заданную *маску*. Включаемые файлы должны содержать синтаксически верные директивы и блоки.

Пример:

```
include mime.types;
include vhosts/*.conf;
```

lock_file

Синтаксис	lock_file <i>файл</i> ;
По умолчанию	lock_file logs/angie.lock;
Контекст	main

Angie использует механизм блокировок для реализации *accept_mutex* и сериализации доступа к разделяемой памяти. На большинстве систем блокировки управляются с помощью атомарных операций, что делает эту директиву ненужной. Однако на некоторых системах используется альтернативный механизм *lock file*. Эта директива устанавливает префикс для имен файлов блокировок.

master_process

Синтаксис	master_process on off;
По умолчанию	master_process on;
Контекст	main

Определяет, будут ли запускаться рабочие процессы. Эта директива предназначена для разработчиков Angie.

multi_accept

Синтаксис	multi_accept on off;
По умолчанию	multi_accept off;
Контекст	events

on	Рабочий процесс будет принимать сразу все новые соединения.
off	Рабочий процесс будет принимать только одно новое соединение за раз.

Примечание

Директива игнорируется при использовании метода обработки соединений *kqueue*, так как он сам задает число новых соединений, ожидающих приема.

pcre_jit

<i>Синтаксис</i>	<code>pcre_jit on off;</code>
По умолчанию	<code>pcre_jit off;</code>
<i>Контекст</i>	<code>main</code>

Разрешает или запрещает использование JIT-компиляции (PCRE JIT) для регулярных выражений, известных на момент парсинга конфигурации.

Использование PCRE JIT заметно ускоряет обработку регулярных выражений.

Примечание

Для работы JIT необходима библиотека PCRE версии 8.20 или выше, собранная с параметром сборки `--enable-jit`. Если Angie собирается с библиотекой PCRE (`--with-pcre=`), поддержка JIT включается с помощью параметра `--with-pcre-jit`.

pid

<i>Синтаксис</i>	<code>pid файл off;</code>
По умолчанию	<code>pid logs/angie.pid;</code>
<i>Контекст</i>	<code>main</code>

Указывает *файл*, где будет храниться идентификатор главного процесса Angie. Файл создается атомарно, что обеспечивает корректность его содержимого. Настройка `off` отключает создание этого файла.

Примечание

Если значение *file* изменяется при переконфигурации, но указывает на симлинк предыдущего PID-файла, файл не будет создан заново.

ssl_engine

<i>Синтаксис</i>	<code>ssl_engine устройство;</code>
По умолчанию	<code>—</code>
<i>Контекст</i>	<code>main</code>

Задаёт название аппаратного SSL-акселератора.

ssl_object_cache_inheritable

<i>Синтаксис</i>	<code>ssl_object_cache_inheritable on off;</code>
Значение по умолчанию	<code>ssl_object_cache_inheritable on;</code>
<i>Контекст</i>	<code>main</code>

Если включено, SSL-объекты (SSL-сертификаты, секретные ключи, доверенные сертификаты СА, списки отзыва сертификатов) наследуются при перезагрузке конфигурации.

SSL-объекты, загруженные из файлов, наследуются, если время их изменения и индекс файла не изменились с момента предыдущей загрузки конфигурации. Секретные ключи, указанные как `engine:name:id`, никогда не наследуются, тогда как ключи, указанные как `data:value`, всегда наследуются.

SSL-объекты, загруженные из переменных, не могут быть унаследованы.

Пример:

```
ssl_object_cache_inheritable on;

http {
    server {
        ssl_certificate     example.com.crt;
        ssl_certificate_key example.com.key;
    }
}
```

thread_pool

<i>Синтаксис</i>	<code>thread_pool имя threads=число [max_queue=число];</code>
По умолчанию	<code>thread_pool default threads=32 max_queue=65536;</code>
<i>Контекст</i>	<code>main</code>

Задаёт *имя* и параметры пула потоков, используемого для многопоточной обработки операций чтения и отправки файлов *без блокирования* рабочих процессов.

Параметр `threads` задаёт число потоков в пуле.

Если все потоки в пуле заняты выполнением заданий, новые задания ждут в очереди. Параметр `max_queue` ограничивает число заданий, ожидающих своего выполнения в очереди. По умолчанию в очереди может находиться до 65536 заданий. Если очередь переполнена, новые задания завершаются с ошибкой.

timer_resolution

<i>Синтаксис</i>	<code>timer_resolution интервал;</code>
По умолчанию	—
<i>Контекст</i>	<code>main</code>

Уменьшает разрешение таймеров времени в рабочих процессах, за счёт чего уменьшается число системных вызовов `gettimeofday()`. По умолчанию `gettimeofday()` вызывается при каждом полу-

чении событий из ядра. При уменьшении разрешения функция вызывается только раз за указанный интервал.

Пример:

```
timer_resolution 100ms;
```

Внутренняя реализация интервала зависит от используемого метода:

- Фильтр EVFILT_TIMER, если используется *kqueue*.
- Функция `timer_create()`, если используется *eventport*.
- Функция `setitimer()`, в противном случае.

use

<i>Синтаксис</i>	<code>use метод;</code>
По умолчанию	—
<i>Контекст</i>	events

Задаёт *метод*, используемый для *обработки соединений*. Обычно нет необходимости задавать его явно, поскольку по умолчанию Angie выбирает наиболее эффективный метод.

user

<i>Синтаксис</i>	<code>user пользователь [группа];</code>
По умолчанию	<code>user <параметр сборки --user> <параметр сборки --group>;</code>
<i>Контекст</i>	main

Задаёт пользователя и группу для рабочих процессов. Если задан только пользователь, для группы также задается указанное имя пользователя.

worker_aio_requests

<i>Синтаксис</i>	<code>worker_aio_requests число;</code>
По умолчанию	<code>worker_aio_requests 32;</code>
<i>Контекст</i>	events

При использовании *aio* с методом обработки соединений *epoll* задаёт максимум операций асинхронного ввода-вывода, ожидающих обработки, для одного рабочего процесса.

worker_connections

<i>Синтаксис</i>	<code>worker_connections число;</code>
По умолчанию	<code>worker_connections 512;</code>
<i>Контекст</i>	events

Задаёт максимум соединений, которые одновременно может открыть рабочий процесс.

Обратите внимание, что это число включает все соединения, такие как соединения с проксируемыми серверами, а не только клиентские. Кроме того, фактическое количество одновременных соединений не может превышать системный лимит на открытые файлы, который можно настроить с помощью `worker_rlimit_nofile`.

worker_cpu_affinity

<i>Синтаксис</i>	<code>worker_cpu_affinity маска_CPU ...;</code> <code>worker_cpu_affinity auto [маска_CPU];</code>
По умолчанию	—
<i>Контекст</i>	main

Привязывает рабочие процессы к группам процессоров. Каждая группа процессоров задается битовой маской разрешенных процессоров. Для каждого рабочего процесса должна быть задана отдельная группа. По умолчанию рабочие процессы не привязаны к конкретным процессорам.

Пример:

```
worker_processes 4;
worker_cpu_affinity 0001 0010 0100 1000;
```

Эта конфигурация привязывает каждый рабочий процесс к отдельному процессору.

В качестве альтернативы:

```
worker_processes 2;
worker_cpu_affinity 0101 1010;
```

Это привязывает первый рабочий процесс к CPU0 и CPU2, а второй рабочий процесс к CPU1 и CPU3. Такая настройка подходит для гипертренинга.

Специальное значение `auto` позволяет автоматически привязывать рабочие процессы к доступным процессорам:

```
worker_processes auto;
worker_cpu_affinity auto;
```

С помощью необязательной маски можно ограничить процессоры, доступные для автоматической привязки:

```
worker_cpu_affinity auto 01010101;
```

Примечание

Директива доступна только на FreeBSD и Linux.

worker_priority

<i>Синтаксис</i>	<code>worker_priority число;</code>
По умолчанию	<code>worker_priority 0;</code>
<i>Контекст</i>	main

Задаёт приоритет планирования рабочих процессов подобно тому, как это делается командой `nice`: отрицательное *число* означает более высокий приоритет. Диапазон возможных значений — от -20 до 20.

Пример:

```
worker_priority -10;
```

worker_processes

<i>Синтаксис</i>	<code>worker_processes число auto;</code>
По умолчанию	<code>worker_processes 1;</code>
<i>Контекст</i>	main

Задаёт число рабочих процессов.

Оптимальное значение зависит от разных факторов, включая число ядер процессора, количество жестких дисков и характер нагрузки. Если вы не уверены, рекомендуется начать с числа доступных ядер процессора. Значение `auto` пытается автоматически определить оптимальное количество.

worker_rlimit_core

<i>Синтаксис</i>	<code>worker_rlimit_core размер;</code>
По умолчанию	—
<i>Контекст</i>	main

Меняет ограничение на наибольший размер дампа памяти (`RLIMIT_CORE`) для рабочих процессов. Используется для увеличения ограничения без перезапуска главного процесса.

worker_rlimit_nofile

<i>Синтаксис</i>	<code>worker_rlimit_nofile число;</code>
По умолчанию	—
<i>Контекст</i>	main

Меняет ограничение на максимальное число открытых файлов (`RLIMIT_NOFILE`) для рабочих процессов. Используется для увеличения ограничения без перезапуска главного процесса.

worker_shutdown_timeout

<i>Синтаксис</i>	<code>worker_shutdown_timeout</code> <i>время</i> ;
По умолчанию	—
<i>Контекст</i>	main

Задаёт таймаут в секундах для постепенного завершения рабочих процессов. По истечении указанного времени Angie попытается закрыть все открытые сейчас соединения, чтобы ускорить завершение работы.

Постепенное завершение работы инициируется отправкой сигнала QUIT главному процессу, который приказывает рабочим процессам прекратить принимать новые подключения, позволяя завершить уже существующие. Рабочие процессы продолжают обрабатывать активные запросы до их завершения, после чего корректно завершают свою работу. Если соединения остаются открытыми дольше `worker_shutdown_timeout`, Angie принудительно закроет эти соединения для завершения работы. Также клиентские постоянные соединения закрываются только в случае, если они неактивны не менее времени, заданного в *lingering_timeout*.

working_directory

<i>Синтаксис</i>	<code>working_directory</code> <i>каталог</i> ;
По умолчанию	—
<i>Контекст</i>	main

Задаёт каталог, который будет текущим для рабочего процесса. Основное применение — запись дампа памяти, поэтому рабочий процесс должен иметь права на запись в этот каталог.

HTTP-модули

<i>HTTP</i>	Основная функциональность для обработки HTTP-запросов и ответов, управления HTTP
<i>Access</i>	Контроль доступа на основе IP-адресов и диапазонов CIDR.
<i>ACME</i>	Автоматическое получение и обновление SSL-сертификатов по протоколу ACME для HTTP
<i>Addition</i>	Вставка заданного фрагмента до или после тела ответа.
<i>API</i>	RESTful HTTP-интерфейс для получения базовой информации о веб-сервере и его статике
<i>Auth Basic</i>	Базовая HTTP-аутентификация для контроля доступа по имени пользователя и паролю.
<i>Auth Request</i>	Авторизация с помощью подзапроса к внешнему HTTP-сервису.
<i>AutoIndex</i>	Автоматический листинг директорий без индексного файла.
<i>Browser</i> (устарел)	Определение браузера на основе заголовка User-Agent .
<i>Charset</i>	Настройка и преобразование кодировки ответа.
<i>DAV</i>	Управление файлами на сервере по протоколу WebDAV.
<i>Empty GIF</i>	Отдача однопиксельного прозрачного GIF.
<i>FastCGI</i>	Проксирование запроса к FastCGI-серверу.
<i>FLV</i>	Псевдо-стриминг файлов в формате Flash Video (FLV).
<i>Geo</i>	Преобразование IP-адресов в заданные значения переменных.
<i>gRPC</i>	Проксирование запроса к gRPC-серверу.
<i>GunZIP</i>	Распаковка сжатых GZip-ответов для их модификации и в случаях, когда клиент не подд
<i>GZip</i>	Сжатие ответов методом GZip для экономии трафика.
<i>GZip Static</i>	Отдача статических файлов, предварительно сжатых методом GZip.
<i>Headers</i>	Изменение полей заголовка ответа.
<i>HTTP2</i>	Обработка запросов по протоколу HTTP/2.

Таблица 1 – продолжение с предыдущей с

<i>HTTP3</i>	Обработка запросов по протоколу HTTP/3.
<i>Index</i>	Настройка индексных файлов, обслуживающих запросы с косой чертой в конце (/).
<i>Limit Conn</i>	Ограничение числа одновременных запросов (активных соединений) для защиты от пере
<i>Limit Req</i>	Ограничение частоты запросов для защиты от перегрузки и подбора паролей.
<i>Log</i>	Настройка журнала запросов для отслеживания обращений к ресурсам с целью мониторин
<i>Map</i>	Преобразование переменных на основе predetermined пар "ключ-значение".
<i>Memcached</i>	Получение ответов от Memcached-сервера.
<i>Mirror</i>	Зеркалирование запросов на другие серверы.
<i>MP4</i>	Псевдо-стриминг файлов в формате MP4.
<i>Prometheus</i>	Метрики сервера в формате, совместимом с Prometheus, для мониторинга и сбора статис
<i>Proxy</i>	Реверсивное проксирование запросов к другим HTTP-серверам.
<i>Random Index</i>	Случайный выбор индексного файла для запросов, оканчивающихся косой чертой (/).
<i>RealIP</i>	Определение адреса и порта клиента при работе за другим прокси-сервером.
<i>Referer</i>	Валидация значений заголовка Referer .
<i>Rewrite</i>	Модификация URI запроса, перенаправления, установка переменных и выбор configura
<i>SCGI</i>	Проксирование запроса к SCGI-серверу.
<i>Secure Link</i>	Создание защищенных ссылок с возможностью ограничения срока доступа.
<i>Slice</i>	Разделение запроса на множество подзапросов к отдельным фрагментам для лучшего кэ
<i>Split Clients</i>	Создание переменных для A/B-тестирования, канареечных релизов, шардинга и других с
<i>SSI</i>	Обработка команд SSI (Server Side Includes) в ответах.
<i>SSL</i>	Настройка SSL/TLS для обработки запросов по протоколу HTTPS.
<i>Stub Status</i> (устарел)	Глобальные счетчики соединений и запросов в текстовом формате.
<i>Sub</i>	Поиск и замена фрагментов в теле ответа.
<i>Upstream</i>	Настройка групп проксируемых серверов для балансировки нагрузки.
<i>Upstream Probe</i>	Настройка активных проверок работоспособности для групп проксируемых серверов.
<i>UserID</i>	Выдача и обработка cookie с уникальным идентификатором клиента для отслеживания с
<i>uWSGI</i>	Проксирование запроса к uWSGI-серверу.

HTTP-модуль

Базовый HTTP-модуль реализует основную функциональность HTTP-сервера: это определение серверных блоков, настройка локаций для маршрутизации запросов, передача статических файлов и контроль доступа, настройка перенаправлений, поддержка соединений keep-alive и управление заголовками запросов и ответов.

Остальные модули этого раздела расширяют эту функциональность, позволяя гибко настраивать и оптимизировать работу HTTP-сервера под различные сценарии и требования.

Директивы

absolute_redirect

<i>Синтаксис</i>	absolute_redirect on off;
По умолчанию	absolute_redirect on;
<i>Контекст</i>	http, server, location

Если запрещено, то перенаправления, выдаваемые Angie, будут относительными.

См. также директивы *server_name_in_redirect* и *port_in_redirect*.

aio

<i>Синтаксис</i>	<code>aio on off threads [=null];</code>
По умолчанию	<code>aio off;</code>
<i>Контекст</i>	<code>http, server, location</code>

Разрешает или запрещает использование файлового асинхронного ввода-вывода (AIO) во FreeBSD и Linux:

```
location /video/ {
    aio on;
    output_buffers 1 64k;
}
```

Во FreeBSD AIO можно использовать, начиная с FreeBSD 4.3. До FreeBSD 11.0 AIO можно либо собрать в ядре статически:

```
options VFS_AIO
```

либо загрузить динамически через загружаемый модуль ядра:

```
kldload aio
```

В Linux AIO можно использовать только начиная с версии ядра 2.6.22. Кроме того, необходимо также дополнительно включить *directio*, иначе чтение будет блокирующимся:

```
location /video/ {
    aio on;
    directio 512;
    output_buffers 1 128k;
}
```

В Linux *directio* можно использовать только для чтения блоков, выровненных на границу 512 байт (или 4K для XFS). Невыровненный конец файла будет читаться блокированно. То же относится к запросам с указанием диапазона запрашиваемых байт (byte-range requests) и к запросам FLV не с начала файла: чтение невыровненных начала и конца ответа будет блокирующимся.

При одновременном включении AIO и *sendfile* в Linux для файлов, размер которых больше либо равен указанному в директиве *directio*, будет использоваться AIO, а для файлов меньшего размера или при выключенном *directio* — *sendfile*:

```
location /video/ {
    sendfile on;
    aio on;
    directio 8m;
}
```

Кроме того, читать и *отправлять* файлы можно в многопоточном режиме, не блокируя при этом рабочий процесс:

```
location /video/ {
    sendfile on;
    aio threads;
}
```

Операции чтения или отправки файлов будут обрабатываться потоками из указанного *пула*. Если пул потоков не задан явно, используется пул с именем **default**. Имя пула может быть задано при

помощи переменных:

```
aio threads=pool$disk;
```

Использование `aio on` требует сборки с конфигурационным параметром `--with-file-aio`. Для использования `aio threads` требуется сборка с параметром `--with-threads`.

В настоящий момент многопоточность совместима только с методами *epoll*, *kqueue* и *eventport*. Отправка файлов в многопоточном режиме поддерживается только на Linux.

См. также директиву *sendfile*.

aio_write

<i>Синтаксис</i>	<code>aio_write on off;</code>
По умолчанию	<code>aio_write off;</code>
<i>Контекст</i>	<code>http, server, location</code>

При включенном *aio* разрешает его использование для записи файлов. В настоящий момент это работает только при использовании `aio threads` и ограничено записью временных файлов с данными, полученными от проксируемых серверов.

alias

<i>Синтаксис</i>	<code>alias путь;</code>
По умолчанию	—
<i>Контекст</i>	<code>location</code>

Задаёт замену для указанного `location`'а. Например, при такой конфигурации:

```
location /i/ {
    alias /data/w3/images/;
}
```

на запрос `/i/top.gif` будет отдан файл `/data/w3/images/top.gif`.

В значении параметра *путь* можно использовать переменные, кроме *\$document_root* и *\$realpath_root*.

Если `alias` используется внутри `location`'а, заданного регулярным выражением, то регулярное выражение должно содержать группы захвата, а сам `alias` — ссылки на эти группы, например:

```
location ~ ^/users/(.+\. (? : gif | jpe ? g | png ) )$ {
    alias /data/w3/images/$1;
}
```

Если `location` и последняя часть значения директивы совпадают:

```
location /images/ {
    alias /data/w3/images/;
}
```

то лучше воспользоваться директивой *root*:

```
location /images/ {
    root /data/w3;
}
```

auth_delay

<i>Синтаксис</i>	<code>auth_delay время;</code>
По умолчанию	<code>auth_delay 0s;</code>
<i>Контекст</i>	http, server, location

Задерживает обработку неавторизованных запросов с кодом ответа 401 для предотвращения атак по времени в случае ограничения доступа по *паролю* или по *результату подзапроса*.

auto_redirect

<i>Синтаксис</i>	<code>auto_redirect [on off default];</code>
По умолчанию	<code>auto_redirect default;</code>
<i>Контекст</i>	http, server, location

Контролирует поведение *перенаправления*, когда префикс `location` заканчивается косой чертой:

```
location /prefix/ {
    auto_redirect on;
}
```

Здесь запрос к `/prefix` будет перенаправлен на `/prefix/`.

Значение `on` включает перенаправление в явном виде; `off` отключает его. Если указано `default`, перенаправление включается, только если `location` обрабатывает запросы через *api*, *proxy_pass*, *fastcgi_pass*, *uwsgi_pass*, *scgi_pass*, *memcached_pass*, или *grpc_pass*.

chunked_transfer_encoding

<i>Синтаксис</i>	<code>chunked_transfer_encoding on off;</code>
По умолчанию	<code>chunked_transfer_encoding on;</code>
<i>Контекст</i>	http, server, location

Позволяет запретить формат передачи данных частями (chunked transfer encoding) в HTTP/1.1. Это может понадобиться при использовании программ, не поддерживающих chunked encoding, несмотря на требования стандарта.

client

<i>Синтаксис</i>	<code>client { ... }</code>
По умолчанию	—
<i>Контекст</i>	<code>http</code>

Создает специальный контекст `client` для обработки внутренних HTTP-запросов, которые Angie выполняет самостоятельно без участия внешних клиентов.

Контекст `client` изолирует служебный трафик различных модулей Angie от пользовательского трафика, позволяя дополнительно управлять им. Внутри этого контекста можно определять только именованные `location` (с префиксом `@`); они недоступны для внешних HTTP-запросов и могут вызываться только программно через внутренние механизмы сервера.

Контекст `client` используется для:

- отправки запросов к центру сертификации в модуле *ACME* через предопределенный `location @acme`, который можно дополнительно настроить с помощью директив модуля *Proxy*;
- проверки состояния проксируемых серверов через *upstream_probe*;
- режима *sticky learn* с `remote_action` в потоковом модуле *Upstream*.

Поддержка нескольких блоков `client` позволяет группировать общие настройки для нескольких блоков `location` внутри каждого блока, что помогает избежать дублирования конфигурации.

При этом директивы, указанные в каждом блоке `client`, наследуются только явно объявленными внутри него блоками `location`. В частности, поэтому они не влияют на конфигурацию других модулей, которые неявно используют блок `client` для исходящих запросов (например, *ACME*).

Пример использования нескольких блоков `client` с наследованием настроек:

```
client {

    proxy_set_header Host docker.example.com;
    proxy_set_header Authorization "Basic QWxhZGRpbjpvGVuIHNlc2FtZQ==";

    location @docker_events {

    }

    location @docker_containers {

    }
}

client {

    proxy_method GET;
    proxy_set_header Host backend.example.com;
    proxy_set_header X-Real-IP $remote_addr;

    location @health_check {

        proxy_pass http://upstream-server/health;
    }
}
```

Примечание

Здесь допускаются те же директивы, что и в обычных `location`, однако реально работают только обработчики контента (например, *autoindex*) и переменных (например, *map*), а также директивы, которые сами порождают запросы, например *upstream_probe*.

Директивы, работающие на других *стадиях обработки запроса* (например, *limit_req*, *auth_request*, *try_files* и т. д.), здесь не работают.

client_body_buffer_size

<i>Синтаксис</i>	<code>client_body_buffer_size размер;</code>
По умолчанию	<code>client_body_buffer_size 8k 16k;</code>
<i>Контекст</i>	<code>http, server, location</code>

Задаёт размер буфера для чтения тела запроса клиента. Если тело запроса больше заданного буфера, то все тело запроса или только его часть записывается *во временный файл*. По умолчанию размер одного буфера равен двум размерам страницы. На *x86*, других 32-битных платформах и *x86-64* это 8К. На других 64-битных платформах это обычно 16К.

client_body_in_file_only

<i>Синтаксис</i>	<code>client_body_in_file_only on clean off;</code>
По умолчанию	<code>client_body_in_file_only off;</code>
<i>Контекст</i>	<code>http, server, location</code>

Определяет, сохранять ли все тело запроса клиента в файл. Директиву можно использовать для отладки и при использовании переменной *\$request_body_file*.

on	временные файлы по окончании обработки запроса не удаляются
clean	разрешает удалять временные файлы, оставшиеся по окончании обработки запроса

client_body_in_single_buffer

<i>Синтаксис</i>	<code>client_body_in_single_buffer on off;</code>
По умолчанию	<code>client_body_in_single_buffer off;</code>
<i>Контекст</i>	<code>http, server, location</code>

Определяет, сохранять ли все тело запроса клиента в одном буфере. Директива рекомендуется при использовании переменной *\$request_body* для уменьшения требуемого числа операций копирования.

client_body_temp_path

<i>Синтаксис</i>	<code>client_body_temp_path</code> <i>путь</i> [<i>уровень1</i> [<i>уровень2</i> [<i>уровень3</i>]]];
По умолчанию	<code>client_body_temp_path</code> <code>client_body_temp</code> ;
<i>Контекст</i>	http, server, location

Задаёт каталог для хранения временных файлов с телами запросов клиентов. В каталоге может использоваться иерархия подкаталогов до трех уровней. Например, при такой конфигурации

```
client_body_temp_path /spool/angie/client_temp 1 2;
```

путь к временному файлу будет следующего вида:

```
/spool/angie/client_temp/7/45/00000123457
```

client_body_timeout

<i>Синтаксис</i>	<code>client_body_timeout</code> <i>время</i> ;
По умолчанию	<code>client_body_timeout</code> 60s;
<i>Контекст</i>	http, server, location

Задаёт таймаут при чтении тела запроса клиента. Таймаут устанавливается не на всю передачу тела запроса, а только между двумя последовательными операциями чтения. Если по истечении этого времени клиент ничего не передаст, обработка запроса прекращается с ошибкой 408 (Request Time-out).

client_header_buffer_size

<i>Синтаксис</i>	<code>client_header_buffer_size</code> <i>размер</i> ;
По умолчанию	<code>client_header_buffer_size</code> 1k;
<i>Контекст</i>	http, server

Задаёт размер буфера для чтения заголовка запроса клиента. Для большинства запросов достаточно буфера размером в 1К байт. Однако если в запросе есть длинные cookies, или же запрос пришел от WAP-клиента, то он может не поместиться в 1К. Поэтому, если строка запроса или поле заголовка запроса не помещаются полностью в этот буфер, то выделяются буферы большего размера, задаваемые директивой *large_client_header_buffers*.

Если директива указана на уровне *server*, то может использоваться значение из сервера по умолчанию. Подробнее см. в разделе *Выбор виртуального сервера*.

client_header_timeout

<i>Синтаксис</i>	<code>client_header_timeout время;</code>
По умолчанию	<code>client_header_timeout 60s;</code>
<i>Контекст</i>	http, server

Задаёт таймаут при чтении заголовка запроса клиента. Если по истечении этого времени клиент не передаст полностью заголовок, обработка запроса прекращается с ошибкой 408 (Request Time-out).

client_max_body_size

<i>Синтаксис</i>	<code>client_max_body_size размер;</code>
По умолчанию	<code>client_max_body_size 1m;</code>
<i>Контекст</i>	http, server, location

Задаёт максимально допустимый размер тела запроса клиента. Если размер больше заданного, то клиенту возвращается ошибка 413 (Request Entity Too Large). Следует иметь в виду, что браузеры не умеют корректно показывать эту ошибку.

0	отключает проверку размера тела запроса клиента

connection_pool_size

<i>Синтаксис</i>	<code>connection_pool_size размер;</code>
По умолчанию	<code>connection_pool_size 256 512;</code>
<i>Контекст</i>	http, server, location

Позволяет производить точную настройку выделения памяти под конкретные соединения. Эта директива не оказывает существенного влияния на производительность, и ее не следует использовать. По умолчанию:

256 (байт)	на 32-битных платформах
512 (байт)	на 64-битных платформах

default_type

<i>Синтаксис</i>	<code>default_type mime-тип;</code>
По умолчанию	<code>default_type text/plain;</code>
<i>Контекст</i>	http, server, location

Задаёт MIME-тип ответов по умолчанию. Соответствие расширений имен файлов MIME-типу ответов задается с помощью директивы *types*.

directio

<i>Синтаксис</i>	<code>directio размер off;</code>
По умолчанию	<code>directio off;</code>
<i>Контекст</i>	http, server, location

Разрешает использовать флаги `O_DIRECT` (FreeBSD, Linux), `F_NOCACHE` (macOS) или функцию `directio()` (Solaris) при чтении файлов, размер которых больше либо равен указанному. Директива автоматически запрещает использование *sendfile* для данного запроса. Рекомендуется использовать для больших файлов:

```
directio 4m;
```

или при использовании *aio* в Linux.

directio_alignment

<i>Синтаксис</i>	<code>directio_alignment размер;</code>
По умолчанию	<code>directio_alignment 512;</code>
<i>Контекст</i>	http, server, location

Устанавливает выравнивание для *directio*. В большинстве случаев достаточно 512-байтового выравнивания, однако при использовании XFS под Linux его нужно увеличить до 4К.

disable_symlinks

<i>Синтаксис</i>	<code>disable_symlinks off;</code> <code>disable_symlinks on if_not_owner [from=часть];</code>
По умолчанию	<code>disable_symlinks off;</code>
<i>Контекст</i>	http, server, location

Определяет, как следует поступать с символическими ссылками при открытии файлов:

<code>off</code>	Символические ссылки в пути допускаются и не проверяются. Это стандартное поведение.
<code>on</code>	Если любой компонент пути является символической ссылкой, доступ к файлу запрещается.
<code>if_not_owner</code>	Доступ к файлу запрещается, если любой компонент пути является символической ссылкой, а ссылка и объект, на который она ссылается, имеют разных владельцев.
<code>from=часть</code>	При проверке символических ссылок (параметры <code>on</code> и <code>if_not_owner</code>) обычно проверяются все компоненты пути. Можно не проверять символические ссылки в начальной части пути, указав дополнительно параметр <code>from=часть</code> . В этом случае символические ссылки проверяются лишь начиная с компонента пути, который следует за заданной начальной частью. Если значение не является начальной частью проверяемого пути, путь проверяется целиком, как если бы этот параметр не был указан вовсе. Если значение целиком совпадает с именем файла, символические ссылки не проверяются. В значении параметра можно использовать переменные.

Пример:

```
disable_symlinks on from=$document_root;
```

Эта директива доступна только на системах, в которых есть интерфейсы `openat()` и `fstatat()`. К таким системам относятся современные версии FreeBSD, Linux и Solaris.

⚠ Предупреждение

Параметры `on` и `if_not_owner` требуют дополнительных затрат на обработку.

На системах, не поддерживающих операцию открытия каталогов только для поиска, для использования этих параметров требуется, чтобы рабочие процессы имели право читать все проверяемые каталоги.

ℹ Примечание

Модули *AutoIndex*, *Random Index* и *DAV* в настоящий момент игнорируют эту директиву.

error_page

<i>Синтаксис</i>	<code>error_page код ... [=ответ] uri;</code>
По умолчанию	—
<i>Контекст</i>	http, server, location, if в location

Задаёт URI, который будет показываться для указанных ошибок. В значении *uri* можно использовать переменные.

Пример:

```
error_page 404 /404.html;
error_page 500 502 503 504 /50x.html;
```

При этом делается внутреннее перенаправление на указанный *uri*, а метод запроса клиента меняется на "GET" (для всех методов, отличных от "GET" и "HEAD").

Кроме того, можно поменять код ответа на другой, используя синтаксис вида `=ответ`, например:

```
error_page 404 =200 /empty.gif;
```

Если ошибочный ответ обрабатывается проксированным сервером или FastCGI/uwsgi/SCGI/gRPC-сервером, и этот сервер может вернуть разные коды ответов, например, 200, 302, 401 или 404, то можно выдавать возвращаемый им код:

```
error_page 404 = /404.php;
```

Если при внутреннем перенаправлении не нужно менять URI и метод, то можно передать обработку ошибки в именованный `location`:

```
location / {
    error_page 404 = @fallback;
}

location @fallback {
    proxy_pass http://backend;
}
```

Примечание

Если при обработке `uri` происходит ошибка, клиенту возвращается ответ с кодом последней случившейся ошибки.

Также существует возможность использовать перенаправления URL для обработки ошибок:

```
error_page 403      http://example.com/forbidden.html;
error_page 404 =301 http://example.com/notfound.html;
```

В этом случае по умолчанию клиенту возвращается код ответа 302. Его можно изменить только на один из кодов ответа, относящихся к перенаправлениям (301, 302, 303, 307 и 308).

etag

<i>Синтаксис</i>	<code>etag on off;</code>
По умолчанию	<code>etag on;</code>
<i>Контекст</i>	<code>http, server, location</code>

Разрешает или запрещает автоматическую генерацию поля **Etag** заголовка ответа для статических ресурсов.

http

<i>Синтаксис</i>	<code>http { ... }</code>
По умолчанию	—
<i>Контекст</i>	<code>main</code>

Предоставляет контекст конфигурационного файла, в котором указываются директивы HTTP-сервера.

if_modified_since

<i>Синтаксис</i>	<code>if_modified_since off exact before;</code>
По умолчанию	<code>if_modified_since exact;</code>
<i>Контекст</i>	<code>http, server, location</code>

Определяет, как сравнивать время модификации ответа с временем в поле If-Modified-Since заголовка запроса:

off	ответ всегда считается изменившимся
exact	точное совпадение
before	время модификации ответа меньше или равно времени, заданному в поле If-Modified-Since заголовка запроса

ignore_invalid_headers

<i>Синтаксис</i>	<code>ignore_invalid_headers on off;</code>
По умолчанию	<code>ignore_invalid_headers on;</code>
<i>Контекст</i>	<code>http, server</code>

Если включено, Angie игнорирует поля заголовка с недопустимыми именами. Допустимыми считаются имена, состоящие из английских букв, цифр, дефисов и возможно знаков подчеркивания (последнее контролируется директивой [underscores_in_headers](#)).

Если директива указана на уровне [server](#), то может использоваться значение из сервера по умолчанию.

internal

<i>Синтаксис</i>	<code>internal;</code>
По умолчанию	—
<i>Контекст</i>	<code>location</code>

Указывает, что `location` может использоваться только для внутренних запросов. Для внешних запросов будет возвращаться ошибка 404 (Not Found) в контексте данного `location`, что позволяет перенаправить такие запросы с помощью [error_page](#). Внутренними запросами являются:

- запросы, перенаправленные директивами *error_page*, *index*, *random_index* и *try_files*;
- запросы, перенаправленные с помощью поля **X-Accel-Redirect** заголовка ответа вышестоящего сервера;
- подзапросы, формируемые командой **include virtual** модуля *SSI*, директивами модуля *Addition*, а также директивами *auth_request* и *mirror*;
- запросы, измененные директивой *rewrite*.

Пример:

```
error_page 404 /404.html;

location = /404.html {

    internal;
}
```

Благодаря тому, что ошибка 404 возвращается в контексте **location** с директивой **internal**, можно перенаправлять внешние запросы в другое место. Это позволяет использовать один префикс как для внешнего, так и для внутреннего запроса, но с разной обработкой, например:

```
location /path {

    internal;
    error_page 404 =@external;

    proxy_pass https://internal;
}

location @external {

    proxy_pass https://external;
}
```

Здесь внешний запрос **GET /path** будет проксирован на **https://external/path**, а такой же внутренний запрос будет проксирован на **https://internal/path**.

i Примечание

Для предотвращения заикливания, которое может возникнуть при использовании некорректных конфигураций, количество внутренних перенаправлений ограничено десятью. По достижении этого ограничения будет возвращена ошибка 500 (Internal Server Error). В таком случае в лог-файле ошибок можно увидеть сообщение **rewrite or internal redirection cycle**.

keepalive_disable

<i>Синтаксис</i>	keepalive_disable none браузер ...;
По умолчанию	keepalive_disable msie6;
<i>Контекст</i>	http, server, location

Запрещает соединения **keep-alive** с некорректно ведущими себя браузерами. Параметры *браузер* указывают, на какие браузеры это распространяется.

none	разрешает соединения keep-alive со всеми браузерами
msie6	запрещает соединения keep-alive со старыми версиями MSIE после получения запроса POST
safari	запрещает соединения keep-alive с Safari и подобными им браузерами на macOS и подобных ей ОС

keepalive_requests

<i>Синтаксис</i>	keepalive_requests <i>число</i> ;
По умолчанию	keepalive_requests 1000;
<i>Контекст</i>	http, server, location

Задаёт максимальное число запросов, которые можно сделать по одному keep-alive соединению. После того, как сделано максимальное число запросов, соединение закрывается.

Периодическое закрытие соединений необходимо для освобождения памяти, выделенной под конкретные соединения. Поэтому использование слишком большого максимального числа запросов может приводить к чрезмерному потреблению памяти и не рекомендуется.

keepalive_time

<i>Синтаксис</i>	keepalive_time <i>время</i> ;
По умолчанию	keepalive_time 1h;
<i>Контекст</i>	http, server, location

Ограничивает максимальное время, в течение которого могут обрабатываться запросы в рамках соединения keep-alive. По достижении заданного времени соединение закрывается после обработки очередного запроса.

keepalive_timeout

<i>Синтаксис</i>	keepalive_timeout <i>таймаут</i> [<i>заголовок_таймаута</i>];
По умолчанию	keepalive_timeout 75s;
<i>Контекст</i>	http, server, location

<i>таймаут</i>	задаёт время, в течение которого keep-alive соединение с клиентом не будет закрыто со стороны сервера
0	запрещает keep-alive соединения с клиентами

Второй, *необязательный*, параметр задаёт значение в поле Keep-Alive: **timeout=время** заголовка ответа. Два параметра могут отличаться друг от друга.

Поле Keep-Alive: **timeout=время** заголовка понимают Mozilla и Konqueror. MSIE сам закрывает keep-alive соединение примерно через 60 секунд.

large_client_header_buffers

<i>Синтаксис</i>	<code>large_client_header_buffers количество размер;</code>
По умолчанию	<code>large_client_header_buffers 4 8k;</code>
<i>Контекст</i>	http, server

Задаёт максимальное число и размер буферов для чтения большого заголовка запроса клиента. Строка запроса не должна превышать размера одного буфера, иначе клиенту возвращается ошибка 414 (Request-URI Too Large). Поле заголовка запроса также не должно превышать размера одного буфера, иначе клиенту возвращается ошибка 400 (Bad Request). Буферы выделяются только по мере необходимости. По умолчанию размер одного буфера равен 8К байт. Если по окончании обработки запроса соединение переходит в состояние keep-alive, эти буферы освобождаются.

Если директива указана на уровне *server*, то может использоваться значение из сервера по умолчанию.

limit_except

<i>Синтаксис</i>	<code>limit_except метод1 [метод2...] { ... };</code>
По умолчанию	—
<i>Контекст</i>	location

Ограничивает HTTP-методы, доступные внутри location. Параметр *метод* может быть одним из GET, HEAD, POST, PUT, DELETE, MKCOL, COPY, MOVE, OPTIONS, PROPFIND, PROPPATCH, LOCK, UNLOCK или PATCH. Если разрешен метод GET, то метод HEAD также будет разрешен. Доступ к остальным методам может быть ограничен при помощи директив модулей *Access* и *Auth Basic*.

```
limit_except GET {
    allow 192.168.1.0/32;
    deny all;
}
```

Примечание

Ограничение в примере действует для всех методов, **кроме GET и HEAD**.

limit_rate

<i>Синтаксис</i>	<code>limit_rate скорость;</code>
По умолчанию	<code>limit_rate 0;</code>
<i>Контекст</i>	http, server, location, if в location

Ограничивает скорость передачи ответа клиенту. Скорость задается в байтах в секунду. Значение 0 отключает ограничение скорости. Ограничение устанавливается на запрос, поэтому, если клиент одновременно откроет два соединения, суммарная скорость будет вдвое выше заданного ограничения.

В значении параметра можно использовать переменные. Это может быть полезно в случаях, когда скорость нужно ограничивать в зависимости от какого-либо условия:

```
map $slow $rate {
    1    4k;
    2    8k;
}

limit_rate $rate;
```

Ограничение скорости можно также задать в переменной `$limit_rate`, однако использовать данный метод не рекомендуется:

```
server {

    if ($slow) {
        set $limit_rate 4k;
    }

}
```

Кроме того, ограничение скорости может быть задано в поле `X-Accel-Limit-Rate` заголовка ответа проксированного сервера. Эту возможность можно запретить с помощью директив `proxy_ignore_headers`, `fastcgi_ignore_headers`, `uwsgi_ignore_headers` и `scgi_ignore_headers`.

limit_rate_after

<i>Синтаксис</i>	<code>limit_rate_after размер;</code>
По умолчанию	<code>limit_rate_after 0;</code>
<i>Контекст</i>	<code>http, server, location, if в location</code>

Задаёт начальный объём данных, после передачи которого начинает ограничиваться скорость передачи ответа клиенту. В значении параметра можно использовать переменные.

Пример:

```
location /flv/ {
    flv;
    limit_rate_after 500k;
    limit_rate      50k;
}
```

lingering_close

<i>Синтаксис</i>	<code>lingering_close on always off;</code>
По умолчанию	<code>lingering_close on;</code>
<i>Контекст</i>	<code>http, server, location</code>

Управляет закрытием соединений с клиентами.

on	Angie будет <i>ждать</i> и <i>обрабатывать</i> дополнительные данные, поступающие от клиента, перед полным закрытием соединения, но только если эвристика указывает на то, что клиент может еще послать данные.
always	Angie всегда будет ждать и обрабатывать дополнительные данные, поступающие от клиента.
off	Angie не будет ждать поступления дополнительных данных и сразу же закроет соединение. Это поведение нарушает протокол и поэтому не должно использоваться без необходимости.

Для управления закрытием HTTP/2-соединений директива должна быть задана на уровне *server*.

lingering_time

<i>Синтаксис</i>	<code>lingering_time время;</code>
По умолчанию	<code>lingering_time 30s;</code>
<i>Контекст</i>	<code>http, server, location</code>

Если действует *lingering_close*, эта директива задает максимальное время, в течение которого Angie будет обрабатывать (читать и игнорировать) дополнительные данные, поступающие от клиента. По прошествии этого времени соединение будет закрыто, даже если будут еще данные.

lingering_timeout

<i>Синтаксис</i>	<code>lingering_timeout время;</code>
По умолчанию	<code>lingering_timeout 5s;</code>
<i>Контекст</i>	<code>http, server, location</code>

Если действует *lingering_close*, эта директива задает максимальное время ожидания поступления дополнительных данных от клиента. Если в течение этого времени данные не были получены, соединение закрывается. В противном случае данные читаются и игнорируются, и Angie снова ждет поступления данных. Цикл "ждать-читать-игнорировать" повторяется, но не дольше чем задано директивой *lingering_time*.

При постепенном завершении клиентские постоянные соединения закрываются только в случае, если они неактивны не менее времени, заданного в *lingering_timeout*.

Примечание

В nginx аналогичная директива называется `keepalive_min_timeout`.

listen

<i>Синтаксис</i>	<pre>listen адрес[:порт] [default_server] [ssl] [http2 quic] [proxy_protocol] [setfib=число] [fastopen=число] [backlog=число] [rcvbuf=размер] [sndbuf=размер] [accept_filter=фильтр] [deferred] [bind] [ipv6only=on off] [reuseport] [so_keepalive=on off][keepidle]:[keepintvl]:[keepcnt]]; listen порт [default_server] [ssl] [http2 quic] [proxy_protocol] [setfib=число] [fastopen=число] [backlog=число] [rcvbuf=размер] [sndbuf=размер] [accept_filter=фильтр] [deferred] [bind] [ipv6only=on off] [reuseport] [so_keepalive=on off][keepidle]:[keepintvl]:[keepcnt]]; listen unix:путь [default_server] [ssl] [http2 quic] [proxy_protocol] [backlog=число] [rcvbuf=размер] [sndbuf=размер] [accept_filter=фильтр] [deferred] [bind] [so_keepalive=on off][keepidle]:[keepintvl]:[keepcnt]];</pre>
По умолчанию	<code>listen *:80 *:8000;</code>
<i>Контекст</i>	<code>server</code>

Задаёт *адрес* и *порт* для прослушивающего сокета или путь к UNIX-доменному сокету, на котором сервер будет принимать запросы. *Адрес* также может быть именем хоста, например:

```
listen 127.0.0.1:8000;
listen 127.0.0.1;
listen 8000;
listen *:8000;
listen localhost:8000;
```

IPv6-адреса указываются в квадратных скобках:

```
listen [::]:8000;
listen [::1];
```

UNIX-доменные сокеты задаются с префиксом `unix::`

```
listen unix:/var/run/angie.sock;
```

Можно указать *адрес* и *порт* вместе, только *адрес* или только *порт*. Если какие-либо параметры опущены, применяются следующие правила:

- Если указан только *адрес*, используется порт 80.
- Если указан только *порт*, Angie будет слушать на всех доступных интерфейсах IPv4 (и IPv6, если включен). Стоящий первым блок `server` на этом порту будет сервером по умолчанию для запросов с несопоставленным заголовком `Host`.
- Если директива не указана вовсе, Angie использует `*:80`, если запущен с правами суперпользователя, или `*:8000` в противном случае.

<code>default_server</code>	сервер, в котором указан этот параметр, будет сервером по умолчанию для указанной пары <i>адрес:порт</i> (вместе они образуют <i>слушающий сокет</i>). Если же директив с параметром <code>default_server</code> нет, сервером по умолчанию для слушающего сокета будет первый в конфигурации сервер, обслуживающий этот сокет.
<code>ssl</code>	указывает на то, что все соединения, принимаемые на данном слушающем сокете, должны работать в режиме SSL. Это позволяет задать компактную конфигурацию для сервера, работающего сразу в двух режимах — HTTP и HTTPS.
<code>http2</code>	позволяет принимать на слушающем сокете HTTP/2-соединения. Обычно, чтобы это работало, следует также указать параметр <code>ssl</code> , однако Angie можно также настроить и на прием HTTP/2-соединений без SSL. Устарело, начиная с версии 1.2.0. Используйте взамен директиву http2 .
<code>quic</code>	позволяет принимать на этом порту QUIC-соединения. Для использования этой опции в Angie должен быть включен и настроен модуль HTTP3 . Когда <code>quic</code> включен, также можно указать <code>reuseport</code> , чтобы использовать несколько рабочих процессов.
<code>proxy_protocol</code>	указывает на то, что все соединения, принимаемые на данном слушающем сокете, должны использовать протокол PROXY.

В директиве `listen` можно также указать несколько дополнительных параметров, специфичных для связанных с сокетами системных вызовов. Следующие параметры можно задать в любой директиве `listen`, но только один раз для каждого слушающего сокета:

setfib=число	задает таблицу маршрутизации, FIB (параметр <code>SO_SETFIB</code>) для слушающего сокета. В настоящий момент это работает только на FreeBSD.
fastopen=число	включает "TCP Fast Open" для слушающего сокета и ограничивает максимальную длину очереди соединений, которые еще не завершили процесс трехстороннего рукопожатия. Не включайте "TCP Fast Open", не убедившись, что сервер может адекватно обрабатывать многократное получение одного и того же SYN-пакета с данными.
backlog=число	задает параметр <code>backlog</code> в вызове <code>listen()</code> , который ограничивает максимальный размер очереди ожидающих приема соединений. По умолчанию <code>backlog</code> устанавливается равным <code>-1</code> для FreeBSD, DragonFly BSD и macOS, и <code>511</code> для других платформ.
rcvbuf=размер	задает размер буфера приема (параметр <code>SO_RCVBUF</code>) для слушающего сокета.
sndbuf=размер	задает размер буфера передачи (параметр <code>SO_SNDBUF</code>) для слушающего сокета.
accept_filter=фильтр	задает название accept-фильтра (параметр <code>SO_ACCEPTFILTER</code>) для слушающего сокета, который включается для фильтрации входящих соединений перед передачей их в <code>accept()</code> . Работает только на FreeBSD и NetBSD 5.0+. Можно использовать два фильтра: <code>dataready</code> и <code>httpready</code> .
deferred	указывает использовать отложенный <code>accept()</code> (параметр <code>TCP_DEFER_ACCEPT</code> сокета) на Linux.
bind	указывает, что для данного слушающего сокета нужно делать <code>bind()</code> отдельно. Это нужно потому, что если описаны несколько директив <code>listen</code> с одинаковым портом, но разными адресами, и одна из директив <code>listen</code> слушает на всех адресах для данного <i>порта</i> (<code>*:порт</code>), то Angie сделает <code>bind()</code> только на <code>*:порт</code> . Необходимо заметить, что в этом случае для определения адреса, на который пришло соединение, делается системный вызов <code>getsockname()</code> . Если же используются параметры <code>setfib</code> , <code>fastopen</code> , <code>backlog</code> , <code>rcvbuf</code> , <code>sndbuf</code> , <code>accept_filter</code> , <code>deferred</code> , <code>ipv6only</code> , <code>reuseport</code> или <code>so_keepalive</code> , то для данной пары <i>адрес:порт</i> всегда делается отдельный вызов <code>bind()</code> .
ipv6only=on off	определяет (через параметр сокета <code>IPV6_V6ONLY</code>), будет ли слушающий на wildcard-адресе <code>:::</code> IPv6-сокеты принимать только IPv6-соединения, или же одновременно IPv6- и IPv4-соединения. По умолчанию параметр включен. Установить его можно только один раз на старте.
reuseport	указывает, что нужно создавать отдельный слушающий сокет для каждого рабочего процесса (через параметр сокета <code>SO_REUSEPORT</code> для Linux 3.9+ и DragonFly BSD или <code>SO_REUSEPORT_LB</code> для FreeBSD 12+), позволяя ядру распределять входящие соединения между рабочими процессами. В настоящий момент это работает только на Linux 3.9+, DragonFly BSD и FreeBSD 12+.
 Предупреждение Ненадлежащее использование параметра <code>reuseport</code> может быть небезопасно.	
multipath	включает прием соединений по протоколу Multipath TCP (MPTCP), поддерживаемому в ядре Linux с версии 5.6. Параметр несовместим с <code>quic</code> .

`so_keepalive=on | off | [keepidle]:[keepintvl]:[keepcnt]`

Конфигурирует для слушающего сокета поведение "TCP keepalive".

<code>''</code>	если параметр опущен, для сокета будут действовать настройки операционной системы
<code>on</code>	для сокета включается параметр <code>SO_KEEPALIVE</code>
<code>off</code>	для сокета параметр <code>SO_KEEPALIVE</code> выключается

Некоторые операционные системы поддерживают настройку параметров "TCP keepalive" на уровне сокета посредством параметров TCP_KEEPIDLE, TCP_KEEPINTVL и TCP_KEEPCNT. На таких системах их можно сконфигурировать с помощью параметров keepidle, keepintvl и keeppcnt. Один или два параметра могут быть опущены, в таком случае для соответствующего параметра сокета будут действовать стандартные системные настройки. Например,

```
so_keepalive=30m:10
```

установит таймаут бездействия (TCP_KEEPIDLE) в 30 минут, для интервала проб (TCP_KEEPINTVL) будет действовать стандартная системная настройка, а счетчик проб (TCP_KEEPCNT) будет равен 10.

Пример:

```
listen 127.0.0.1 default_server accept_filter=dateready backlog=1024;
```

location

<i>Синтаксис</i>	<code>location ([= ~ ~* ^~] uri @имя)+ { ... }</code>
По умолчанию	—
<i>Контекст</i>	server, location

Устанавливает конфигурацию в зависимости от того, соответствует ли URI запроса какому-либо из выражений сопоставления.

Для сопоставления используется URI запроса в нормализованном виде, после декодирования текста, заданного в виде %XX, преобразования относительных элементов пути "." и ".." в реальные и возможной замены двух и более подряд идущих косых черт на одну.

Задать location можно префиксной строкой или регулярным выражением.

Регулярные выражения задаются с модификатором:

~*	Для поиска совпадения без учета регистра символов
~	С учетом регистра

Чтобы найти location, соответствующий запросу, вначале проверяются location'ы, заданные префиксными строками (префиксные location'ы). Среди них ищется location с совпадающим префиксом максимальной длины и запоминается.

Примечание

Для операционных систем, нечувствительных к регистру символов, таких как macOS, сравнение с префиксными строками производится без учета регистра. Однако сравнение ограничено только однобайтными locale'ями.

Затем проверяются регулярные выражения в порядке их следования в конфигурационном файле. Проверка регулярных выражений прекращается после первого же совпадения, и используется соответствующая конфигурация. Если совпадение с регулярным выражением не найдено, то используется конфигурация запомненного ранее префиксного location'a.

За некоторыми исключениями, о которых говорится ниже, блоки location могут быть вложенными.

Регулярные выражения могут создавать группы захвата, которые затем можно использовать в других директивах.

Если у совпавшего префиксного `location`'а указан модификатор `~~`, то регулярные выражения не проверяются.

Кроме того, с помощью модификатора `=` можно задать точное совпадение URI и `location`. При точном совпадении поиск сразу же прекращается. Например, если запрос `/` случается часто, можно ускорить обработку таких запросов, указав `location =/`, так как поиск прекратится после первого же сравнения. Такой `location` не может иметь вложенные `location`, так как он задает полное совпадение.

Пример:

```
location =/ {
    #конфигурация А
}

location / {
    #конфигурация Б
}

location /documents/ {
    #конфигурация В
}

location ~/images/ {
    #конфигурация Г
}

location ~*\. (gif|jpg|jpeg)$ {
    #конфигурация Д
}
```

- Для запроса `/` будет выбрана конфигурация А,
- для запроса `/index.html` — конфигурация Б,
- для запроса `/documents/document.html` — конфигурация В,
- для запроса `/images/1.gif` — конфигурация Г,
- а для запроса `/documents/1.jpg` — конфигурация Д.

Примечание

Если префиксный `location` задан с косой чертой в конце и включена директива `auto_redirect`, происходит следующее: На запрос с URI без косой черты в конце, в остальном совпадающий с префиксом, будет возвращено постоянное перенаправление с кодом 301, указывающее на URI запроса с добавленной в конце косой чертой.

Если задать `location` с точным совпадением URI, перенаправление не используется:

```
location /user/ {
    proxy_pass http://user.example.com;
}

location =/user {
    proxy_pass http://login.example.com;
}
```

Префикс `@` задает *именованный location*. Такой `location` не используется при обычной обработке запросов, а предназначен только для перенаправления в него запросов. Такие `location` не могут быть вложенными и не могут содержать вложенные `location`.

Комбинированные location

Для удобства несколько `location` с одинаковой конфигурацией можно записать компактно, перечислив в одном `location` сразу несколько выражений сопоставления и задав для них единую конфигурацию. Такие `location` называются *комбинированными*.

Если, например, предположить, что в предыдущем примере конфигурации А, Г и Д совпадают, то их можно записать с помощью комбинированного `location`:

```
location =/  
    ~/images/  
    ~*\. (gif|jpg|jpeg)$ {  
    #общая конфигурация  
}
```

Именованный `location` также может быть частью комбинированного:

```
location =/  
    @named_combined {  
    #...  
}
```

Предупреждение

В комбинированных `location` между модификатором выражения сопоставления и самим выражением не может стоять пробел. Правильно: `location ~*/match(ing|es|er)$ ....`

Примечание

Сейчас комбинированные `location` не могут **непосредственно** содержать директивы `proxy_pass`, в которых задан URI, а также `api` и `alias`. При этом такие директивы можно использовать в других `location`, вложенных в комбинированный.

log_not_found

<i>Синтаксис</i>	<code>log_not_found on off;</code>
По умолчанию	<code>log_not_found on;</code>
<i>Контекст</i>	<code>http, server, location</code>

Разрешает или запрещает записывать в *error_log* ошибки о том, что файл не найден.

log_subrequest

<i>Синтаксис</i>	log_subrequest on off;
По умолчанию	log_subrequest off;
<i>Контекст</i>	http, server, location

Разрешает или запрещает записывать в *access_log* подзапросы.

max_headers

<i>Синтаксис</i>	max_headers <i>число</i> ;
Значение по умолчанию	max_headers 1000;
<i>Контекст</i>	http, server

Устанавливает максимальное количество полей заголовков запроса клиента. При превышении предела клиенту возвращается ошибка 400 (Bad Request).

Если эта директива задана на уровне *server*, может использоваться значение с сервера по умолчанию. Дополнительные сведения см. в разделе *Выбор виртуального сервера*.

max_ranges

<i>Синтаксис</i>	max_ranges <i>число</i> ;
По умолчанию	—
<i>Контекст</i>	http, server, location

Ограничивает максимальное допустимое число диапазонов в запросах с указанием диапазона запрашиваемых байт (byte-range requests). Запросы, превышающие указанное ограничение, обрабатываются как если бы они не содержали указания диапазонов. По умолчанию число диапазонов не ограничено.

0	полностью запрещает поддержку диапазонов

merge_slashes

<i>Синтаксис</i>	merge_slashes on off;
По умолчанию	merge_slashes on;
<i>Контекст</i>	http, server

Разрешает или запрещает преобразование URI путем замены двух и более подряд идущих косых черт ("/") на одну.

Необходимо иметь в виду, что это преобразование необходимо для корректной проверки префиксных строк и регулярных выражений. Если его не делать, то запрос `//scripts/one.php` не попадет в

```
location /scripts/ { }
```

и может быть обслужен как статический файл. Поэтому он преобразуется к виду `/scripts/one.php`.

Запрет преобразования может понадобиться, если в URI используются имена, закодированные методом `base64`, в котором задействован символ `"/"`. Однако из соображений безопасности лучше избегать отключения преобразования.

Если директива указана на уровне `server`, то может использоваться значение из сервера по умолчанию.

msie_padding

<i>Синтаксис</i>	<code>msie_padding on off;</code>
По умолчанию	<code>msie_padding on;</code>
<i>Контекст</i>	<code>http, server, location</code>

Разрешает или запрещает добавлять в ответы для MSIE со статусом больше 400 комментариев для увеличения размера ответа до 512 байт.

msie_refresh

<i>Синтаксис</i>	<code>msie_refresh on off;</code>
По умолчанию	<code>msie_refresh off;</code>
<i>Контекст</i>	<code>http, server, location</code>

Разрешает или запрещает выдавать для MSIE клиентов `refresh`'ы вместо перенаправлений.

open_file_cache

<i>Синтаксис</i>	<code>open_file_cache off;</code> <code>open_file_cache max=N [inactive=время];</code>
По умолчанию	<code>open_file_cache off;</code>
<i>Контекст</i>	<code>http, server, location</code>

Задаёт кэш, в котором могут храниться:

- дескрипторы открытых файлов, информация об их размерах и времени модификации;
- информация о существовании каталогов;
- информация об ошибках поиска файла — "нет файла", "нет прав на чтение" и тому подобное.

Кэширование ошибок нужно разрешить отдельно директивой `open_file_cache_errors`.

max	задает максимальное число элементов в кэше; при переполнении кэша удаляются наименее востребованные элементы (LRU)
inactive	задает время, после которого элемент кэша удаляется, если к нему не было обращений в течение этого времени. По умолчанию: 60 секунд.
off	запрещает кэш.

Пример:

```
open_file_cache          max=1000 inactive=20s;
open_file_cache_valid    30s;
open_file_cache_min_uses 2;
open_file_cache_errors   on;
```

open_file_cache_errors

<i>Синтаксис</i>	<code>open_file_cache_errors on off;</code>
По умолчанию	<code>open_file_cache_errors off;</code>
<i>Контекст</i>	http, server, location

Разрешает или запрещает кэширование ошибок поиска файлов в *open_file_cache*.

open_file_cache_min_uses

<i>Синтаксис</i>	<code>open_file_cache_min_uses число;</code>
По умолчанию	<code>open_file_cache_min_uses 1;</code>
<i>Контекст</i>	http, server, location

Задает минимальное число обращений к файлу в течение времени, заданного параметром *inactive* директивы *open_file_cache*, необходимых для того, чтобы дескриптор файла оставался открытым в кэше.

open_file_cache_valid

<i>Синтаксис</i>	<code>open_file_cache_valid время;</code>
По умолчанию	<code>open_file_cache_valid 60s;</code>
<i>Контекст</i>	http, server, location

Определяет время, через которое следует проверять актуальность информации об элементе в *open_file_cache*.

output_buffers

<i>Синтаксис</i>	<code>output_buffers <i>число</i> <i>размер</i>;</code>
По умолчанию	<code>output_buffers 2 32k;</code>
<i>Контекст</i>	<code>http, server, location</code>

Задаёт *число* и *размер* буферов, используемых при чтении ответа с диска.

port_in_redirect

<i>Синтаксис</i>	<code>port_in_redirect on off;</code>
По умолчанию	<code>port_in_redirect on;</code>
<i>Контекст</i>	<code>http, server, location</code>

Разрешает или запрещает указывать порт в *абсолютных* перенаправлениях, выдаваемых Angie.

Использование в перенаправлениях основного имени сервера управляется директивой *server_name_in_redirect*.

postpone_output

<i>Синтаксис</i>	<code>postpone_output <i>размер</i>;</code>
По умолчанию	<code>postpone_output 1460;</code>
<i>Контекст</i>	<code>http, server, location</code>

Если это возможно, то отправка данных клиенту будет отложена пока Angie не накопит по крайней мере указанное количество байт для отправки.

0	запрещает отложенную отправку данных

read_ahead

<i>Синтаксис</i>	<code>read_ahead <i>размер</i>;</code>
По умолчанию	<code>read_ahead 0;</code>
<i>Контекст</i>	<code>http, server, location</code>

Задаёт ядру размер предчтения при работе с файлами.

На Linux используется системный вызов `posix_fadvise(0, 0, 0, POSIX_FADV_SEQUENTIAL)`, поэтому параметр размер там игнорируется.

На FreeBSD используется системный вызов `fcntl(0_READAHEAD, размер)`, появившийся во FreeBSD 9.0-CURRENT.

recursive_error_pages

<i>Синтаксис</i>	<code>recursive_error_pages on off;</code>
По умолчанию	<code>recursive_error_pages off;</code>
<i>Контекст</i>	http, server, location

Разрешает или запрещает делать несколько перенаправлений через директиву `error_page`. Число таких перенаправлений *ограничено*.

request_pool_size

<i>Синтаксис</i>	<code>request_pool_size размер;</code>
По умолчанию	<code>request_pool_size 4k;</code>
<i>Контекст</i>	http, server

Позволяет производить точную настройку выделения памяти под конкретные запросы. Эта директива не оказывает существенного влияния на производительность, и ее не следует использовать.

reset_timeout_connection

<i>Синтаксис</i>	<code>reset_timeout_connection on off;</code>
По умолчанию	<code>reset_timeout_connection off;</code>
<i>Контекст</i>	http, server, location

Разрешает или запрещает сброс соединений по таймауту, а также при закрытии соединений с помощью нестандартного кода 444. Сброс делается следующим образом. Перед закрытием сокета для него задается параметр `SO_LINGER` с таймаутом 0. После этого при закрытии сокета клиенту отсылается `TCP RST`, а вся память, связанная с этим сокетом, освобождается. Это позволяет избежать длительного нахождения уже закрытого сокета в состоянии `FIN_WAIT1` с заполненными буферами.

Примечание

соединения keep-alive по истечении таймаута закрываются обычным образом.

resolver

<i>Синтаксис</i>	<code>resolver адрес ... [valid=время] [ipv4=on off] [ipv6=on off] [status_zone=зона];</code>
По умолчанию	—
<i>Контекст</i>	http, server, location, upstream

Задаёт серверы DNS, используемые для преобразования имен вышестоящих серверов в адреса, например:

```
resolver 127.0.0.53 [::1]:5353;
```

Адрес может быть указан в виде доменного имени или IP-адреса, и необязательного порта. Если порт не указан, используется порт 53. Серверы DNS опрашиваются циклически.

Примечание

Значение директивы наследуется вложенными блоками и может быть переопределено в них при необходимости. В пределах одного блока допустимо указывать директиву только один раз. Если она повторяется, действует последнее определение.

По умолчанию Angie кэширует ответы, используя значение TTL из ответа DNS. Если директива **resolver** не указана и не выполняются динамические DNS-запросы (например, при использовании фиксированных имен в *Proxy* без переменных), указание резолвера не требуется: имена будут разрешены при запуске с помощью системного резолвера. Необязательный параметр **valid** позволяет это переопределить:

valid	необязательный параметр, позволяет переопределить срок кэширования ответа

```
resolver 127.0.0.53 [::1]:5353 valid=30s;
```

По умолчанию Angie будет искать как IPv4-, так и IPv6-адреса при преобразовании имен в адреса.

ipv4=off	запрещает поиск IPv4-адресов
ipv6=off	запрещает поиск IPv6-адресов

status_zone	необязательный параметр; включает сбор метрик запросов и ответов DNS-сервера (<i>/status/resolvers/<зона></i>) в указанной зоне

Совет

Для предотвращения DNS-спуфинга рекомендуется использовать DNS-серверы в защищенной доверенной локальной сети.

resolver_timeout

<i>Синтаксис</i>	resolver_timeout <i>время</i> ;
По умолчанию	resolver_timeout 30s;
<i>Контекст</i>	http, server, location, upstream

Задаёт таймаут для преобразования имени в адрес, например:

```
resolver_timeout 5s;
```

root

<i>Синтаксис</i>	<code>root путь;</code>
По умолчанию	<code>root html;</code>
<i>Контекст</i>	http, server, location, if в location

Задаёт корневой каталог для запросов. Например, при такой конфигурации

```
location /i/ {
    root /data/w3;
}
```

в ответ на запрос `/i/top.gif` будет отдан файл `/data/w3/i/top.gif`.

В значении параметра путь можно использовать переменные, кроме `$document_root` и `$realpath_root`.

Путь к файлу формируется путем простого добавления URI к значению директивы `root`. Если же URI необходимо поменять, следует воспользоваться директивой *alias*.

satisfy

<i>Синтаксис</i>	<code>satisfy all any;</code>
По умолчанию	<code>satisfy all;</code>
<i>Контекст</i>	http, server, location

Разрешает доступ, если его разрешают все (**all**) или хотя бы один (**any**) из модулей *Access*, *Auth Basic* или *Auth Request*.

```
location / {
    satisfy any;

    allow 192.168.1.0/32;
    deny all;

    auth_basic "closed site";
    auth_basic_user_file conf/htpasswd;
}
```

send_lowat

<i>Синтаксис</i>	<code>send_lowat размер;</code>
По умолчанию	<code>send_lowat 0;</code>
<i>Контекст</i>	http, server, location

При установке этой директивы в ненулевое значение Angie будет пытаться минимизировать число операций отправки на клиентских сокетах либо при помощи флага `NOTE_LOWAT` метода *queue*, либо при помощи параметра сокета `SO_SNDLOWAT`. В обоих случаях будет использован указанный размер.

send_timeout

<i>Синтаксис</i>	<code>send_timeout время;</code>
По умолчанию	<code>send_timeout 60s;</code>
<i>Контекст</i>	http, server, location

Задаёт таймаут при передаче ответа клиенту. Таймаут устанавливается не на всю передачу ответа, а только между двумя операциями записи. Если по истечении этого времени клиент ничего не примет, соединение будет закрыто.

sendfile

<i>Синтаксис</i>	<code>sendfile on off;</code>
По умолчанию	<code>sendfile off;</code>
<i>Контекст</i>	http, server, location, if в location

Разрешает или запрещает использовать `sendfile()`.

Возможно использование *aio* для подгрузки данных для `sendfile()`:

```
location /video/ {
    sendfile      on;
    tcp_nopush    on;
    aio           on;
}
```

В такой конфигурации функция `sendfile()` вызывается с флагом `SF_NODISKIO`, в результате чего она не блокируется на диске, а сообщает об отсутствии данных в памяти. После этого Angie иницирует асинхронную подгрузку данных, читая один байт. При этом ядро FreeBSD подгружает в память первые 128K байт файла, однако при последующих чтениях файл подгружается частями только по 16K. Изменить это можно с помощью директивы *read_ahead*.

sendfile_max_chunk

<i>Синтаксис</i>	<code>sendfile_max_chunk размер;</code>
По умолчанию	<code>sendfile_max_chunk 2m;</code>
<i>Контекст</i>	http, server, location

Ограничивает объём данных, который может передан за один вызов `sendfile()`. Без этого ограничения одно быстрое соединение может целиком захватить рабочий процесс.

server

<i>Синтаксис</i>	<code>server { ... }</code>
По умолчанию	—
<i>Контекст</i>	http

Задаёт конфигурацию для виртуального сервера. Четкого разделения виртуальных серверов на IP-based (на основании IP-адреса) и name-based (на основании поля "Host" заголовка запроса) нет. Вместо этого директивами *listen* описываются все адреса и порты, на которых нужно принимать соединения для этого сервера, а в директиве *server_name* указываются все имена серверов.

Подробнее: *Как обрабатываются запросы*

server_name

<i>Синтаксис</i>	<code>server_name имя ...;</code>
По умолчанию	<code>server_name "";</code>
<i>Контекст</i>	server

Задаёт имена виртуального сервера, например:

```
server {
    server_name example.com www.example.com;
}
```

Первое имя становится основным именем сервера.

В именах серверов можно использовать звездочку ("*") для замены первой или последней части имени:

```
server {
    server_name example.com *.example.com www.example.*;
}
```

Такие имена называются именами с маской.

Два первых вышеприведенных имени можно объединить в одно:

```
server {
    server_name .example.com;
}
```

В качестве имени сервера можно также использовать регулярное выражение, указав перед ним тильду ("~"):

```
server {
    server_name ~^www\d+\.example\.com$ www.example.com;
}
```

Регулярное выражение может содержать группы захвата, которые могут затем использоваться в других директивах:

```
server {
    server_name ~^(www\.)?(.+)$;
```

```
location / {
    root /sites/$2;
}

server {
    server_name _;

    location / {
        root /sites/default;
    }
}
```

Именованные группы захвата в регулярном выражении создают переменные, которые могут затем использоваться в других директивах:

```
server {
    server_name ~^(www\.)?(?<domain>.+)$;

    location / {
        root /sites/$domain;
    }
}

server {
    server_name _;

    location / {
        root /sites/default;
    }
}
```

Примечание

Если параметр директивы задан как *\$hostname*, используется имя хоста веб-сервера.

Можно также указать пустое имя сервера (""):

```
server {
    server_name www.example.com "";
}
```

При поиске виртуального сервера по имени, которому соответствует несколько указанных вариантов (например, одновременно подходят и имя с маской, и регулярное выражение), будет выбран первый подходящий вариант в следующем порядке приоритета:

- точное имя;
- самое длинное имя с маской в начале, например `*.example.com`;
- самое длинное имя с маской в конце, например `mail.*`;
- первое подходящее регулярное выражение (в порядке следования в конфигурации), в том числе пустое имя.

Предупреждение

Чтобы использовать `server_name` с TLS, необходима терминация TLS-соединения. Эта директива сопоставляется с `Host` в HTTP-запросе, поэтому рукопожатие должно быть закончено, а соединение расшифровано.

server_name_in_redirect

<i>Синтаксис</i>	<code>server_name_in_redirect on off;</code>
По умолчанию	<code>server_name_in_redirect off;</code>
<i>Контекст</i>	http, server, location

Разрешает или запрещает использовать в *абсолютных* перенаправлениях, выдаваемых Angie, основное имя сервера, задаваемое директивой `server_name`.

<code>on</code>	используется основное имя сервера, задаваемое директивой <code>server_name</code>
<code>off</code>	используется имя, указанное в поле "Host" заголовка запроса. Если же этого поля нет, то используется IP-адрес сервера.

Использование в перенаправлениях порта управляется директивой `port_in_redirect`.

server_names_hash_bucket_size

<i>Синтаксис</i>	<code>server_names_hash_bucket_size размер;</code>
По умолчанию	<code>server_names_hash_bucket_size 32 64 128;</code>
<i>Контекст</i>	http

Задаёт размер корзины в хэш-таблицах имен серверов. Значение по умолчанию зависит от размера строки кэша процессора. Подробнее настройка хэш-таблиц обсуждается *отдельно*.

server_names_hash_max_size

<i>Синтаксис</i>	<code>server_names_hash_max_size размер;</code>
По умолчанию	<code>server_names_hash_max_size 512;</code>
<i>Контекст</i>	http

Задаёт максимальный размер хэш-таблиц имен серверов. Подробнее настройка хэш-таблиц обсуждается *отдельно*.

server_tokens

<i>Синтаксис</i>	<code>server_tokens on off build строка;</code>
По умолчанию	<code>server_tokens on;</code>
<i>Контекст</i>	http, server, location

Разрешает или запрещает указывать версию Angie на страницах ошибок и в поле заголовка ответа **Server**. Если указан параметр `build`, то вместе с версией будет также указано имя сборки, заданное соответствующим параметром скрипта.

В Angie директива может быть задана *строкой*, которая может также содержать переменные. Тогда на страницах ошибок и в поле заголовка ответа **Server** вместо имени сервера, версии и имени сборки будет указываться значение этой строки с подставленными переменными. Пустая *строка* запрещает выдачу поля **Server**.

status_zone

<i>Синтаксис</i>	<code>status_zone off зона ключ zone=зона[:число];</code>
По умолчанию	—
<i>Контекст</i>	server, location, if в location

Выделяет зону разделяемой памяти для сбора метрик `/status/http/location_zones/<зона>` и `/status/http/server_zones/<зона>`.

Несколько контекстов **server** могут совместно использовать одну и ту же зону для сбора данных; особое значение `off` выключает сбор данных во вложенных блоках **location**.

Синтаксис с одним значением *зоны* объединяет все метрики для текущего контекста в одну зону разделяемой памяти:

```
server {
    listen 80;
    server_name *.example.com;

    status_zone single;
    # ...
}
```

Альтернативный синтаксис позволяет задавать следующие параметры:

<i>ключ</i>	Строка с переменными, значение которой определяет группировку запросов в зоне. Все запросы, дающие одинаковые значения после подстановки, объединяются в одну группу. Если подстановка возвращает пустое значение, метрики не обновляются.
<i>зона</i>	Имя зоны разделяемой памяти.
<i>число</i> (необязательный)	Максимальное количество отдельных групп для сбора метрик. Если новые значения <i>ключа</i> превышают этот лимит, они объединяются в группу zone . Значение по умолчанию — 1.

В следующем примере все запросы с одинаковым значением `$host` группируются в **host_zone**. Метрики собираются отдельно для каждого уникального значения `$host` до тех пор, пока количе-

ство групп метрик не достигнет 10. После этого любые новые значения `$host` будут добавляться в группу `host_zone`:

```
server {

    listen 80;
    server_name *.example.com;

    status_zone $host zone=host_zone:10;

    location / {

        proxy_pass http://example.com;
    }
}
```

Результирующие метрики разделяются по отдельным хостам в выводе API.

subrequest_output_buffer_size

<i>Синтаксис</i>	<code>subrequest_output_buffer_size размер;</code>
По умолчанию	<code>subrequest_output_buffer_size 4k 8k;</code>
<i>Контекст</i>	<code>http, server, location</code>

Задаёт размер буфера, используемого для хранения тела ответа подзапроса. По умолчанию размер одного буфера равен размеру страницы памяти. В зависимости от платформы это или 4К, или 8К, однако его можно сделать меньше.

Примечание

Директива применима только для подзапросов, тело ответа которых сохраняется в памяти. Например, подобные подзапросы создаются при помощи [SSI](#).

tcp_nodelay

<i>Синтаксис</i>	<code>tcp_nodelay on off;</code>
По умолчанию	<code>tcp_nodelay on;</code>
<i>Контекст</i>	<code>http, server, location</code>

Разрешает или запрещает использование параметра `TCP_NODELAY`. Параметр включается при переходе соединения в состояние `keep-alive`. Также, он включается на SSL-соединениях, при небуферизованном проксировании и при *проксировании* [WebSocket](#).

tcp_nopush

<i>Синтаксис</i>	<code>tcp_nopush on off;</code>
По умолчанию	<code>tcp_nopush off;</code>
<i>Контекст</i>	<code>http, server, location</code>

Разрешает или запрещает использование параметра сокета TCP_NOPUSH во FreeBSD или TCP_CORK в Linux. Параметр включается только при использовании *sendfile*. Включение параметра позволяет:

- передавать заголовок ответа и начало файла в одном пакете в Linux и во FreeBSD 4.*;
- передавать файл полными пакетами.

try_files

<i>Синтаксис</i>	<code>try_files файл ... uri;</code> <code>try_files файл ... =код;</code>
По умолчанию	—
<i>Контекст</i>	<code>server, location</code>

Проверяет существование файлов в заданном порядке и использует для обработки запроса первый найденный файл, причем обработка делается в контексте этого же `location`'а. Путь к файлу строится из параметра *файл* в соответствии с директивами *root* и *alias*. С помощью косой черты в конце имени можно проверить существование каталога, например, `$uri/`. В случае, если ни один файл не найден, производится внутреннее перенаправление на *uri*, заданный последним параметром.

Например:

```
location /images/ {
    try_files $uri /images/default.gif;
}

location = /images/default.gif {
    expires 30s;
}
```

Последний параметр может быть URI для внутреннего перенаправления, ссылкой на именованный `location` (например, `@drupal`) или кодом ответа в форме `=код` (например, `=404`):

```
location / {
    try_files $uri $uri/index.html $uri.html =404;
}
```

Следует помнить, что чрезмерное использование директивы `try_files` увеличивает число системных вызовов, что может негативно сказаться на производительности.

Так, не следует использовать `try_files` для формирования поведения, фактически дублирующего поведение по умолчанию, например:

```
location /bad_pattern {

    # try_files $uri $uri/ =404; # не рекомендуется!
}
```

Также не следует использовать `try_files` исключительно для перенаправления при отсутствии файла. Дело в том, что директива `try_files` имеет две особенности:

- Во-первых, она проверяет существование каждого файла, что увеличивает нагрузку на систему.
- Во-вторых, любые ошибки открытия файла (например, `too many open files`, ошибки прав доступа) также считаются отсутствием файла и вызывают переход к запасному обработчику, что может привести к подмене ошибок 5xx успешными ответами и некорректному кэшированию.

Так, на практике можно встретить следующую проблемную конструкцию:

```
location / {
    try_files $uri $uri/ @drupal;  # не рекомендуется!
}
```

Ее проблема в том, что единственная цель здесь — перенаправление. Использование `try_files` приводит к перечисленным выше недостаткам, но не дает никаких преимуществ, поскольку проверка существования файлов не нужна. Правильное решение — использовать директиву `error_page`, которая лишена этих недостатков:

```
error_page 404 = @drupal;
log_not_found off;
```

Напротив, в следующем примере:

```
location ~ /\.php$ {
    try_files $uri @drupal;

    fastcgi_pass ...;

    fastcgi_param SCRIPT_FILENAME /path/to$fastcgi_script_name;

    # ...
}
```

Директива `try_files` проверяет существование PHP-файла, прежде чем передать запрос настроенному в том же блоке FastCGI-серверу; здесь использование `try_files` оправдано.

Пример использования при проксировании Mongrel:

```
location / {
    try_files /system/maintenance.html
              $uri $uri/index.html $uri.html
              @mongrel;
}

location @mongrel {
    proxy_pass http://mongrel;
}
```


Пример использования вместе с Drupal/FastCGI:

```
location / {
    error_page 404 = @drupal;
}

location ~ /\.php$ {
    try_files $uri @drupal;

    fastcgi_pass ...;

    fastcgi_param SCRIPT_FILENAME /path/to$fastcgi_script_name;
    fastcgi_param SCRIPT_NAME      $fastcgi_script_name;
    fastcgi_param QUERY_STRING     $args;

    # ... прочие fastcgi_param
}

location @drupal {
    fastcgi_pass ...;

    fastcgi_param SCRIPT_FILENAME /path/to/index.php;
    fastcgi_param SCRIPT_NAME      /index.php;
    fastcgi_param QUERY_STRING     q=$uri&$args;

    # ... прочие fastcgi_param
}
```

Пример использования вместе с Wordpress и Joomla:

```
location / {
    error_page 404 = @wordpress;
}

location ~ /\.php$ {
    try_files $uri @wordpress;

    fastcgi_pass ...;

    fastcgi_param SCRIPT_FILENAME /path/to$fastcgi_script_name;
    # ... прочие fastcgi_param
}

location @wordpress {
    fastcgi_pass ...;

    fastcgi_param SCRIPT_FILENAME /path/to/index.php;
    # ... прочие fastcgi_param
}
```

types

<i>Синтаксис</i>	<code>types { ... }</code>
По умолчанию	<code>types text/html html; image/gif gif; image/jpeg jpg;</code>
<i>Контекст</i>	http, server, location

Задаёт соответствие расширений имен файлов и MIME-типов ответов. Расширения нечувствительны к регистру символов. Одному MIME-типу может соответствовать несколько расширений, например:

```
types {
    application/octet-stream bin exe dll;
    application/octet-stream deb;
    application/octet-stream dmg;
}
```

Достаточно полная таблица соответствий входит в дистрибутив Angie и находится в файле `conf/mime.types`.

Для того чтобы для определенного location'a для всех ответов выдавался MIME-тип "application/octet-stream", можно использовать следующее:

```
location /download/ {
    types { }
    default_type application/octet-stream;
}
```

types_hash_bucket_size

<i>Синтаксис</i>	<code>types_hash_bucket_size размер;</code>
По умолчанию	<code>types_hash_bucket_size 64;</code>
<i>Контекст</i>	http, server, location

Задаёт размер корзины в хэш-таблицах типов. Подробнее настройка хэш-таблиц обсуждается [отдельно](#).

types_hash_max_size

<i>Синтаксис</i>	<code>types_hash_max_size размер;</code>
По умолчанию	<code>types_hash_max_size 1024;</code>
<i>Контекст</i>	http, server, location

Задаёт максимальный размер хэш-таблиц типов. Подробнее настройка хэш-таблиц обсуждается [отдельно](#).

underscores_in_headers

<i>Синтаксис</i>	<code>underscores_in_headers on off;</code>
По умолчанию	<code>underscores_in_headers off;</code>
<i>Контекст</i>	<code>http, server</code>

Разрешает или запрещает использование символов подчеркивания в полях заголовка запроса клиента. Если использование символов подчеркивания запрещено, поля заголовка запроса, в именах которых есть подчеркивания, помечаются как недопустимые и подпадают под действие директивы *ignore_invalid_headers*.

Если директива указана на уровне *server*, то может использоваться значение из сервера по умолчанию.

variables_hash_bucket_size

<i>Синтаксис</i>	<code>variables_hash_bucket_size размер;</code>
По умолчанию	<code>variables_hash_bucket_size 64;</code>
<i>Контекст</i>	<code>http</code>

Задаёт размер корзины в хэш-таблице переменных. Подробнее настройка хэш-таблиц обсуждается *отдельно*.

variables_hash_max_size

<i>Синтаксис</i>	<code>variables_hash_max_size размер;</code>
По умолчанию	<code>variables_hash_max_size 1024;</code>
<i>Контекст</i>	<code>http</code>

Задаёт максимальный размер хэш-таблиц переменных. Подробнее настройка хэш-таблиц обсуждается *отдельно*.

Встроенные переменные

Модуль `http_core` поддерживает встроенные переменные, имена которых совпадают с именами переменных веб-сервера Apache. Прежде всего, это переменные, представляющие из себя поля заголовка запроса клиента, такие как `$http_user_agent`, `$http_cookie` и тому подобное. Кроме того, есть и другие переменные:

`$angie_version`

версия Angie

`$arg_<имя>`

аргумент *имя* в строке запроса

`$args`

аргументы в строке запроса

`$binary_remote_addr`

адрес клиента в бинарном виде, длина значения всегда 4 байта для IPv4-адресов или 16 байт для IPv6-адресов

`$body_bytes_sent`

число байт, переданное клиенту, без учета заголовка ответа; переменная совместима с параметром "%B" модуля Apache `mod_log_config`

`$bytes_sent`

число байт, переданных клиенту

`$connection`

порядковый номер соединения

`$connection_requests`

текущее число запросов в соединении

`$connection_time`

время соединения в секундах с точностью до миллисекунд

`$content_length`

поле `Content-Length` заголовка запроса

`$content_type`

поле Content-Type заголовка запроса

`$cookie_<имя>`

cookie с указанным *именем*

`$document_root`

значение директивы *root* или *alias* для текущего запроса

`$document_uri`

то же, что и *\$uri*

`$host`

в порядке приоритета: имя хоста из строки запроса, или имя хоста из поля "Host" заголовка запроса, или имя сервера, соответствующего запросу

`$hostname`

имя хоста

`$http_<имя>`

произвольное поле заголовка запроса; последняя часть имени переменной соответствует имени поля, приведенному к нижнему регистру, с заменой символов тире на символы подчеркивания

`$https`

on если соединение работает в режиме SSL, либо пустая строка

`$is_args`

?, если в строке запроса есть аргументы, и пустая строка, если их нет

`$limit_rate`

установка этой переменной позволяет ограничивать скорость передачи ответа, см. *limit_rate*

`$msec`

текущее время в секундах с точностью до миллисекунд

`$pid`

номер (PID) рабочего процесса

`$pipe`

р если запрос был pipelined, иначе .

`$proxy_protocol_addr`

Адрес клиента, полученный из заголовка протокола PROXY.

Протокол PROXY должен быть предварительно включен при помощи установки параметра `proxy_protocol` в директиве *listen*.

`$proxy_protocol_port`

Порт клиента, полученный из заголовка протокола PROXY.

Протокол PROXY должен быть предварительно включен при помощи установки параметра `proxy_protocol` в директиве *listen*.

`$proxy_protocol_server_addr`

Адрес сервера, полученный из заголовка протокола PROXY.

Протокол PROXY должен быть предварительно включен при помощи установки параметра `proxy_protocol` в директиве *listen*.

`$proxy_protocol_server_port`

Порт сервера, полученный из заголовка протокола PROXY.

Протокол PROXY должен быть предварительно включен при помощи установки параметра `proxy_protocol` в директиве *listen*.

`$proxy_protocol_tlv_<имя>`

TLV, полученный из заголовка протокола PROXY. *Имя* может быть именем типа TLV или его числовым значением. В последнем случае значение задается в шестнадцатеричном виде и должно начинаться с 0x:

```
$proxy_protocol_tlv_alpn
$proxy_protocol_tlv_0x01
```

SSL TLV могут также быть доступны как по имени типа TLV, так и по его числовому значению, оба должны начинаться с `ssl_`:

```
$proxy_protocol_tlv_ssl_version
$proxy_protocol_tlv_ssl_0x21
```

Поддерживаются следующие имена типов TLV:

- `alpn` (0x01) - протокол более высокого уровня, используемый поверх соединения
- `authority` (0x02) - значение имени хоста, передаваемое клиентом
- `unique_id` (0x05) - уникальный идентификатор соединения
- `netns` (0x30) - имя пространства имен
- `ssl` (0x20) - структура SSL TLV в бинарном виде

Поддерживаются следующие имена типов SSL TLV:

- `ssl_version` (0x21) - версия SSL, используемая в клиентском соединении
- `ssl_cn` (0x22) - Common Name сертификата
- `ssl_cipher` (0x23) - имя используемого шифра
- `ssl_sig_alg` (0x24) - алгоритм, используемый для подписи сертификата
- `ssl_key_alg` (0x25) - алгоритм публичного ключа

Также поддерживается следующее специальное имя типа SSL TLV:

- `ssl_verify` - результат проверки клиентского сертификата: 0, если клиент предоставил сертификат и он был успешно верифицирован, либо ненулевое значение

Протокол PROXY должен быть предварительно включен при помощи установки параметра `proxy_protocol` в директиве *listen*.

`$query_string`

то же, что и *\$args*

`$realpath_root`

абсолютный путь, соответствующий значению директивы *root* или *alias* для текущего запроса, в котором все символические ссылки преобразованы в реальные пути

`$remote_addr`

адрес клиента

`$remote_port`

порт клиента

`$remote_user`

имя пользователя, использованное в Basic аутентификации

`$request`

первоначальная строка запроса целиком

`$request_body`

Тело запроса. Значение переменной *появляется* в *location*'ах, обрабатываемых директивами *proxy_pass*, *fastcgi_pass*, *uwsgi_pass* и *scgi_pass*, когда тело было прочитано в *буфер в памяти*.

`$request_body_file`

Имя временного файла, в котором хранится тело запроса. По завершении обработки файл необходимо удалить. Для того чтобы тело запроса всегда записывалось в файл, следует включить *client_body_in_file_only*. При передаче имени временного файла в проксированном запросе или в запросе к FastCGI/uwsgi/SCGI-серверу следует запретить передачу самого тела директивами *proxy_pass_request_body off*, *fastcgi_pass_request_body off*, *uwsgi_pass_request_body off* или *scgi_pass_request_body off* соответственно.

`$request_completion`

OK, если запрос завершился, либо пустая строка

`$request_filename`

путь к файлу для текущего запроса, формируемый из директив *root* или *alias* и URI запроса

`$request_id`

уникальный идентификатор запроса, сформированный из 16 случайных байт, в шестнадцатеричном виде

`$request_length`

длина запроса (включая строку запроса, заголовок и тело запроса)

`$request_method`

метод запроса, обычно GET или POST

`$request_time`

время обработки запроса в секундах с точностью до миллисекунд; время, прошедшее с момента чтения первых байт от клиента

`$request_uri`

первоначальный URI запроса целиком (с аргументами)

`$scheme`

схема запроса, "http" или "https"

`$sent_http_<имя>`

произвольное поле заголовка ответа; последняя часть имени переменной соответствует имени поля, приведенному к нижнему регистру, с заменой символов тире на символы подчеркивания

`$sent_trailer_<имя>`

произвольное поле, отправленное в конце ответа; последняя часть имени переменной соответствует имени поля, приведенному к нижнему регистру, с заменой символов тире на символы подчеркивания

`$server_addr`

Адрес сервера, принявшего запрос.

Получение значения этой переменной обычно требует одного системного вызова. Чтобы избежать системного вызова, в директивах *listen* следует указывать адреса и использовать параметр `bind`.

`$server_name`

имя сервера, принявшего запрос

`$server_port`

порт сервера, принявшего запрос

`$server_protocol`

протокол запроса, обычно "HTTP/1.0", "HTTP/1.1" или "HTTP/2.0"

`$status`

статус ответа

`$time_iso8601`

локальное время в формате по стандарту ISO 8601

`$time_local`

локальное время в Common Log Format

`$tcpinfo_rtt`, `$tcpinfo_rttvar`, `$tcpinfo_snd_cwnd`, `$tcpinfo_rcv_space`

информация о клиентском TCP-соединении; доступна на системах, поддерживающих параметр сокета TCP_INFO

`$uri`

Текущий URI запроса в *нормализованном* виде.

Значение `$uri` может изменяться в процессе обработки запроса, например, при внутренних перенаправлениях или при использовании индексных файлов.

Access

Модуль управляет доступом к ресурсам сервера на основе IP-адресов клиентов или сетей. Он позволяет разрешать или блокировать доступ для определенных IP-адресов, IP-диапазонов или UNIX-сокеты, чтобы повысить безопасность, ограничивая доступ к важным разделам веб-сайта или приложения.

Доступ также можно ограничить с помощью пароля, используя модуль *Auth Basic*, или на основе результата подзапроса, используя модуль *Auth Request*. Чтобы одновременно применять ограничения по адресу и паролю, используйте директиву *satisfy*.

Пример конфигурации

```
location / {
    deny 192.168.1.1;
    allow 192.168.1.0/24;
    allow 10.1.1.0/16;
    allow 2001:0db8::/32;
    deny all;
}
```

Правила обрабатываются последовательно до первого совпадения. В этом примере доступ разрешен только для IPv4-сетей 10.1.1.0/16 и 192.168.1.0/24, за исключением отдельного адреса 192.168.1.1, и для IPv6-сети 2001:0db8::/32. Когда правил много, предпочтительно использовать переменные из модуля *Geo*.

Директивы

allow

<i>Синтаксис</i>	<code>allow адрес CIDR unix: all;</code>
По умолчанию	—
<i>Контекст</i>	<code>http, server, location, limit_except</code>

Разрешает доступ для указанной сети или адреса. Специальное значение `all` означает все IP-адреса клиентов.

Специальное значение `unix:` разрешает доступ для любых UNIX-сокетов.

deny

<i>Синтаксис</i>	<code>deny адрес CIDR unix: all;</code>
По умолчанию	—
<i>Контекст</i>	<code>http, server, location, limit_except</code>

Запрещает доступ для указанной сети или адреса. Специальное значение `all` означает все IP-адреса клиентов.

Специальное значение `unix:` запрещает доступ для любых UNIX-сокетов.

ACME

Обеспечивает автоматическое получение сертификатов с использованием протокола ACME.

При сборке из исходного кода модуль не собирается по умолчанию; его необходимо включить с помощью параметра сборки `--with-http_acme_module`. В пакетах и образах из наших репозиториях модуль включен в сборку.

Пример конфигурации

Примеры конфигурации и инструкции по настройке см. в разделе `angie__acme_config`.

Директивы

acme

<i>Синтаксис</i>	<code>acme имя;</code>
По умолчанию	—
<i>Контекст</i>	<code>server</code>

Для всех доменов, указанных в директивах `server_name` во всех блоках `server`, которые ссылаются на *клиент ACME* с именем `имя`, будет получен единый сертификат; если изменится конфигурация `server_name`, сертификат будет обновлен для учета изменений.

При каждом запуске Angie для всех доменов, у которых отсутствует действующий сертификат, запрашиваются новые сертификаты. Возможные причины включают истечение срока действия сертификатов, отсутствие файлов или невозможность прочитать их, а также изменения в настройках сертификатов.

Примечание

Сейчас домены, заданные через регулярные выражения, не поддерживаются и будут пропускаться.

Домены со звездочкой поддерживаются только в режиме `challenge=dns` в `acme_client`.

Эта директива может быть указана несколько раз для загрузки сертификатов разных типов, например RSA и ECDSA:

```
server {
    listen 443 ssl;
    server_name example.com www.example.com;

    ssl_certificate $acme_cert_rsa;
    ssl_certificate_key $acme_cert_key_rsa;

    ssl_certificate $acme_cert_ecdsa;
    ssl_certificate_key $acme_cert_key_ecdsa;

    acme rsa;
    acme ecdsa;
}
```

`acme_client`

<i>Синтаксис</i>	<code>acme_client имя uri [enabled=on off] [key_type=тип] [key_bits=число] [email=email] [max_cert_size=число] [renew_before_expiry=время] [renew_on_load] [retry_after_error=off время] [challenge=dns http] [account_key=файл];</code>
По умолчанию	—
<i>Контекст</i>	<code>http</code>

Определяет клиент ACME с глобально уникальным *именем*. Оно должно быть допустимым для каталога, представляет собой строку с переменными и будет использоваться без учета регистра.

Совет

Задаваемое здесь имя клиента идентифицирует его в конфигурации Angie, позволяя сопоставить между собой директивы `acme_client`, *аcme* и *переменные* модуля, использующие это имя; не следует путать его с именем вашего домена или сервера.

Вторым обязательным параметром является *uri* каталога ACME. Например, URI каталога Let's Encrypt ACME указан как <https://acme-v02.api.letsencrypt.org/directory>.

i Примечание

Модуль АСМЕ добавляет в контекст *client* именованный *location @acme*, который можно использовать для настройки запросов к каталогу АСМЕ; По умолчанию в этом *location* задана директива *proxy_pass* с *uri* каталога, к которой можно добавить другие настройки из модуля *Proxy*.

Чтобы директива работала, в том же контексте должен быть настроен *resolver*.

i Примечание

Для тестирования удостоверяющие центры обычно предоставляют отдельные тестовые среды. Например, среда тестирования Let's Encrypt — <https://acme-staging-v02.api.letsencrypt.org/directory>.

enabled	Включает или отключает обновление сертификатов для клиента; это полезно, например, для временной приостановки без удаления клиента из конфигурации. По умолчанию: on .
key_type	Тип алгоритма закрытого ключа для сертификата. Допустимые значения: rsa , ecdsa . По умолчанию: ecdsa .
key_bits	Количество битов в ключе сертификата. По умолчанию: 256 для ecdsa , 2048 для rsa .
email	Необязательный адрес электронной почты для обратной связи; используется при создании учетной записи на сервере СА.
max_cert_size	Задаёт максимальный допустимый размер файла нового сертификата в байтах, чтобы зарезервировать место для нового сертификата в разделяемой памяти; чем больше число доменов, для которых запрашивается сертификат, тем больше требуется места. Если в момент запуска сертификат уже существует, но его размер превышает значение max_cert_size , значение max_cert_size динамически увеличивается до размера существующего файла сертификата. Если размер сертификата, полученного при обновлении, превышает max_cert_size , процесс обновления завершится с ошибкой. По умолчанию: 8192.
renew_before_exp	<i>Время</i> до истечения срока действия сертификата, когда должно начаться его обновление. По умолчанию: 30d.
renew_on_load	Указывает, что сертификат следует принудительно обновлять при каждой загрузке конфигурации.
retry_after_error	<i>Время</i> до повторной попытки, если получить сертификат не удалось. Если задано значение off , клиент не будет снова пытаться получить сертификат после ошибки. По умолчанию: 2h.
challenge	Задаёт тип верификации для ACME-клиента. Допустимые значения: dns , http . По умолчанию: http .
account_key	Указывает полный путь к файлу, содержащему ключ в формате PEM. Это удобно, если вы хотите использовать существующий ключ аккаунта вместо автоматической генерации, или если вам нужно использовать один ключ для нескольких ACME-клиентов. Поддерживаемые типы ключей: • RSA-ключи с длиной, кратной 8, в диапазоне от 2048 до 8192 бит. • ECDSA-ключи с длиной 256, 384 или 521 бит. При указании параметра account_key следует убедиться, что файл ключа действительно существует. Если файл отсутствует, Angie попытается создать его по указанному пути. Следует учитывать, что ключи для ACME-клиентов создаются в том порядке, в каком соответствующие клиенты упомянуты в конфигурации в директивах <i>acme_client</i> , <i>acme</i> или <i>acme_hook</i> . Поэтому, если один клиент должен использовать ключ, созданный для другого, этот другой клиент должен стоять в конфигурации раньше. Кроме того, ключи создаются только для клиентов, у которых задан параметр enabled=on .

acme_client_path

<i>Синтаксис</i>	<code>acme_client_path путь;</code>
По умолчанию	—
<i>Контекст</i>	http

Переопределяет *путь* к каталогу для хранения сертификатов и ключей, заданному при сборке с помощью параметра сборки `--http-acme-client-path`.

acme_dns_port

<i>Синтаксис</i>	<code>acme_dns_port порт ip[:порт] [ip6][:порт];</code>
Значение по умолчанию	<code>acme_dns_port 53;</code>
<i>Контекст</i>	http

Указывает порт, который модуль использует для обработки DNS-запросов от ACME-сервера по UDP. Номер порта должен быть в диапазоне от 1 до 65535.

Также поддерживается указание IP-адреса вместе с опциональным портом. Могут быть использованы как IPv4-адреса в виде `ip:порт`, так и IPv6-адреса в виде `[ip6]:порт`:

```
acme_dns_port 8053;
acme_dns_port 127.0.0.1;
acme_dns_port [::1];
```

Чтобы использовать номер порта 1024 или ниже, Angie должен работать с привилегиями *супер-пользователя*.

acme_hook

<i>Синтаксис</i>	<code>acme_hook имя [uri];</code>
По умолчанию	—
<i>Контекст</i>	location

Директива связывает сервер с указанным ACME-клиентом. Через контекст `location`, где она находится, производятся вызовы обработчиков (хуков), реализуемых каким-либо внешним сервисом.

<i>имя</i>	Указывает соответствующий ACME-клиент.
<i>uri</i>	Строка с переменными; задает строку запроса для вызова обработчиков. По умолчанию — <code>/</code> .

Например, следующая конфигурация передает значения *переменных хука* в приложение FastCGI через строку запроса:

```
acme_hook example uri=/acme_hook/$acme_hook_name?domain=$acme_hook_domain&key=$acme_
↪hook_keyauth;
fastcgi_param REQUEST_URI $request_uri;
fastcgi_pass ...;
```

Встроенные переменные

`$acme_cert_<имя>`

Содержимое последнего файла сертификата (если он есть), полученного клиентом с этим *именем*.

`$acme_cert_key_<имя>`

Содержимое файла ключа сертификата, используемого клиентом с этим *именем*.

Примечание

Файл сертификата доступен, только если клиент АСМЕ получил хотя бы один сертификат, а вот файл ключа доступен сразу после запуска.

`$acme_hook_challenge`

Тип проверки. Возможные значения: **dns**, **http**.

`$acme_hook_client`

Имя АСМЕ-клиента, инициирующего запрос.

`$acme_hook_domain`

Проверяемый домен. Если это wildcard-домен, он будет передан без префикса ***..**

`$acme_hook_keyauth`

Строка авторизации:

- При DNS-проверке используется как значение TXT-записи, имя которой формируется как `_acme-challenge. + $acme_hook_domain + ..`
- При HTTP-проверке эта строка должна использоваться в качестве содержимого ответа, запрашиваемого АСМЕ-сервером.

`$acme_hook_name`

Имя хука. Для разных типов проверки оно может иметь разные значения и смысл:

Значение	Смысл при DNS-проверке	Смысл при HTTP-проверке
add (добавление хука)	Необходимо добавить соответствующую TXT-запись в конфигурацию DNS.	Необходимо подготовить ответ на соответствующий HTTP-запрос.
remove (удаление хука)	Можно удалить TXT-запись из конфигурации DNS.	Данный HTTP-запрос более не актуален; можно удалить ранее созданный файл со строкой авторизации.

`$acme_hook_token`

Токен для проверки. При HTTP-проверке используется как имя запрашиваемого файла: `/.well-known/acme-challenge/ + $acme_hook_token`.

Addition

Фильтр, добавляющий текст до и после ответа.

При сборке из исходного кода модуль не собирается по умолчанию; его необходимо включить с помощью параметра сборки `--with-http_addition_module`. В пакетах и образах из наших репозиториях модуль включен в сборку.

Пример конфигурации

```
location / {
    add_before_body /before_action;
    add_after_body  /after_action;
}
```

Директивы

add_before_body

<i>Синтаксис</i>	<code>add_before_body uri;</code>
По умолчанию	—
<i>Контекст</i>	<code>http, server, location</code>

Добавляет перед телом ответа текст, выдаваемый в результате работы заданного подзапроса. Пустая строка ("") в качестве параметра отменяет добавление, унаследованное с предыдущего уровня конфигурации.

add_after_body

<i>Синтаксис</i>	<code>add_after_body uri;</code>
По умолчанию	—
<i>Контекст</i>	<code>http, server, location</code>

Добавляет после тела ответа текст, выдаваемый в результате работы заданного подзапроса. Пустая строка ("") в качестве параметра отменяет добавление, унаследованное с предыдущего уровня конфигурации.

addition_types

<i>Синтаксис</i>	<code>addition_types mime-mun ...;</code>
По умолчанию	<code>addition_types text/html;</code>
<i>Контекст</i>	<code>http, server, location</code>

Разрешает добавлять текст в ответах с указанными MIME-типами в дополнение к `text/html`. Специальное значение `*` соответствует любому MIME-типу.

API

Модуль API реализует HTTP RESTful интерфейс для получения базовой информации о веб-сервере в формате JSON, а также *статистики* по клиентским соединениям, зонам разделяемой памяти, DNS-запросам, HTTP-запросам, кэшу HTTP-ответов, сессиям модуля *stream* и зонам модулей *limit_conn http*, *limit_conn stream*, *limit_req* и *http upstream*.

Интерфейс принимает HTTP-методы GET и HEAD; запрос с другим методом вызовет ошибку:

```
{
  "error": "MethodNotAllowed",
  "description": "The POST method is not allowed for the requested API element \"/\
→ ". "
}
```

В Angie PRO в этом интерфейсе есть раздел *динамической конфигурации*, позволяющий менять настройки без перезагрузки конфигурации или перезапуска; сейчас доступна конфигурация отдельных серверов в составе *upstream*.

Директивы

api

<i>Синтаксис</i>	<code>api путь;</code>
По умолчанию	—
<i>Контекст</i>	<code>location</code>

Включает HTTP RESTful интерфейс в `location`.

Параметр *путь* является обязательным. Подобно директиве *alias*, задает путь для замены указанного в `location`, но по дереву API, а не файловой системы.

Если указан в префиксном `location`:

```
location /stats/ {
    api /status/http/server_zones/;
}
```

часть URI запроса, совпадающая с префиксом `/stats/`, будет заменена на путь, указанный в параметре *путь*: `/status/http/server_zones/`. К примеру, по запросу `/stats/foo/` будет доступен элемент API `/status/http/server_zones/foo/`.

Допускается использование переменных: `api /status/$module/server_zones/$name/` и использование внутри regex `location`:

```
location ~/api/([~/]+)/(.*)$ {
    api /status/http/$1_zones/$2;
}
```

Здесь параметр *путь* определяет полный путь к элементу API; так, из запроса к `/api/location/data/` будут выделены переменные:

```
$1 = "location"
$2 = "data/"
```

И конечный запрос будет иметь вид `/status/http/location_zones/data/`.

Примечание

В Angie PRO можно разделить *API динамической конфигурации* и неизменяемый *API статуса*, отражающий текущее состояние:

```
location /config/ {
    api /config;
}

location /status/ {
    api /status;
}
```

Также параметр *путь* позволяет управлять доступом к API:

```
location /status/ {
    api /status;

    allow 127.0.0.1;
    deny all;
}
```

Или же:

```
location /blog/requests/ {
    api /status/http/server_zones/blog/requests;

    auth_basic "blog";
    auth_basic_user_file conf/htpasswd;
}
```

Примечание

Если `api` стоит в `location` с косой чертой в конце префикса (например, `location /name/`), и при этом в директиве `auto_redirect` указано `default`, запросы без косой черты в конце будут перенаправляться (`/name -> /name/`).

api_config_files

<i>Синтаксис</i>	api_config_files on off;
По умолчанию	off
<i>Контекст</i>	location

Включает или отключает добавление объекта **config_files**, перечисляющего содержимое всех файлов конфигурации Angie, загруженных сейчас экземпляром сервера, в состав раздела API */status/angie/*. Например, при такой конфигурации:

```
location /status/ {
    api /status/;
    api_config_files on;
}
```

Запрос к */status/angie/* возвращает приблизительно следующее:

```
{
  "version": "|version|",
  "address": "192.168.16.5",
  "generation": 1,
  "load_time": "|sampledateshort|T12:58:39.789Z",
  "config_files": {
    "/etc/angie/angie.conf": "...",
    "/etc/angie/mime.types": "..."
  }
}
```

По умолчанию вывод отключен, так как файлы конфигурации могут содержать особо чувствительные, конфиденциальные сведения.

Метрики

Angie публикует статистику использования в разделе API */status/*; открыть доступ к ней можно, задав соответствующий **location**. Полный доступ:

```
location /status/ {
    api /status/;
}
```

Пример частичного доступа, уже приводившийся выше:

```
location /stats/ {
    api /status/http/server_zones/;
}
```

Пример конфигурации

С конфигурацией, включающей location /status/, зоны resolver, http в upstream, http server, location, cache, limit_conn в http и limit_req:

```
http {

    resolver 127.0.0.53 status_zone=resolver_zone;
    proxy_cache_path /var/cache/angie/cache keys_zone=cache_zone:2m;
    limit_conn_zone $binary_remote_addr zone=limit_conn_zone:10m;
    limit_req_zone $binary_remote_addr zone=limit_req_zone:10m rate=1r/s;

    upstream upstream {
        zone upstream 256k;
        server backend.example.com service=_example._tcp resolve max_conns=5;
        keepalive 4;
    }

    server {
        server_name www.example.com;
        listen 443 ssl;

        status_zone http_server_zone;
        proxy_cache cache_zone;
        proxy_cache_valid 200 10m;

        access_log /var/log/access.log main;

        location / {
            root /usr/share/angie/html;
            status_zone location_zone;
            limit_conn limit_conn_zone 1;
            limit_req zone=limit_req_zone burst=5;
        }
        location /status/ {
            api /status/;

            allow 127.0.0.1;
            deny all;
        }
    }
}
```

В ответ на запрос `curl https://www.example.com/status/` Angie возвращает:

дерево JSON

```
{
  "angie": {
    "version": "|version|",
    "address": "192.168.16.5",
    "generation": 1,
    "load_time": "|sampledateshort|T12:58:39.789Z"
  },
  "connections": {
```

```

    "accepted":2257,
    "dropped":0,
    "active":3,
    "idle":1
  },
  "slabs": {
    "cache_zone": {
      "pages": {
        "used":2,
        "free":506
      },
      "slots": {
        "64": {
          "used":1,
          "free":63,
          "reqs":1,
          "fails":0
        },
        "512": {
          "used":1,
          "free":7,
          "reqs":1,
          "fails":0
        }
      }
    },
    "limit_conn_zone": {
      "pages": {
        "used":2,
        "free":2542
      },
      "slots": {
        "64": {
          "used":1,
          "free":63,
          "reqs":74,
          "fails":0
        },
        "128": {
          "used":1,
          "free":31,
          "reqs":1,
          "fails":0
        }
      }
    },
    "limit_req_zone": {
      "pages": {
        "used":2,
        "free":2542
      }
    }
  }
}

```

```

    },
    "slots": {
      "64": {
        "used":1,
        "free":63,
        "reqs":1,
        "fails":0
      },
      "128": {
        "used":2,
        "free":30,
        "reqs":3,
        "fails":0
      }
    }
  },
  "http": {
    "server_zones": {
      "http_server_zone": {
        "ssl": {
          "handshaked":4174,
          "reuses":0,
          "timedout":0,
          "failed":0
        },
        "requests": {
          "total":4327,
          "processing":0,
          "discarded":8
        },
        "responses": {
          "200":4305,
          "302":12,
          "404":4
        },
        "data": {
          "received":733955,
          "sent":59207757
        }
      }
    },
    "location_zones": {
      "location_zone": {
        "requests": {
          "total":4158,
          "discarded":0
        },
        "responses": {

```

```

        "200":4157,
        "304":1
    },

    "data": {
        "received":538200,
        "sent":177606236
    }
},
"cache": {
    "cache_zone": {
        "size":0,
        "cold":false,
        "hit": {
            "responses":0,
            "bytes":0
        },

        "stale": {
            "responses":0,
            "bytes":0
        },

        "updating": {
            "responses":0,
            "bytes":0
        },

        "revalidated": {
            "responses":0,
            "bytes":0
        },

        "miss": {
            "responses":0,
            "bytes":0,
            "responses_written":0,
            "bytes_written":0
        },

        "expired": {
            "responses":0,
            "bytes":0,
            "responses_written":0,
            "bytes_written":0
        },

        "bypass": {
            "responses":0,
            "bytes":0,
            "responses_written":0,
            "bytes_written":0
        }
    }
},

```



```

"limit_conns": {
  "limit_conn_zone": {
    "passed":73,
    "skipped":0,
    "rejected":0,
    "exhausted":0
  }
},

"limit_reqs": {
  "limit_req_zone": {
    "passed":54816,
    "skipped":0,
    "delayed":65,
    "rejected":26,
    "exhausted":0
  }
},

"upstreams": {
  "upstream": {
    "peers": {
      "192.168.16.4:80": {
        "server":"backend.example.com",
        "service":"_example._tcp",
        "backup":false,
        "weight":5,
        "state":"up",
        "selected": {
          "current":2,
          "total":232
        },
        "max_conns":5,
        "responses": {
          "200":222,
          "302":12
        },
        "data": {
          "sent":543866,
          "received":27349934
        },
        "health": {
          "fails":0,
          "unavailable":0,
          "downtime":0
        },
        "sid":"<server_id>"
      }
    },
    "keepalive":2
  }
}

```

```

    },
    "resolvers": {
      "resolver_zone": {
        "queries": {
          "name": 442,
          "srv": 2,
          "addr": 0
        },
        "responses": {
          "success": 440,
          "timedout": 1,
          "format_error": 0,
          "server_failure": 1,
          "not_found": 1,
          "unimplemented": 0,
          "refused": 1,
          "other": 0
        }
      }
    }
  }
}

```

Набор метрик можно запросить по отдельной ветви JSON, построив соответствующий запрос. Например:

```

$ curl https://www.example.com/status/angie
$ curl https://www.example.com/status/connections
$ curl https://www.example.com/status/slabs
$ curl https://www.example.com/status/slabs/<зона>/slots
$ curl https://www.example.com/status/slabs/<зона>/slots/64
$ curl https://www.example.com/status/http/
$ curl https://www.example.com/status/http/server_zones
$ curl https://www.example.com/status/http/server_zones/<http_server_zone>
$ curl https://www.example.com/status/http/server_zones/<http_server_zone>/ssl

```

Примечание

По умолчанию модуль использует для дат строки в формате ISO 8601; чтобы вместо этого использовать целочисленный формат эпохи UNIX, добавьте параметр `date=epoch` к строке запроса:

```

$ curl https://www.example.com/status/angie/load_time

"2024-04-01T00:59:59+01:00"

$ curl https://www.example.com/status/angie/load_time?date=epoch

1711929599

```

Состояние сервера

/status/angie

```
{
  "version": "|version|",
  "build_time": "|sampledateshort|T16:05:43.805Z",
  "address": "192.168.16.5",
  "generation": 1,
  "load_time": "|sampledateshort|T16:15:43.805Z"
  "config_files": {
    "/etc/angie/angie.conf": "...",
    "/etc/angie/mime.types": "..."
  }
}
```

version	Строка; версия запущенного сервера Angie
build	Строка; сборка, если указана при компиляции
build_time	Строка; время сборки исполняемого файла Angie в формате <i>даты</i>
address	Строка; адрес сервера, принявшего запрос к API
generation	Число; версия (поколение) конфигурации, отсчитываемая с последнего запуска Angie
load_time	Строка; время последней перезагрузки конфигурации в формате <i>даты</i> ; строковые значения даются с миллисекундным разрешением
config_files	Объект; его члены — абсолютные имена всех файлов конфигурации Angie, загруженных сейчас экземпляром сервера, а их значения — строковые представления содержимого файлов, например:

```
{
  "/etc/angie/angie.conf": "server {\n  listen 80;\n  # ... \n\n}\n\n"
```

Предупреждение

Объект `config_files` есть в `/status/angie/`, только если включена директива `api_config_files`.

Соединения

/status/connections

```
{
  "accepted": 2257,
  "dropped": 0,
  "active": 3,
  "idle": 1
}
```

accepted	Число; суммарное количество принятых клиентских соединений
dropped	Число; суммарное количество сброшенных клиентских соединений
active	Число; текущее количество активных клиентских соединений
idle	Число; текущее количество бездействующих клиентских соединений

Зоны разделяемой памяти с распределением slab

/status/slabs/<зона>

Статистика для зон разделяемой памяти с распределением slab, таких как *limit_conn*, *limit_req* и *HTTP cache*:

```
limit_conn_zone $binary_remote_addr zone=limit_conn_zone:10m;
limit_req_zone $binary_remote_addr zone=limit_req_zone:10m rate=1r/s;
proxy_cache cache_zone;
proxy_cache_valid 200 10m;
```

В указанной таким образом зоне разделяемой памяти будет собираться следующая статистика:

pages	Объект; статистика по страницам памяти
used	Число; текущее количество используемых страниц памяти
free	Число; текущее количество свободных страниц памяти
slots	Объект; статистика по слотам памяти, по каждому из размеров. slots содержит данные по размеру слота памяти (8, 16, 32, и т.д., вплоть до половины размера страницы памяти в байтах)
used	Число; текущее количество используемых слотов памяти заданного размера
free	Число; текущее количество свободных слотов памяти заданного размера
reqs	Число; суммарное количество попыток выделения памяти указанного размера
fails	Число; количество неудавшихся попыток выделения памяти указанного размера

Пример:

```
{
  "pages": {
    "used": 2,
    "free": 506
  },
  "slots": {
    "64": {
      "used": 1,
      "free": 63,
      "reqs": 1,
      "fails": 0
    }
  }
}
```

DNS-запросы к резолверу

/status/resolvers/<зона>

Для сбора статистики в директиве *resolver* нужно задать параметр *status_zone* (*HTTP* или *Stream*):

```
resolver 127.0.0.53 status_zone=resolver_zone;
```

В указанной таким образом зоне разделяемой памяти будет собираться следующая статистика:

queries	Объект; статистика запросов
name	Число; количество запросов на преобразование имен в адреса (A- и AAAA-запросы)
srv	Число; количество запросов на преобразование сервисов в адреса (SRV запросы)
addr	Число; количество запросов на преобразование адресов в имена (PTR-запросы)
responses	Объект; статистика ответов
success	Число; количество успешных ответов
timedout	Число; количество запросов, не дождавшихся ответа
format_error	Число; количество ответов с кодом 1 (Format Error)
server_failure	Число; количество ответов с кодом 2 (Server Failure)
not_found	Число; количество ответов с кодом 3 (Name Error)
unimplemented	Число; количество ответов с кодом 4 (Not Implemented)
refused	Число; количество ответов с кодом 5 (Refused)
other	Число; количество запросов, завершенных с другим ненулевым кодом
sent	Объект; статистика отправленных DNS-запросов
a	Число; количество запросов типа A
aaaa	Число; количество запросов типа AAAA
ptr	Число; количество запросов типа PTR
srv	Число; количество запросов типа SRV

Коды ответов описаны в [RFC 1035](#), часть 4.1.1.

Различные типы DNS-записей описаны в [RFC 1035](#), [RFC 2782](#) и [RFC 3596](#).

Пример:

```
{
  "queries": {
    "name": 442,
    "srv": 2,
    "addr": 0
  },

  "responses": {
    "success": 440,
    "timedout": 1,
    "format_error": 0,
    "server_failure": 1,
    "not_found": 1,
    "unimplemented": 0,
    "refused": 1,
  },

  "sent": {
    "a": 185,
```

```
"aaaa": 245,
"srv": 2,
"ptr": 12
}
}
```

HTTP server и location

/status/http/server_zones/<зона>

Для сбора статистики в контексте *server* нужно задать директиву *status_zone*:

```
server {
    ...
    status_zone server_zone;
}
```

Для группировки метрик по пользовательскому значению используйте альтернативный синтаксис. В этом примере метрики агрегируются по *\$host*, и каждая группа выводится как отдельная зона:

```
status_zone $host zone=server_zone:5;
```

В указанной таким образом зоне разделяемой памяти будет собираться следующая статистика:

ssl	Объект; SSL-метрики. Присутствует, если в server есть listen ssl ;
handshaked	Число; суммарное количество успешных SSL-рукопожатий
reuses	Число; суммарное количество повторных использований SSL-сессий во время SSL-рукопожатий
timedout	Число; суммарное количество SSL-рукопожатий с истекшим таймаутом
failed	Число; суммарное количество неуспешных SSL-рукопожатий
requests	Объект; метрики запросов
total	Число; суммарное количество клиентских запросов
processing	Число; текущее количество обслуживаемых клиентских запросов
discarded	Число; суммарное количество запросов завершенных без отправки ответа
responses	Объект; метрики ответов
<code>	Число; ненулевое количество ответов со статусом <code> (100-599)
xxx	Число; ненулевое количество ответов с другим кодом статуса
data	Объект; метрики данных
received	Число; суммарное количество байт, полученное от клиентов
sent	Число; суммарное количество байт, отправленное клиентам

Пример:

```
{
  "ssl":{
    "handshaked":4174,
    "reuses":0,
    "timedout":0,
    "failed":0
  },
  "requests":{
    "total":4327,
    "processing":0,
```

```

    "discarded":0
  },

  "responses":{
    "200":4305,
    "302":6,
    "304":12,
    "404":4
  },

  "data":{
    "received":733955,
    "sent":59207757
  }
}

```

/status/http/location_zones/<зона>

Для сбора статистики в контексте *location* или *if в location* нужно задать директиву *status_zone*:

```

location / {
    root /usr/share/angie/html;
    status_zone location_zone;

    if ($request_uri ~* "~/condition") {
        # ...
        status_zone if_location_zone;
    }
}

```

Для группировки метрик по пользовательскому значению используйте альтернативный синтаксис. В этом примере метрики агрегируются по *\$host*, и каждая группа выводится как отдельная зона:

```
status_zone $host zone=server_zone:5;
```

В указанной таким образом зоне разделяемой памяти будет собираться следующая статистика:

requests	Объект; метрики запросов
total	Число; суммарное количество клиентских запросов
discarded	Число; суммарное количество запросов завершенных без отправки ответа
responses	Объект; метрики ответов
<code>	Число; ненулевое количество ответов со статусом <code> (100-599)
xxx	Число; ненулевое количество ответов с другим кодом статуса
data	Объект; метрики данных
received	Число; суммарное количество байт, полученное от клиентов
sent	Число; суммарное количество байт, отправленное клиентам

Пример:

```

{
  "requests": {
    "total": 4158,
    "discarded": 0
  },

```

```
"responses": {
  "200": 4157,
  "304": 1
},

"data": {
  "received": 538200,
  "sent": 177606236
}
}
```

Stream server

/status/stream/server_zones/<зона>

Для сбора статистики в контексте *server* нужно задать директиву *status_zone*:

```
server {
  ...
  status_zone server_zone;
}
```

Для группировки метрик по пользовательскому значению используйте альтернативный синтаксис. В этом примере метрики агрегируются по *\$host*, и каждая группа выводится как отдельная зона:

```
status_zone $host zone=server_zone:5;
```

В указанной таким образом зоне разделяемой памяти будет собираться следующая статистика:

ssl	Объект; SSL-метрики. Присутствует, если в server есть listen ssl ;
handshaked	Число; суммарное количество успешных SSL-рукопожатий
reuses	Число; суммарное количество повторных использований SSL-сессий во время SSL-рукопожатий
timedout	Число; суммарное количество SSL-рукопожатий с истекшим таймаутом
failed	Число; суммарное количество неуспешных SSL-рукопожатий
connections	Объект; метрики соединений
total	Число; суммарное количество клиентских соединений
processing	Число; текущее количество обслуживаемых клиентских соединений
discarded	Число; суммарное количество клиентских соединений, завершенных без создания сессии
passed	Число; суммарное количество клиентских соединений, переданных на другой прослушивающий порт директивами pass
sessions	Объект; метрики сессий
success	Число; количество сессий, завершенных с кодом 200, что означает успешное завершение
invalid	Число; количество сессий, завершенных с кодом 400, случается, когда сервер не может прочитать данные от клиента, например, заголовок PROXY protocol
forbidden	Число; количество сессий, завершенных с кодом 403, когда доступ запрещен, например, ограничен для определенного адреса клиента
internal_error	Число; количество сессий, завершенных с кодом 500, внутренняя ошибка сервера
bad_gateway	Число; количество сессий, завершенных с кодом 502, Bad Gateway, если, например, сервер в upstream недоступен или не может быть выбран
service_unavailable	Число; количество сессий, завершенных с кодом 503, Service Unavailable, если, например, доступ ограничен числом входящих соединений
data	Объект; метрики данных
received	Число; суммарное количество байт, полученное от клиентов
sent	Число; суммарное количество байт, отправленное клиентам

Пример:

```
{
  "ssl": {
    "handshaked": 24,
    "reuses": 0,
    "timedout": 0,
    "failed": 0
  },

  "connections": {
    "total": 24,
    "processing": 1,
    "discarded": 0,
    "passed": 2
  },

  "sessions": {
    "success": 24,
    "invalid": 0,
    "forbidden": 0,
    "internal_error": 0,
    "bad_gateway": 0,
    "service_unavailable": 0
  },
}
```

```
"data": {
  "received": 2762947,
  "sent": 53495723
}
```

HTTP caches

```
proxy_cache cache_zone;
proxy_cache_valid 200 10m;
```

/status/http/caches/<cache>

Для каждой зоны, сконфигурированной в *proxy_cache*, хранятся следующие данные:

```
{
  "name_zone": {
    "size": 0,
    "cold": false,
    "hit": {
      "responses": 0,
      "bytes": 0
    },
    "stale": {
      "responses": 0,
      "bytes": 0
    },
    "updating": {
      "responses": 0,
      "bytes": 0
    },
    "revalidated": {
      "responses": 0,
      "bytes": 0
    },
    "miss": {
      "responses": 0,
      "bytes": 0,
      "responses_written": 0,
      "bytes_written": 0
    },
    "expired": {
      "responses": 0,
      "bytes": 0,
      "responses_written": 0,
      "bytes_written": 0
    },
    "bypass": {
```

```

    "responses": 0,
    "bytes": 0,
    "responses_written": 0,
    "bytes_written": 0
  }
}
}

```

size	Число; текущий размер кэша
max_size	Число; ограничение на максимальный размер кэша, если задано в конфигурации
cold	Логическое значение; true , пока <i>загрузчик кэша</i> подгружает данные с диска
hit	Объект; метрики возвращенных из кэша ответов (<i>proxy_cache_valid</i>)
responses	Число; суммарное количество ответов, прочитанных из кэша
bytes	Число; суммарное количество байт, прочитанных из кэша
stale	Объект; метрики просроченных ответов, возвращенных из кэша (<i>proxy_cache_use_stale</i>)
responses	Число; суммарное количество ответов, прочитанных из кэша
bytes	Число; суммарное количество байт, прочитанных из кэша
updating	Объект; метрики просроченных ответов, возвращенных из кэша, пока данные в кэше обновляются (<i>proxy_cache_use_stale updating</i>)
responses	Число; суммарное количество ответов, прочитанных из кэша
bytes	Число; суммарное количество байт, прочитанных из кэша
revalidated	Объект; метрики просроченных и ревалидированных ответов, возвращенных из кэша (<i>proxy_cache_revalidate</i>)
responses	Число; суммарное количество ответов, прочитанных из кэша
bytes	Число; суммарное количество байт, прочитанных из кэша
miss	Объект; метрики ответов, не найденных в кэше
responses	Число; суммарное количество соответствующих ответов
bytes	Число; суммарное количество байт, прочитанных с проксируемого сервера
responses_written	Число; суммарное количество ответов, записанных в кэш
bytes_written	Число; суммарное количество байт, записанных в кэш
expired	Объект; количество ответов, возвращенных не из кэша, т.к. просрочены
responses	Число; суммарное количество соответствующих ответов
bytes	Число; суммарное количество байт, прочитанных с проксируемого сервера
responses_written	Число; суммарное количество ответов, записанных в кэш
bytes_written	Число; суммарное количество байт, записанных в кэш
bypass	Объект; статистика ответов, возвращенных в обход кэша (<i>proxy_cache_bypass</i>)
responses	Число; суммарное количество соответствующих ответов
bytes	Число; суммарное количество байт, прочитанных с проксируемого сервера
responses_written	Число; суммарное количество ответов, записанных в кэш
bytes_written	Число; суммарное количество байт, записанных в кэш

В Angie PRO при включении *шардинга кэша* с помощью директив *proxy_cache_path* отдельные шарды указываются как объекты-члены в объекте **shards**:

shards	Объект; его члены — отдельные шарды
<shard>	Объект; представляет отдельный шард, а имя объекта — путь кэша
size	Число; текущий размер шарда
max_size	Число; максимальный размер шарда, если задан в конфигурации
cold	Логическое значение; true , пока <i>загрузчик кэша</i> подгружает данные с диска

```
{
  "name_zone": {
    "shards": {
      "/path/to/shard1": {
        "size": 0,
        "cold": false
      },
      "/path/to/shard2": {
        "size": 0,
        "cold": false
      }
    }
  }
}
```

limit_conn

```
limit_conn_zone $binary_remote_addr zone=limit_conn_zone:10m;
```

/status/http/limit_conns/<зона>, /status/stream/limit_conns/<зона>

Каждая из сконфигурированных зон: *limit_conn в http* или *limit_conn в stream* содержит следующие данные:

```
{
  "passed": 73,
  "skipped": 0,
  "rejected": 0,
  "exhausted": 0
}
```

passed	Число; суммарное количество переданных на проксируемый сервер соединений
skipped	Число; суммарное количество соединений, переданных с нулевым или превосходящим 255 байт <key>
rejected	Число; суммарное количество соединений сверх сконфигурированного ограничения
exhausted	Число; суммарное количество соединений, сброшенных из-за переполнения хранилища зоны

limit_req

```
limit_req_zone $binary_remote_addr zone=limit_req_zone:10m rate=1r/s;
```

/status/http/limit_reqs/<зона>

Каждая из сконфигурированных зон *limit_req* содержит следующие данные:

```
{
  "passed": 54816,
  "skipped": 0,
  "delayed": 65,
  "rejected": 26,
  "exhausted": 0
}
```

passed	Число; суммарное количество проксированных соединений
skipped	Число; суммарное количество соединений, переданных с нулевым или превосходящим 255 байт <key>
delayed	Число; суммарное количество задержанных соединений
rejected	Число; суммарное количество сброшенных соединений
exhausted	Число; суммарное количество соединений, сброшенных из-за переполнения хранилища зоны

HTTP upstream

Чтобы включить сбор следующих метрик, задайте директиву *zone* в контексте *upstream*, например:

```
upstream upstream {
  zone upstream 256k;
  server backend.example.com service=_example._tcp resolve max_conns=5;
  keepalive 4;
}
```

/status/http/upstreams/<upstream>

где <upstream> — имя *апстрима*, в конфигурации которого указана директива *zone*.

```
{
  "peers": {
    "192.168.16.4:80": {
      "server": "backend.example.com",
      "service": "_example._tcp",
      "backup": false,
      "weight": 5,
      "state": "up",
      "selected": {
        "current": 2,
        "total": 232
      },
      "max_conns": 5,
    }
  }
}
```

```
    "responses": {
      "200": 222,
      "302": 12
    },

    "data": {
      "sent": 543866,
      "received": 27349934
    },

    "health": {
      "fails": 0,
      "unavailable": 0,
      "downtime": 0
    },

    "sid": "<server_id>"
  },

  "keepalive": 2
}
```

peers	Объект; содержит метрики всех пиров апстрима во вложенных объектах, имена которых — канонические представления адресов этих пиров. Внутри каждого вложенного объекта:
server	Строка; сервер, как он указан в директиве <i>server</i>
service	Строка; имя сервиса, указанное в директиве <i>server</i> , если сконфигурировано
slow_start	Число; заданное для сервера значение <i>slow_start</i> , выраженное в секундах. При задании значения через <i>соответствующий подраздел</i> API динамической конфигурации можно указать не только число секунд, но и <i>время</i> с точностью до миллисекунд.
backup	Логическое значение; true для backup серверов
weight	Число; сконфигурированный <i>weight</i>
state	Строка; текущее состояние пира, и какие запросы ему отправляются: • busy: указывает, что число запросов на сервер достигло ограничения, заданного <i>max_conns</i>, и новые запросы на него не отправляются; • down: отключен вручную, не отправляются никакие запросы; • recovering: восстанавливается после сбоя согласно <i>slow_start</i>, отправляются все больше запросов; • unavailable: достиг предела <i>max_fails</i>, отправляются пробные клиентские запросы с интервалом <i>fail_timeout</i>; • up: работоспособен, запросы отправляются как обычно; Дополнительные состояния в Angie PRO: • checking: настроен как essential и проверяется, отправляются только <i>проверочные запросы</i>; • draining: аналогичен down, но отправляются запросы сессий, привязанных ранее через <i>sticky</i>; • unhealthy: неработающий, отправляются только <i>проверочные запросы</i>.
selected	Объект; статистика выбора пиров
current	Число; текущее количество соединений к пиру
total	Число; общее количество запросов переданных пиру
last	Строка или число; время последнего выбора пира в формате <i>даты</i>
max_conns	Число; <i>максимальное</i> количество одновременных активных соединений к пиру, если сконфигурировано
responses	Объект; статистика ответов
<code>	Число; ненулевое количество ответов со статусом <code> (100-599)
xxx	Число; ненулевое количество ответов с другим кодом статуса
data	Объект; метрики данных
received	Число; суммарное количество байт, полученное от пира
sent	Число; суммарное количество байт, отправленное пиру
health	Объект; статистика по состоянию пира
fails	Число; общее количество неудачных попыток работы с пиром
unavailable	Число; столько раз пир становился unavailable по достижении значения <i>max_fails</i>
downtime	Число; суммарное время (в миллисекундах), в течение которого пир был недоступен для выбора как unavailable
downstart	Строка или число; время, когда пир стал unavailable , в формате <i>даты</i>
header_time	Число; среднее время (в миллисекундах) получения заголовков ответа от сервера; см. директиву <i>response_time_factor</i>
response_time	Число; среднее время (в миллисекундах) получения ответа от сервера; см. директиву <i>response_time_factor</i>
sid	Строка; <i>id сервера</i> , указанный в конфигурации апстрима
keepalive	Число; текущее количество кэшированных соединений
backup_switch	Объект; содержит текущее состояние логики активного резервирования, присутствует, если для апстрима настроен <i>backup_switch</i>
active	Число; уровень активной группы, которая сейчас используется для балансировки запросов. Если активная группа является основной, значение равно 0
timeout	Число; оставшееся время ожидания в миллисекундах, после которого балансировщик перепроверит на наличие здоровых узлов группы с

health/probes

Изменено в версии 1.2.0: PRO

Если для апстрима настроены проверки *upstream_probe*, то в объекте **health** также есть вложенный объект **probes**, содержащий счетчики проверок работоспособности сервера, а **state**, помимо значений из таблицы выше, может принимать значения **checking** и **unhealthy**:

```
{
  "192.168.16.4:80": {
    "state": "unhealthy",
    "...": "...",
    "health": {
      "...": "...",
      "probes": {
        "count": 10,
        "fails": 10,
        "last": "|sampledateshort|T09:56:07Z"
      }
    }
  }
}
```

Значение **checking** у **state** не учитывается в **downtime** и означает, что сервер, проверка которого настроена с параметром **essential**, еще не проверялся; значение **unhealthy** — что сервер неработающий. Оба эти состояния также означают, что сервер не участвует в балансировке. Детали проверок см. в описании *upstream_probe*.

Счетчики в **probes**:

count	Число; общее количество проверок этого сервера
fails	Число; количество неуспешных проверок
last	Строка или число; время последней проверки в формате <i>даты</i>

queue

Изменено в версии 1.4.0: PRO

Если для апстрима настроена *очередь запросов*, то в объекте апстрима также есть вложенный объект **queue**, содержащий счетчики запросов в очереди:

```
{
  "queue": {
    "queued": 20112,
    "waiting": 1011,
    "dropped": 6031,
    "timedout": 560,
    "overflows": 13
  }
}
```

Значения счетчиков суммируются по всем рабочим процессам:

queued	Число; общее количество запросов, попавших в очередь
waiting	Число; текущее количество запросов в очереди
dropped	Число; общее количество запросов, удаленных из очереди из-за того, что клиент преждевременно закрыл соединение
timedout	Число; общее количество запросов, удаленных из очереди по таймауту
overflows	Число; общее количество случаев переполнения очереди

Stream upstream

Чтобы включить сбор следующих метрик, задайте директиву `zone` в контексте `upstream`, например:

```
upstream upstream {
    zone upstream 256k;
    server backend.example.com service=_example._tcp resolve max_conns=5;
    keepalive 4;
}
```

`/status/stream/upstreams/<upstream>`

Здесь `<upstream>` — имя *апстрима*, в конфигурации которого использована директива `zone`.

```
{
  "peers": {
    "192.168.16.4:1935": {
      "server": "backend.example.com",
      "service": "_example._tcp",
      "backup": false,
      "weight": 5,
      "state": "up",
      "selected": {
        "current": 2,
        "total": 232
      },
      "max_conns": 5,
      "data": {
        "sent": 543866,
        "received": 27349934
      },
      "health": {
        "fails": 0,
        "unavailable": 0,
        "downtime": 0
      }
    }
  }
}
```

peers	Объект; содержит метрики всех пиров апстрима во вложенных объектах, имена которых — канонические представления адресов этих пиров. Внутри каждого вложенного объекта:
server	Строка; адрес, заданный директивой <i>server</i>
service	Строка; имя сервиса, если оно указано в директиве <i>server</i>
slow_start	Число; заданное для сервера значение <i>slow_start</i> , выраженное в секундах. При задании значения через <i>соответствующий подраздел</i> API динамической конфигурации можно указать не только число секунд, но и <i>время</i> с точностью до миллисекунд.
backup	Логическое значение; true для запасных серверов
weight	Число; заданный для пира <i>вес</i>
state	Строка; текущее состояние пира, и какие запросы ему отправляются: • busy: указывает, что число запросов на сервер достигло ограничения, заданного <i>max_conns</i>, и новые запросы на него не отправляются; • down: отключен вручную, не отправляются никакие запросы; • recovering: восстанавливается после сбоя согласно <i>slow_start</i>, отправляются все больше запросов; • unavailable: достиг предела <i>max_fails</i>, отправляются пробные клиентские запросы с интервалом <i>fail_timeout</i>; • up: работоспособен, запросы отправляются как обычно. Дополнительные состояния в Angie PRO: • checking: настроен как essential и проверяется, отправляются только <i>проверочные запросы</i>; • draining: аналогичен down, но отправляются запросы сессий, привязанных ранее через <i>sticky</i>; • unhealthy: неработающий, отправляются только <i>проверочные запросы</i>.
selected	Объект; статистика выбора этого пира для подключения
current	Число; текущее количество подключений к пиру
total	Число; общее количество подключений, направленных этому пиру
last	Строка или число; время, когда пир был выбран в последний раз, в формате <i>даты</i>
max_conns	Число; <i>максимальное</i> количество одновременных активных подключений к пиру, если оно задано
data	Объект; статистика передачи данных
received	Число; общее количество байт, полученное от пира
sent	Число; общее количество байт, отправленное пиру
health	Объект; статистика по состоянию пира
fails	Число; общее количество неудачных попыток связаться с пиром
unavailable	Число; общее количество переходов в состояние unavailable по достижении значения <i>max_fails</i>
downtime	Число; общее время в миллисекундах, в течение которого пир находился в состоянии unavailable (недоступен для выбора)
downstart	Строка или число; время, когда пир последний раз стал unavailable , в формате <i>даты</i>
connect_time	Число; среднее время (в миллисекундах) установления соединения с сервером; см. директиву <i>response_time_factor</i>
first_byte_time	Число; среднее время (в миллисекундах) получения первого байта ответа от сервера; см. директиву <i>response_time_factor</i>
last_byte_time	Число; среднее время (в миллисекундах) получения полного ответа от сервера; см. директиву <i>response_time_factor</i>
backup_switch (PRO 1.10.0+)	Объект; содержит текущее состояние логики активного резервирования, присутствует, если для апстрима настроен <i>backup_switch</i>
active	Число; уровень активной группы, которая сейчас используется для балансировки запросов. Если активная группа является основной, значение равно 0

3.6. t **Справочная информация**; оставшееся время ожидания в миллисекундах, после которого балансировщик перепроверит на наличие здоровых узлов группы с меньшим уровнем, начиная с основной, а группы с большим уровнем не проверяются; не отображается для основной группы (уровень 0)

Изменено в версии 1.4.0: PRO

Если в Angie PRO для апстрима настроены проверки *upstream_probe*, то в объекте **health** также есть вложенный объект **probes**, содержащий счетчики проверок работоспособности сервера, а **state**, помимо значений из таблицы выше, может принимать значения **checking** и **unhealthy**:

```
{
  "192.168.16.4:80": {
    "state": "unhealthy",
    "...": "...",
    "health": {
      "...": "...",
      "probes": {
        "count": 2,
        "fails": 2,
        "last": "|sampledateshort|T11:03:54Z"
      }
    }
  }
}
```

Значение **checking** у **state** означает, что сервер, проверка которого настроена с параметром **essential**, еще не проверялся; значение **unhealthy** — что сервер неработающий. Оба эти состояния также означают, что сервер не участвует в балансировке. Детали проверок см. в описании *upstream_probe*.

Счетчики в **probes**:

count	Число; общее количество проверок этого сервера
fails	Число; количество неуспешных проверок
last	Строка или число; время последней проверки в формате <i>даты</i>

API динамической конфигурации

В составе API есть раздел **/config**, позволяющий динамически менять конфигурацию Angie в формате JSON с помощью HTTP-запросов **PUT**, **PATCH** и **DELETE**. Все изменения атомарны: новые настройки применяются целиком либо не применяются вовсе. При ошибке Angie сообщит, в чем причина.

Подразделы **/config**

Сейчас в разделе **/config** доступна настройка отдельных серверов в составе апстримов для модулей *HTTP* и *stream*; число настроек, к которым применима динамическая конфигурация, плавно увеличивается.

/config/http/upstreams/<upstream>/servers/<имя>

Позволяет настраивать отдельные серверы в составе апстрима, в том числе добавлять новые и удалять настроенные.

Параметры в составе пути URI:

<code><upstream></code>	Имя блока <code>upstream</code> ; чтобы настраивать его через <code>/config</code> , в нем должна быть задана директива <code>zone</code> , определяющая зону разделяемой памяти.
<code><имя></code>	Имя конкретного сервера в составе указанного <code><upstream></code> ; задается в формате <code><service>@<host></code> , где: • <code><service>@</code> — необязательная часть, задающая имя сервиса в целях разрешения SRV-записей. • <code><host></code> — доменное имя сервиса (при наличии <code>resolve</code>) или IP-адрес; также можно указать порт.

Например, для следующей конфигурации:

```
upstream backend {
    server backend.example.com service=_http._tcp resolve;
    server 127.0.0.1;
    zone backend 1m;
}
```

Допустимы такие имена серверов:

```
$ curl http://127.0.0.1/config/http/upstreams/backend/servers/_http._tcp@backend.
→example.com/
$ curl http://127.0.0.1/config/http/upstreams/backend/servers/127.0.0.1:80/
```

Этот подраздел API позволяет задавать параметры `weight`, `max_conns`, `max_fails`, `fail_timeout`, `backup`, `down` и `sid`, описанные в разделе *server*.

Примечание

Отдельного параметра `drain` здесь нет; включить режим `drain` можно, задав для `down` строковое значение `drain`:

```
$ curl -X PUT -d "\"drain\" \" \
http://127.0.0.1/config/http/upstreams/backend/servers/backend.example.com/down
```

Пример:

```
$ curl http://127.0.0.1/config/http/upstreams/backend/servers/backend.example.com?
→defaults=on
```

```
{
  "weight": 1,
  "max_conns": 0,
  "max_fails": 1,
  "fail_timeout": 10,
  "backup": true,
  "down": false,
  "sid": ""
}
```

Фактически доступны будут только те параметры, которые поддерживает текущий метод балансировки нагрузки *апстрима*. Так, если апстрим настроен с методом балансировки `random`:

```
upstream backend {
    zone backend 256k;
```

```
server backend.example.com resolve max_conns=5;
random;
}
```

То добавить в него новый сервер с параметром `backup` невозможно:

```
$ curl -X PUT -d '{"backup": true}' \
  http://127.0.0.1/config/http/upstreams/backend/servers/backend1.example.com
```

```
{
  "error": "FormatError",
  "description": "The \"backup\" field is unknown."
}
```

Примечание

Даже с совместимым методом балансировки параметр `backup` можно задать лишь при добавлении нового сервера.

`/config/stream/upstreams/<upstream>/servers/<имя>`

Позволяет настраивать отдельные серверы в составе апстрима, в том числе добавлять новые и удалять настроенные.

Параметры в составе пути URI:

<code><upstream></code>	Имя блока <code>upstream</code> ; чтобы настраивать его через <code>/config</code> , в нем должна быть задана директива <code>zone</code> , определяющая зону разделяемой памяти.
<code><имя></code>	Имя конкретного сервера в составе указанного <code><upstream></code> ; задается в формате <code><service>@<host></code> , где: <code><service>@</code> — необязательная часть, задающая имя сервиса в целях разрешения SRV-записей. <code><host></code> — доменное имя сервиса (при наличии <code>resolve</code>) или IP-адрес; также можно указать порт.

Например, для следующей конфигурации:

```
upstream backend {
  server backend.example.com:8080 service=_example._tcp resolve;
  server 127.0.0.1:12345;
  zone backend 1m;
}
```

Допустимы такие имена серверов:

```
$ curl http://127.0.0.1/config/stream/upstreams/backend/servers/_example._tcp@backend.
↪example.com:8080/
$ curl http://127.0.0.1/config/stream/upstreams/backend/servers/127.0.0.1:12345/
```

Этот подраздел API позволяет задавать параметры `weight`, `max_conns`, `max_fails`, `fail_timeout`, `backup` и `down`, описанные в разделе `server`.

Примечание

Отдельного параметра **drain** здесь нет; включить режим **drain** можно, задав для **down** строковое значение **drain**:

```
$ curl -X PUT -d \"drain\" \
http://127.0.0.1/config/stream/upstreams/backend/servers/backend.example.com/down
```

Пример:

```
curl http://127.0.0.1/config/stream/upstreams/backend/servers/backend.example.com?
↳defaults=on
```

```
{
  "weight": 1,
  "max_conns": 0,
  "max_fails": 1,
  "fail_timeout": 10,
  "backup": true,
  "down": false,
}
```

Фактически доступны будут только те параметры, которые поддерживает текущий метод балансировки нагрузки *апстрима*. Так, если апстрим настроен с методом балансировки **random**:

```
upstream backend {
  zone backend 256k;
  server backend.example.com resolve max_conns=5;
  random;
}
```

То добавить в него новый сервер с параметром **backup** невозможно:

```
$ curl -X PUT -d '{ "backup": true }' \
http://127.0.0.1/config/stream/upstreams/backend/servers/backend1.example.com
```

```
{
  "error": "FormatError",
  "description": "The \"backup\" field is unknown."
}
```

Примечание

Даже с совместимым методом балансировки параметр **backup** можно задать лишь при добавлении нового сервера.

При удалении серверов можно установить аргумент **connection_drop=<значение>**, чтобы переопределить настройки *proxy_connection_drop*:

```
$ curl -X DELETE \
http://127.0.0.1/config/stream/upstreams/backend/servers/backend1.example.com?
↳connection_drop=off

$ curl -X DELETE \
http://127.0.0.1/config/stream/upstreams/backend/servers/backend2.example.com?
↳connection_drop=on
```

```
$ curl -X DELETE \
    http://127.0.0.1/config/stream/upstreams/backend/servers/backend3.example.com?
↪connection_drop=1000
```

HTTP-методы

Рассмотрим семантику каждого из применимых к этому разделу HTTP-методов на примере следующей конфигурации апстрима:

```
http {
    # ...

    upstream backend {
        zone upstream 256k;
        server backend.example.com resolve max_conns=5;
        # ...
    }

    server {
        # ...

        location /config/ {
            api /config/;

            allow 127.0.0.1;
            deny all;
        }
    }
}
```

GET

HTTP-метод GET позволяет запросить сущность по любому существующему пути в пределах /config так же, как это делается в других разделах API.

Например, для ветки серверов апстрима /config/http/upstreams/backend/servers/ допустимы такие запросы:

```
$ curl http://127.0.0.1/config/http/upstreams/backend/servers/backend.example.com/max_
↪conns
$ curl http://127.0.0.1/config/http/upstreams/backend/servers/backend.example.com
$ curl http://127.0.0.1/config/http/upstreams/backend/servers
$ # ...
$ curl http://127.0.0.1/config
```

Получить параметры по умолчанию можно с аргументом defaults=on:

```
$ curl http://127.0.0.1/config/http/upstreams/backend/servers?defaults=on
```

```
{
    "backend.example.com": {
        "weight": 1,
        "max_conns": 5,
        "max_fails": 1,
```

```

    "fail_timeout": 10,
    "backup": false,
    "down": false,
    "sid": ""
  }
}

```

PUT

HTTP-метод PUT позволяет создать новую JSON-сущность по указанному пути или *полностью* заменить существующую.

Например, чтобы добавить не заданный ранее параметр `max_fails` у сервера `backend.example.com` в апстриме `backend`:

```

$ curl -X PUT -d '2' \
  http://127.0.0.1/config/http/upstreams/backend/servers/backend.example.com/max_
↪fails

```

```

{
  "success": "Updated",
  "description": "Existing configuration API entity \"/config/http/upstreams/
↪backend/servers/backend.example.com/max_fails\" was updated with replacing."
}

```

Проверим изменения:

```

$ curl http://127.0.0.1/config/http/upstreams/backend/servers/backend.example.com

```

```

{
  "max_conns": 5,
  "max_fails": 2
}

```

DELETE

HTTP-метод DELETE удаляет *ранее заданные* настройки по указанному пути; при этом восстанавливаются значения по умолчанию, если они есть.

Например, чтобы удалить измененный ранее параметр `max_fails` у сервера `backend.example.com` в апстриме `backend`:

```

$ curl -X DELETE \
  http://127.0.0.1/config/http/upstreams/backend/servers/backend.example.com/max_
↪fails

```

```

{
  "success": "Reset",
  "description": "Configuration API entity \"/config/http/upstreams/backend/servers/
↪backend.example.com/max_fails\" was reset to default."
}

```

Проверим изменения с аргументом `defaults=on`:


```
$ curl http://127.0.0.1/config/http/upstreams/backend/servers/backend.example.com?
↳defaults=on
```

```
{
  "weight": 1,
  "max_conns": 5,
  "max_fails": 1,
  "fail_timeout": 10,
  "backup": false,
  "down": false,
  "sid": ""
}
```

Параметр `max_fails` вернулся к значению по умолчанию.

При удалении серверов можно установить аргумент `connection_drop=<значение>`, чтобы перепределить настройки `proxy_connection_drop`, `grpc_connection_drop`, `fastcgi_connection_drop`, `scgi_connection_drop` и `uwsgi_connection_drop`:

```
$ curl -X DELETE \
  http://127.0.0.1/config/http/upstreams/backend/servers/backend1.example.com?
↳connection_drop=off

$ curl -X DELETE \
  http://127.0.0.1/config/http/upstreams/backend/servers/backend2.example.com?
↳connection_drop=on

$ curl -X DELETE \
  http://127.0.0.1/config/http/upstreams/backend/servers/backend3.example.com?
↳connection_drop=1000
```

PATCH

HTTP-метод `PATCH` позволяет создать новую сущность по указанному пути либо частично заменить или дополнить существующую ([RFC 7386](#)), отправив в данных JSON-определение.

Метод работает так: если сущности, указанные в новом определении, уже есть в конфигурации, они будут перезаписаны; если их нет, то они будут добавлены.

Например, чтобы поменять значение параметра `down` у сервера `backend.example.com` в апстриме `backend`, оставив прочее без изменений:

```
$ curl -X PATCH -d '{"down": true}' \
  http://127.0.0.1/config/http/upstreams/backend/servers/backend.example.com
```

```
{
  "success": "Updated",
  "description": "Existing configuration API entity \"/config/http/upstreams/
↳backend/servers/backend.example.com\" was updated with merging."
}
```

Проверим изменения:

```
$ curl http://127.0.0.1/config/http/upstreams/backend/servers/backend.example.com
```

```
{
  "max_conns": 5,
```

```
"down": true
}
```

Обратите внимание, что переданный с запросом PATCH JSON-объект *слился* с уже существующим, а не заменил его целиком, как было бы с PUT.

Особый случай представляют значения `null`; они используются для удаления отдельных элементов конфигурации в ходе такого слияния.

Примечание

Такое удаление аналогично действию DELETE; в частности, восстанавливаются значения по умолчанию.

Например, чтобы удалить добавленный ранее параметр `down` и одновременно с этим изменить `max_conns`:

```
$ curl -X PATCH -d '{ "down": null, "max_conns": 6 }' \
  http://127.0.0.1/config/http/upstreams/backend/servers/backend.example.com
```

```
{
  "success": "Updated",
  "description": "Existing configuration API entity \"/config/http/upstreams/
  ↪backend/servers/backend.example.com\" was updated with merging."
}
```

Проверим изменения:

```
$ curl http://127.0.0.1/config/http/upstreams/backend/servers/backend.example.com
```

```
{
  "max_conns": 6
}
```

Параметр `down`, для которого было передано значение `null`, удален; значение `max_conns` изменено.

Auth Basic

Позволяет ограничить доступ к ресурсам с проверкой имени и пароля пользователя по протоколу "HTTP Basic Authentication".

Ограничить доступ можно также по *адресу* или по *результату подзапроса*. Одновременное ограничение доступа по адресу и паролю управляется директивой *satisfy*.

Пример конфигурации

```
location / {
    auth_basic          "closed site";
    auth_basic_user_file conf/htpasswd;
}
```

Директивы

auth_basic

Синтаксис	auth_basic строка off;
По умолчанию	auth_basic off;
Контекст	http, server, location, limit_except

Включает проверку имени и пароля пользователя по протоколу "HTTP Basic Authentication". Заданный параметр используется в качестве *realm*. В значении параметра допустимо использование переменных.

off	отменяет действие унаследованной с предыдущего уровня конфигурации директивы <i>auth_basic</i>

auth_basic_user_file

Синтаксис	auth_basic_user_file файл;
По умолчанию	—
Контекст	http, server, location, limit_except

Задаёт *файл*, в котором хранятся имена и пароли пользователей. Формат файла следующий:

```
# комментарий
имя1:пароль1
имя2:пароль2:комментарий
имя3:пароль3
```

В имени *файла* можно использовать переменные.

Поддерживаются следующие типы паролей:

- зашифрованные функцией *crypt()*; могут быть созданы с помощью утилиты **htpasswd** из дистрибутива HTTP-сервера Apache или команды `"openssl passwd"`;
- хэшированные с помощью алгоритма, основанного на MD5, по версии Apache (apr1); могут быть созданы теми же инструментами;
- заданные согласно синтаксису "*{схема}данные*" как описано в [RFC 2307](#); в настоящий момент реализованы схемы *PLAIN* (в качестве примера, не следует применять), *SHA* (простое SHA-1 хэширование, не следует применять) и *SSHA* (SHA-1 хэширование с солью, используется в некоторых программах, в частности OpenLDAP и Dovecot).

Предупреждение

Поддержка схемы *SHA* была добавлена лишь для облегчения процесса миграции файлов паролей с других веб-серверов. Ее не следует применять для новых паролей, т.к. используемое при этом SHA-1 хэширование без соли уязвимо к взлому при помощи [радужных таблиц](#).

Auth Request

Предоставляет возможность авторизации клиента, основанной на результате подзапроса. Если подзапрос возвращает код ответа 2xx, доступ разрешается. Если 401 или 403 — доступ запрещается с соответствующим кодом ошибки. Любой другой код ответа, возвращаемый подзапросом, считается ошибкой.

При ошибке 401 клиенту также передается заголовок **WWW-Authenticate** из ответа подзапроса.

При сборке из исходного кода модуль не собирается по умолчанию; его необходимо включить с помощью параметра сборки `--with-http_auth_request_module`. В пакетах и образах из наших репозиториях модуль включен в сборку.

Модуль может быть скомбинирован с другими модулями доступа, такими как *Access* и *Auth Basic* с помощью директивы *satisfy*.

Пример конфигурации

```
location /private/ {
    auth_request /auth;
    # ...
}

location = /auth {
    proxy_pass ...;
    proxy_pass_request_body off;
    proxy_set_header Content-Length "";
    proxy_set_header X-Original-URI $request_uri;
}
```

Директивы

auth_request

<i>Синтаксис</i>	<code>auth_request uri off;</code>
По умолчанию	<code>auth_request off;</code>
<i>Контекст</i>	http, server, location

Включает авторизацию, основанную на результате выполнения подзапроса, и задает URI, на который будет отправлен подзапрос.

auth_request_set

<i>Синтаксис</i>	<code>auth_request_set \$переменная значение;</code>
По умолчанию	—
<i>Контекст</i>	http, server, location

Устанавливает переменную в запросе в заданное значение после завершения запроса авторизации. Значение может содержать переменные из запроса авторизации, например, `$upstream_http_*`.

AutoIndex

Обслуживает запросы, оканчивающиеся косой чертой (/), и выдает листинг каталога. Обычно запрос попадает к модулю AutoIndex, когда модуль *Index* не нашел индексный файл.

Пример конфигурации

```
location / {
    autoindex on;
}
```

Директивы

autoindex

<i>Синтаксис</i>	<code>autoindex on off;</code>
По умолчанию	<code>autoindex off;</code>
<i>Контекст</i>	http, server, location

Разрешает или запрещает вывод листинга каталога.

autoindex_exact_size

<i>Синтаксис</i>	<code>autoindex_exact_size on off;</code>
По умолчанию	<code>autoindex_exact_size on;</code>
<i>Контекст</i>	http, server, location

Для *формата* HTML определяет, как выводить размеры файлов в листинге каталога: точно или округляя до килобайт, мегабайт и гигабайт.

autoindex_format

<i>Синтаксис</i>	<code>autoindex_format html xml json jsonp;</code>
По умолчанию	<code>autoindex_format html;</code>
<i>Контекст</i>	http, server, location

Задаёт формат вывода листинга каталога.

При использовании формата JSONP имя callback-функции задается в аргументе запроса `callback`. Если аргумент отсутствует или имеет пустое значение, то используется формат JSON.

Вывод в формате XML может быть преобразован при помощи модуля XSLT.

Форматы вывода

Поля объекта в ответах содержат следующие данные:

Поле	Описание
name	Имя файла или каталога
type	Тип объекта: file или directory
size	Размер объекта согласно <i>autoindex_exact_size</i> ; для каталогов — 0
mtime	Время последнего изменения в формате Unix time

HTML

```
<html>
<head>
  <title>Index of /files/</title>
</head>
<body>
  <h1>Index of /files/</h1>
  <hr>
  <pre>
    <a href="..">../</a>
    <a href="example.txt">example.txt</a>                12-Jun-2025  14:21  ┐
↪1234
    <a href="image.png">image.png</a>                12-Jun-2025  14:21  ┐
↪4321
  </pre>
  <hr>
</body>
</html>
```

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<listing>
<file>
  <name>example.txt</name>
  <type>file</type>
  <size>1234</size>
  <mtime>2025-06-12T14:21:00Z</mtime>
</file>
<file>
  <name>image.png</name>
  <type>file</type>
  <size>4321</size>
  <mtime>2025-06-12T14:21:00Z</mtime>
</file>
</listing>
```

JSON

```
[
{
  "name": "example.txt",
  "type": "file",
  "size": 1234,
  "mtime": "2025-06-12T14:21:00Z"
},
```

```
{
  "name": "image.png",
  "type": "file",
  "size": 4321,
  "mtime": "2025-06-12T14:21:00Z"
}
```

JSONP

```
callback([
{
  "name": "example.txt",
  "type": "file",
  "size": 1234,
  "mtime": "2025-06-12T14:21:00Z"
},
{
  "name": "image.png",
  "type": "file",
  "size": 4321,
  "mtime": "2025-06-12T14:21:00Z"
}
]);
```

autoindex_localtime

<i>Синтаксис</i>	autoindex_localtime on off;
По умолчанию	autoindex_localtime off;
<i>Контекст</i>	http, server, location

Для *формата* HTML определяет, в какой временной зоне выводить время в листинге каталога: в локальной или в UTC.

Browser

Создает переменные, значения которых зависят от значения поля **User-Agent** в заголовке запроса.

Переменные

`$modern_browser`

равна значению, заданному директивой *modern_browser_value*, если браузер опознан как современный;

`$ancient_browser`

равна значению, заданному директивой `ancient_browser_value`, если браузер опознан как устаревший;

`$msie`

равна "1", если браузер опознан как MSIE любой версии.

Пример конфигурации

Выбор индексного файла:

```
modern_browser_value "modern.";

modern_browser msie      5.5;
modern_browser gecko     1.0.0;
modern_browser opera     9.0;
modern_browser safari    413;
modern_browser konqueror 3.0;

index index.${modern_browser}.html index.html;
```

Перенаправление для старых браузеров:

```
modern_browser msie      5.0;
modern_browser gecko     0.9.1;
modern_browser opera     8.0;
modern_browser safari    413;
modern_browser konqueror 3.0;

modern_browser unlisted;

ancient_browser Links Lynx netscape4;

if ($ancient_browser) {
    rewrite ^ /ancient.html;
}
```

Директивы

`ancient_browser`

<i>Синтаксис</i>	<code>ancient_browser строка ...;</code>
По умолчанию	—
<i>Контекст</i>	http, server, location

Задаёт подстроки, при нахождении которых в поле **User-Agent** заголовка запроса браузер считается устаревшим. Специальная строка "netscape4" соответствует регулярному выражению "^Mozilla/[1-4]".

ancient_browser_value

<i>Синтаксис</i>	<code>ancient_browser_value строка;</code>
По умолчанию	<code>ancient_browser_value 1;</code>
<i>Контекст</i>	http, server, location

Задаёт значение для переменных *\$ancient_browser*.

modern_browser

<i>Синтаксис</i>	<code>modern_browser браузер версия;</code> <code>modern_browser unlisted;</code>
По умолчанию	—
<i>Контекст</i>	http, server, location

Задаёт версию браузера, начиная с которой он считается современным. В качестве браузера можно задать *msie*, *gecko* (браузеры, созданные на основе Mozilla), *opera*, *safari* или *konqueror*.

Версии можно задать в форматах X, X.X, X.X.X или X.X.X.X. Максимальные значения для каждого из форматов соответственно — 4000, 4000.99, 4000.99.99 и 4000.99.99.99.

Специальное значение **unlisted** указывает считать современным браузер, не описанный директивами *modern_browser* и *ancient_browser*. В противном случае неперечисленный браузер будет считаться устаревшим. Если в заголовке запроса нет поля **User-Agent**, то браузер считается неперечисленным.

modern_browser_value

<i>Синтаксис</i>	<code>modern_browser_value строка;</code>
По умолчанию	<code>modern_browser_value 1;</code>
<i>Контекст</i>	http, server, location

Задаёт значение для переменных *\$modern_browser*.

Charset

Добавляет указанную кодировку в поле **Content-Type** заголовка ответа. Кроме того, модуль может перекодировать данные из одной кодировки в другую с некоторыми ограничениями:

- перекодирование осуществляется только в одну сторону — от сервера к клиенту,
- перекодироваться могут только однобайтные кодировки
- или однобайтные кодировки в UTF-8 и обратно.

Пример конфигурации

```
include      conf/koi-win;

charset      windows-1251;
source_charset koi8-r;
```

Директивы

charset

<i>Синтаксис</i>	<code>charset кодировка off;</code>
По умолчанию	<code>charset off;</code>
<i>Контекст</i>	http, server, location, if в location

Добавляет указанную кодировку в поле **Content-Type** заголовка ответа. Если эта кодировка отличается от указанной в директиве `source_charset`, то выполняется перекодирование.

Параметр `off` отменяет добавление кодировки в поле **Content-Type** заголовка ответа.

Кодировка может быть задана с помощью переменной:

```
charset $charset;
```

В этом случае необходимо, чтобы все возможные значения переменной присутствовали хотя бы один раз в любом месте конфигурации в виде директив `charset_map`, `charset` или `source_charset`. Для кодировок `utf-8`, `windows-1251` и `koi8-r` для этого достаточно включить в конфигурацию файлы `conf/koi-win`, `conf/koi-utf` и `conf/win-utf`. Для других кодировок можно просто сделать фиктивную таблицу перекодировки, например:

```
charset_map iso-8859-5 _ { }
```

Кроме того, кодировка может быть задана в поле **X-Accel-Charset** заголовка ответа. Эту возможность можно запретить с помощью директив `proxy_ignore_headers`, `fastcgi_ignore_headers`, `uwsgi_ignore_headers`, `scgi_ignore_headers` и `grpc_ignore_headers`.

charset_map

<i>Синтаксис</i>	<code>charset_map кодировка1 кодировка2 { ... }</code>
По умолчанию	—
<i>Контекст</i>	http

Описывает таблицу перекодирования из одной кодировки в другую. Таблица для обратного перекодирования строится на основании тех же данных. Коды символов задаются в шестнадцатеричном виде. Неописанные символы в пределах 80-FF заменяются на "?". При перекодировании из UTF-8 символы, отсутствующие в однобайтной кодировке, заменяются на "`&#XXXX;`".

Пример:

```
charset_map koi8-r windows-1251 {
    C0 FE ; # small yu
    C1 E0 ; # small a
```

```
C2 E1 ; # small b
C3 F6 ; # small ts
}
```

При описании таблицы перекодирования в UTF-8, коды кодировки UTF-8 должны быть указаны во второй колонке, например:

```
charset_map koi8-r utf-8 {
    C0 D18E ; # small yu
    C1 D0B0 ; # small a
    C2 D0B1 ; # small b
    C3 D186 ; # small ts
}
```

Полные таблицы преобразования из *koi8-r* в *windows-1251* и из *koi8-r* и *windows-1251* в *utf-8* входят в дистрибутив и находятся в файлах *conf/koi-win*, *conf/koi-utf* и *conf/win-utf*.

charset_types

<i>Синтаксис</i>	<code>charset_types mime-mun ...;</code>
По умолчанию	<code>charset_types text/html text/xml text/plain text/vnd.wap.wml application/javascript application/rss+xml;</code>
<i>Контекст</i>	<code>http, server, location</code>

Разрешает работу модуля в ответах с указанными MIME-типами в дополнение к `text/html`. Специальное значение `*` соответствует любому MIME-типу.

override_charset

<i>Синтаксис</i>	<code>override_charset on off;</code>
По умолчанию	<code>override_charset off;</code>
<i>Контекст</i>	<code>http, server, location, if в location</code>

Определяет, выполнять ли перекодирование для ответов, полученных от проксированного сервера или от FastCGI/uwsgi/SCGI/gRPC-сервера, если в ответах уже указана кодировка в поле **Content-Type** заголовка ответа. Если перекодирование разрешено, то в качестве исходной кодировки используется кодировка, указанная в полученном ответе.

Примечание

Если ответ был получен в подзапросе, то, независимо от значения директивы *override_charset*, всегда выполняется перекодирование из кодировки ответа в кодировку основного запроса.

source_charset

<i>Синтаксис</i>	<code>source_charset</code> <i>кодировка</i> ;
По умолчанию	—
<i>Контекст</i>	http, server, location, if в location

Задаёт исходную кодировку ответа. Если эта кодировка отличается от указанной в директиве `charset`, то выполняется перекодирование.

DAV

Предназначен для автоматизации задач управления файлами на сервере по протоколу WebDAV. Модуль обрабатывает HTTP- и WebDAV-методы PUT, DELETE, MKCOL, COPY и MOVE.

При сборке из исходного кода модуль не собирается по умолчанию; его необходимо включить с помощью параметра сборки `--with-http_dav_module`. В пакетах и образах из наших репозиториях модуль включен в сборку.

Примечание

WebDAV-клиенты, которые требуют для работы дополнительных WebDAV-методов, не будут работать с этим модулем.

Пример конфигурации

```
location / {
    root                /data/www;

    client_body_temp_path /data/client_temp;

    dav_methods PUT DELETE MKCOL COPY MOVE;

    create_full_put_path on;
    dav_access          group:rw all:r;

    limit_except GET {
        allow 192.168.1.0/32;
        deny  all;
    }
}
```

Директивы

create_full_put_path

<i>Синтаксис</i>	<code>create_full_put_path on off;</code>
По умолчанию	<code>create_full_put_path off;</code>
<i>Контекст</i>	<code>http, server, location</code>

По спецификации WebDAV-метод PUT может создавать файл только в уже существующем каталоге. Данная директива разрешает создавать все необходимые промежуточные каталоги.

dav_access

<i>Синтаксис</i>	<code>dav_access пользователи:права ...;</code>
По умолчанию	<code>dav_access user:rw;</code>
<i>Контекст</i>	<code>http, server, location</code>

Задаёт права доступа для создаваемых файлов и каталогов, например,

```
dav_access user:rw group:rw all:r;
```

Если заданы какие-либо права для group или all, то права для user указывать необязательно:

```
dav_access group:rw all:r;
```

dav_methods

<i>Синтаксис</i>	<code>dav_methods off метод ...;</code>
По умолчанию	<code>dav_methods off;</code>
<i>Контекст</i>	<code>http, server, location</code>

Разрешает указанные HTTP- и WebDAV-методы. Параметр `off` запрещает все методы, обрабатываемые данным модулем. Поддерживаются следующие методы: PUT, DELETE, MKCOL, COPY и MOVE.

Файл, загружаемый методом PUT, записывается во временный файл, а потом этот файл переименовывается. Временный файл и его постоянное место хранения могут располагаться на разных файловых системах. Однако нужно учитывать, что в этом случае вместо дешевой операции переименовывания в пределах одной файловой системы файл копируется с одной файловой системы на другую. Поэтому лучше, если сохраняемые файлы будут находиться на той же файловой системе, что и каталог с временными файлами, задаваемый директивой `client_body_temp_path` для данного location.

При создании файла с помощью метода PUT можно задать дату модификации, передав ее в поле заголовка Date.

min_delete_depth

<i>Синтаксис</i>	min_delete_depth число;
По умолчанию	min_delete_depth 0;
<i>Контекст</i>	http, server, location

Разрешает методу DELETE удалять файлы при условии, что число элементов в пути запроса не меньше заданного. Например, директива

```
min_delete_depth 4;
```

разрешает удалять файлы по запросам

```
/users/00/00/name
/users/00/00/name/pic.jpg
/users/00/00/page.html
```

и запрещает удаление

```
/users/00/00
```

Empty GIF

Выдает однопиксельный прозрачный GIF.

Пример конфигурации

```
location = /\.gif {
    empty_gif;
}
```

Директивы

empty_gif

<i>Синтаксис</i>	empty_gif;
По умолчанию	—
<i>Контекст</i>	location

Разрешает в содержащем location выдавать однопиксельный прозрачный GIF.

FastCGI

Позволяет передавать запросы FastCGI-серверу.

Пример конфигурации

```
location / {
    fastcgi_pass localhost:9000;
    fastcgi_index index.php;

    fastcgi_param SCRIPT_FILENAME /home/www/scripts/php$fastcgi_script_name;
    fastcgi_param QUERY_STRING $query_string;
    fastcgi_param REQUEST_METHOD $request_method;
    fastcgi_param CONTENT_TYPE $content_type;
    fastcgi_param CONTENT_LENGTH $content_length;
}
```

Директивы

fastcgi_bind

<i>Синтаксис</i>	<code>fastcgi_bind адрес [transparent] off;</code>
По умолчанию	—
<i>Контекст</i>	http, server, location

Задаёт локальный IP-адрес с необязательным портом, который будет использоваться в исходящих соединениях с FastCGI-сервером. В значении параметра допустимо использование переменных. Специальное значение **off** отменяет действие унаследованной с предыдущего уровня конфигурации директивы *fastcgi_bind*, позволяя системе самостоятельно выбирать локальный IP-адрес и порт.

Параметр **transparent** позволяет задать нелокальный IP-адрес, который будет использоваться в исходящих соединениях с FastCGI-сервером, например, реальный IP-адрес клиента:

```
fastcgi_bind $remote_addr transparent;
```

Для работы параметра обычно требуется запустить рабочие процессы Angie с привилегиями *superпользователя*. В Linux это не требуется, так как если указан параметр *transparent*, то рабочие процессы наследуют *capability CAP_NET_RAW* из главного процесса.

Примечание

Необходимо настроить таблицу маршрутизации ядра для перехвата сетевого трафика с FastCGI-сервера.

fastcgi_buffer_size

<i>Синтаксис</i>	<code>fastcgi_buffer_size размер;</code>
По умолчанию	<code>fastcgi_buffer_size 4k 8k;</code>
<i>Контекст</i>	http, server, location

Задаёт размер буфера, в который будет читаться первая часть ответа, получаемого от FastCGI-сервера. В этой части ответа обычно находится небольшой заголовок ответа. По умолчанию размер одного буфера равен размеру страницы памяти. В зависимости от платформы это или 4К, или 8К, однако его можно сделать меньше.

fastcgi_buffering

<i>Синтаксис</i>	<code>fastcgi_buffering on off;</code>
По умолчанию	<code>fastcgi_buffering on;</code>
<i>Контекст</i>	http, server, location

Разрешает или запрещает использовать буферизацию ответов FastCGI-сервера.

off	Angie принимает ответ FastCGI-сервера как можно быстрее, сохраняя его в буферы, заданные директивами <code>fastcgi_buffer_size</code> и <code>fastcgi_buffers</code> . Если ответ не влезает целиком в память, то его часть может быть записана на диск во <i>временный файл</i> . Запись во временные файлы контролируется директивами <code>fastcgi_max_temp_file_size</code> и <code>fastcgi_temp_file_write_size</code> .
on	ответ синхронно передается клиенту сразу же по мере его поступления. Angie не пытается считать весь ответ FastCGI-сервера. Максимальный размер данных, который Angie может принять от сервера за один раз, задается директивой <code>fastcgi_buffer_size</code> .

Буферизация может быть также включена или выключена путем передачи значения "yes" или "no" в поле X-Accel-Buffering заголовка ответа. Эту возможность можно запретить с помощью директивы `fastcgi_ignore_headers`.

fastcgi_buffers

<i>Синтаксис</i>	<code>fastcgi_buffers число размер;</code>
По умолчанию	<code>fastcgi_buffers 8 4k 8k;</code>
<i>Контекст</i>	http, server, location

Задаёт число и размер буферов для одного соединения, в которые будет читаться ответ, получаемый от FastCGI-сервера.

По умолчанию размер одного буфера равен размеру страницы. В зависимости от платформы это или 4К, или 8К.

fastcgi_busy_buffers_size

<i>Синтаксис</i>	<code>fastcgi_busy_buffers_size размер;</code>
По умолчанию	<code>fastcgi_busy_buffers_size 8k 16k;</code>
<i>Контекст</i>	<code>http, server, location</code>

При включенной *буферизации* ответов FastCGI-сервера, ограничивает суммарный размер буферов, которые могут быть заняты для отправки ответа клиенту, пока ответ еще не прочитан целиком. Оставшиеся буферы тем временем могут использоваться для чтения ответа и, при необходимости, буферизации части ответа во временный файл. По умолчанию размер ограничен двумя буферами, заданными директивами *fastcgi_buffer_size* и *fastcgi_buffers*.

fastcgi_cache

<i>Синтаксис</i>	<code>fastcgi_cache зона off;</code>
По умолчанию	<code>fastcgi_cache off;</code>
<i>Контекст</i>	<code>http, server, location</code>

Задаёт зону разделяемой памяти, используемую для кэширования. Одна и та же зона может использоваться в нескольких местах. В значении параметра можно использовать переменные. Параметр `off` запрещает кэширование, унаследованное с предыдущего уровня конфигурации.

fastcgi_cache_background_update

<i>Синтаксис</i>	<code>fastcgi_cache_background_update on off;</code>
По умолчанию	<code>fastcgi_cache_background_update off;</code>
<i>Контекст</i>	<code>http, server, location</code>

Позволяет запустить фоновый подзапрос для обновления просроченного элемента кэша, в то время как клиенту возвращается устаревший кэшированный ответ. Использование устаревшего кэшированного ответа в момент его обновления должно быть *разрешено*.

fastcgi_cache_bypass

<i>Синтаксис</i>	<code>fastcgi_cache_bypass строка ...;</code>
По умолчанию	<code>—</code>
<i>Контекст</i>	<code>http, server, location</code>

Задаёт условия, при которых ответ не будет браться из кэша. Если значение хотя бы одного из строковых параметров непустое и не равно "0", то ответ не берется из кэша:

```
fastcgi_cache_bypass $cookie_nocache $arg_nocache$arg_comment;
fastcgi_cache_bypass $http_pragma $http_authorization;
```

Можно использовать совместно с директивой *fastcgi_no_cache*.

fastcgi_cache_key

<i>Синтаксис</i>	<code>fastcgi_cache_key строка;</code>
По умолчанию	—
<i>Контекст</i>	http, server, location

Задаёт ключ для кэширования, например,

```
fastcgi_cache_key localhost:9000$request_uri;
```

fastcgi_cache_lock

<i>Синтаксис</i>	<code>fastcgi_cache_lock on off;</code>
По умолчанию	<code>fastcgi_cache_lock off;</code>
<i>Контекст</i>	http, server, location

Если включено, одновременно только одному запросу будет позволено заполнить новый элемент кэша, идентифицируемый согласно директиве *fastcgi_cache_key*, передав запрос на FastCGI-сервер. Остальные запросы этого же элемента будут либо ожидать появления ответа в кэше, либо освобождения блокировки этого элемента, в течение времени, заданного директивой *fastcgi_cache_lock_timeout*.

fastcgi_cache_lock_age

<i>Синтаксис</i>	<code>fastcgi_cache_lock_age время;</code>
По умолчанию	<code>fastcgi_cache_lock_age 5s;</code>
<i>Контекст</i>	http, server, location

Если последний запрос, переданный на FastCGI-сервер для заполнения нового элемента кэша, не завершился за указанное время, на FastCGI-сервер может быть передан ещё один запрос.

fastcgi_cache_lock_timeout

<i>Синтаксис</i>	<code>fastcgi_cache_lock_timeout время;</code>
По умолчанию	<code>fastcgi_cache_lock_timeout 5s;</code>
<i>Контекст</i>	http, server, location

Задаёт таймаут для *fastcgi_cache_lock*. По истечении указанного времени запрос будет передан на FastCGI-сервер, однако ответ не будет кэширован.

fastcgi_cache_max_range_offset

<i>Синтаксис</i>	<code>fastcgi_cache_max_range_offset число;</code>
По умолчанию	—
<i>Контекст</i>	http, server, location

Задаёт смещение в байтах для запросов с указанием диапазона запрашиваемых байт (byte-range requests). Если диапазон находится за указанным смещением, range-запрос будет передан на FastCGI-сервер и ответ не будет кэширован.

fastcgi_cache_methods

<i>Синтаксис</i>	<code>fastcgi_cache_methods GET HEAD POST ...;</code>
По умолчанию	<code>fastcgi_cache_methods GET HEAD;</code>
<i>Контекст</i>	http, server, location

Если метод запроса клиента указан в этой директиве, то ответ будет кэширован. Методы "GET" и "HEAD" всегда добавляются в список, но тем не менее рекомендуется перечислять их явно. См. также директиву *fastcgi_no_cache*.

fastcgi_cache_min_uses

<i>Синтаксис</i>	<code>fastcgi_cache_min_uses число;</code>
По умолчанию	<code>fastcgi_cache_min_uses 1;</code>
<i>Контекст</i>	http, server, location

Задаёт число запросов, после которого ответ будет кэширован.

Предупреждение

Метаданные кэша хранятся в разделяемой памяти. Ручное удаление файлов кэша не сбрасывает счетчики и может привести к непредсказуемому поведению. Для полного сброса остановите сервер, удалите директорию кэша и запустите снова.

Примечание

Сторонние модули очистки кэша (например, external-cache-purge) удаляют только файлы, но не сбрасывают счетчик *fastcgi_cache_min_uses*. Директива предназначена для защиты кэша от загрязнения редкими запросами, и сброс счетчика при очистке может негативно повлиять на производительность.

fastcgi_cache_path

<i>Синтаксис</i>	<code>fastcgi_cache_path <i>путь</i> [levels=<i>уровни</i>] [use_temp_path=<i>on</i> <i>off</i>] [keys_zone=<i>имя:размер</i>] [inactive=<i>время</i>] [max_size=<i>размер</i>] [min_free=<i>размер</i>] [manager_files=<i>число</i>] [manager_sleep=<i>время</i>] [manager_threshold=<i>время</i>] [loader_files=<i>число</i>] [loader_sleep=<i>время</i>] [loader_threshold=<i>время</i>];</code>
По умолчанию	—
<i>Контекст</i>	http, server, location

Задаёт путь и другие параметры кэша. Данные кэша хранятся в файлах. Ключом и именем файла в кэше является результат функции MD5 от проксированного URL.

Параметр `levels` задаёт уровни иерархии кэша: можно задать от 1 до 3 уровней, на каждом уровне допускаются значения 1 или 2. Например, при использовании

```
fastcgi_cache_path /data/angie/cache levels=1:2 keys_zone=one:10m;
```

имена файлов в кэше будут такого вида:

```
/data/angie/cache/c/29/b7f54b2df7773722d382f4809d65029c
```

Кэшируемый ответ сначала записывается во временный файл, а потом этот файл переименовывается. Временные файлы и кэш могут располагаться на разных файловых системах. Однако нужно учитывать, что в этом случае вместо дешевой операции переименовывания в пределах одной файловой системы файл копируется с одной файловой системы на другую. Поэтому лучше, если кэш будет находиться на той же файловой системе, что и каталог с временными файлами.

Какой из каталогов будет использоваться для временных файлов определяется параметром `use_temp_path`.

on	Если параметр не задан или установлен в значение "on", будет использоваться каталог, задаваемый директивой <code>fastcgi_temp_path</code> для данного <code>location</code> .
off	временные файлы будут располагаться непосредственно в каталоге кэша.

Кроме того, все активные ключи и информация о данных хранятся в зоне разделяемой памяти, имя и размер которой задаются параметром `keys_zone`. Зоны размером в 1 мегабайт достаточно для хранения около 8 тысяч ключей. Метаданные кэша хранятся в разделяемой памяти.

Если к данным кэша не обращаются в течение времени, заданного параметром `inactive`, то данные удаляются, независимо от их свежести.

По умолчанию `inactive` равен 10 минутам.

Специальный процесс **менеджера кэша** следит за максимальным размером кэша, а также за минимальным объемом свободного места на файловой системе с кэшем, и удаляет наименее востребованные данные при превышении максимального размера кэша или недостаточном объеме свободного места. Удаление данных происходит итерациями.

max_size	максимальное пороговое значение размера кэша
min_free	минимальное пороговое значение объема свободного места на файловой системе с кэшем
manager_files	максимальное количество удаляемых элементов кэша за одну итерацию По умолчанию: 100
manager_threshold	ограничивает время работы одной итерации По умолчанию: 200 миллисекунд
manager_sleep	время, в течение которого выдерживается пауза между итерациями По умолчанию: 50 миллисекунд

Через минуту после старта Angie активируется специальный процесс **загрузчика кэша**, который загружает в зону кэша информацию о ранее кэшированных данных, хранящихся на файловой системе. Загрузка также происходит итерациями.

<code>loader_files</code>	максимальное количество элементов кэша к загрузке в одну итерацию По умолчанию: 100
<code>loader_threshold</code>	ограничивает время работы одной итерации По умолчанию: 200 миллисекунд
<code>loader_sleep</code>	время, в течение которого выдерживается пауза между итерациями По умолчанию: 50 миллисекунд

`fastcgi_cache_revalidate`

<i>Синтаксис</i>	<code>fastcgi_cache_revalidate on off;</code>
По умолчанию	<code>fastcgi_cache_revalidate off;</code>
<i>Контекст</i>	<code>http, server, location</code>

Разрешает ревалидацию просроченных элементов кэша при помощи условных запросов с полями заголовка `If-Modified-Since` и `If-None-Match`.

`fastcgi_cache_use_stale`

<i>Синтаксис</i>	<code>fastcgi_cache_use_stale error timeout invalid_header updating http_500 http_503 http_403 http_429 off ...;</code>
По умолчанию	<code>fastcgi_cache_use_stale off;</code>
<i>Контекст</i>	<code>http, server, location</code>

Определяет, в каких случаях можно использовать устаревший кэшированный ответ. Параметры директивы совпадают с параметрами директивы `fastcgi_next_upstream`.

<code>error</code>	позволяет использовать устаревший кэшированный ответ при невозможности выбора FastCGI-сервера для обработки запроса.
<code>updating</code>	дополнительный параметр, разрешает использовать устаревший кэшированный ответ, если на данный момент он уже обновляется. Это позволяет минимизировать число обращений к FastCGI-серверам при обновлении кэшированных данных.

Использование устаревшего кэшированного ответа может также быть разрешено непосредственно в заголовке ответа на определенное количество секунд после того, как ответ устарел.

- Расширение `stale-while-revalidate` поля заголовка `Cache-Control` разрешает использовать устаревший кэшированный ответ, если на данный момент он уже обновляется.
- Расширение `stale-if-error` поля заголовка `Cache-Control` разрешает использовать устаревший кэшированный ответ в случае ошибки.

Примечание

Такой способ менее приоритетен, чем задание параметров директивы.

Чтобы минимизировать число обращений к FastCGI-серверам при заполнении нового элемента кэша, можно воспользоваться директивой *fastcgi_cache_lock*.

fastcgi_cache_valid

Синтаксис	<code>fastcgi_cache_valid [код ...] время;</code>
По умолчанию	—
Контекст	http, server, location

Задаёт время кэширования для разных кодов ответа. Например, директивы

```
fastcgi_cache_valid 200 302 10m;
fastcgi_cache_valid 404 1m;
```

задают время кэширования 10 минут для ответов с кодами 200 и 302 и 1 минуту для ответов с кодом 404.

Если указано только время кэширования,

```
fastcgi_cache_valid 5m;
```

то кэшируются только ответы 200, 301 и 302.

Кроме того, можно кэшировать любые ответы с помощью параметра *any*:

```
fastcgi_cache_valid 200 302 10m;
fastcgi_cache_valid 301 1h;
fastcgi_cache_valid any 1m;
```

Примечание

Параметры кэширования могут также быть заданы непосредственно в заголовке ответа. Такой способ приоритетнее, чем задание времени кэширования с помощью директивы.

- Поле заголовка **X-Accel-Expires** задаёт время кэширования ответа в секундах. Значение 0 запрещает кэшировать ответ. Если значение начинается с префикса @, оно задаёт абсолютное время в секундах с начала эпохи, до которого ответ может быть кэширован.
- Если в заголовке нет поля **X-Accel-Expires**, параметры кэширования определяются по полям заголовка **Expires** или **Cache-Control**.
- Ответ, в заголовке которого есть поле **Set-Cookie**, не будет кэшироваться.
- Ответ, в заголовке которого есть поле **Vary** со специальным значением "*", не будет кэшироваться. Ответ, в заголовке которого есть поле **Vary** с другим значением, будет кэширован с учетом соответствующих полей заголовка запроса.

Обработка одного или более из этих полей заголовка может быть отключена при помощи директивы *fastcgi_ignore_headers*.

fastcgi_catch_stderr

<i>Синтаксис</i>	<code>fastcgi_catch_stderr строка;</code>
По умолчанию	—
<i>Контекст</i>	http, server, location

Задаёт строку для поиска в потоке ошибок ответа, полученного от FastCGI-сервера. Если строка найдена, то считается, что FastCGI-сервер вернул *неверный* ответ. Это позволяет обрабатывать ошибки приложений в Angie, например:

```
location /php/ {
    fastcgi_pass backend:9000;
    ...
    fastcgi_catch_stderr "PHP Fatal error";
    fastcgi_next_upstream error timeout invalid_header;
}
```

fastcgi_connect_timeout

<i>Синтаксис</i>	<code>fastcgi_connect_timeout время;</code>
По умолчанию	<code>fastcgi_connect_timeout 60s;</code>
<i>Контекст</i>	http, server, location

Задаёт таймаут для установления соединения с FastCGI-сервером. Необходимо иметь в виду, что этот таймаут обычно не может превышать 75 секунд.

fastcgi_connection_drop

<i>Синтаксис</i>	<code>fastcgi_connection_drop время on off;</code>
По умолчанию	<code>fastcgi_connection_drop off;</code>
<i>Контекст</i>	http, server, location

Настраивает завершение всех соединений с проксируемым сервером, если он был удален из группы или помечен как постоянно недоступный в результате процесса *resolve* или команды *API DELETE*.

Соединение завершается, когда обрабатывается следующее событие чтения или записи для клиента или проксируемого сервера.

Установка *времени* включает *таймаут* до завершения соединения; при выборе значения *on* соединения завершаются немедленно.

fastcgi_force_ranges

<i>Синтаксис</i>	<code>fastcgi_force_ranges on off;</code>
По умолчанию	<code>fastcgi_force_ranges off;</code>
<i>Контекст</i>	http, server, location

Включает поддержку диапазонов запрашиваемых байт (byte-range) для кэшированных и некешированных ответов FastCGI-сервера вне зависимости от наличия поля **Accept-Ranges** в заголовках этих ответов.

fastcgi_hide_header

<i>Синтаксис</i>	<code>fastcgi_hide_header поле;</code>
По умолчанию	—
<i>Контекст</i>	http, server, location

По умолчанию Angie не передает клиенту поля заголовка **Status** и **X-Accel-...** из ответа FastCGI-сервера. Директива *fastcgi_hide_header* задает дополнительные поля, которые не будут передаваться. Если же передачу полей нужно разрешить, можно воспользоваться директивой *fastcgi_pass_header*.

fastcgi_ignore_client_abort

<i>Синтаксис</i>	<code>fastcgi_ignore_client_abort on off;</code>
По умолчанию	<code>fastcgi_ignore_client_abort off;</code>
<i>Контекст</i>	http, server, location

Определяет, закрывать ли соединение с FastCGI-сервером в случае, если клиент закрыл соединение, не дождавшись ответа.

fastcgi_ignore_headers

<i>Синтаксис</i>	<code>fastcgi_ignore_headers поле;</code>
По умолчанию	—
<i>Контекст</i>	http, server, location

Запрещает обработку некоторых полей заголовка из ответа FastCGI-сервера. В директиве можно указать поля **X-Accel-Redirect**, **X-Accel-Expires**, **X-Accel-Limit-Rate**, **X-Accel-Buffering**, **X-Accel-Charset**, **Expires**, **Cache-Control**, **Set-Cookie** и **Vary**.

Если не запрещено, обработка этих полей заголовка заключается в следующем:

- **X-Accel-Expires**, **Expires**, **Cache-Control**, **Set-Cookie** и **Vary** задают *параметры кэширования* ответа;
- **X-Accel-Redirect** производит *внутреннее перенаправление* на указанный URI;

- X-Accel-Limit-Rate задает *ограничение скорости* передачи ответа клиенту;
- X-Accel-Buffering включает или выключает *буферизацию* ответа;
- X-Accel-Charset задает желаемую *кодировку* ответа.

fastcgi_index

<i>Синтаксис</i>	fastcgi_index <i>имя</i> ;
По умолчанию	—
<i>Контекст</i>	http, server, location

Задаёт имя файла, который при создании переменной `$fastcgi_script_name` будет добавляться после URI, если URI заканчивается косой чертой. Например, при таких настройках

```
fastcgi_index index.php;
fastcgi_param SCRIPT_FILENAME /home/www/scripts/php$fastcgi_script_name;
```

и запросе `/page.php` параметр `SCRIPT_FILENAME` будет равен `/home/www/scripts/php/page.php`,

а при запросе `/` - `/home/www/scripts/php/index.php`.

fastcgi_intercept_errors

<i>Синтаксис</i>	fastcgi_intercept_errors on off;
По умолчанию	fastcgi_intercept_errors off;
<i>Контекст</i>	http, server, location

Определяет, передавать ли клиенту ответы FastCGI-сервера с кодом больше либо равным 300, или же перехватывать их и перенаправлять на обработку Angie с помощью директивы `error_page`.

fastcgi_keep_conn

<i>Синтаксис</i>	fastcgi_keep_conn on off;
По умолчанию	fastcgi_keep_conn off;
<i>Контекст</i>	http, server, location

По умолчанию FastCGI-сервер будет закрывать соединение сразу же после отправки ответа. При установке значения `on` Angie указывает FastCGI-серверу оставлять соединения открытыми. Это в частности требуется для функционирования *постоянных соединений* с FastCGI-серверами.

fastcgi_limit_rate

<i>Синтаксис</i>	fastcgi_limit_rate <i>скорость</i> ;
По умолчанию	fastcgi_limit_rate 0;
<i>Контекст</i>	http, server, location

Ограничивает скорость чтения ответа от проксируемого сервера. *Скорость* задается в байтах в секунду; можно использовать переменные.

0	отключает ограничение скорости

Примечание

Ограничение устанавливается на запрос, поэтому, если Angie одновременно откроет два соединения к FastCGI-серверу, суммарная скорость будет вдвое выше заданного ограничения. Ограничение работает только в случае, если включена *буферизация* ответов FastCGI-сервера.

fastcgi_max_temp_file_size

<i>Синтаксис</i>	fastcgi_max_temp_file_size <i>размер</i> ;
По умолчанию	fastcgi_max_temp_file_size 1024m;
<i>Контекст</i>	http, server, location

Если включена *буферизация* ответов FastCGI-сервера, и ответ не помещается целиком в буферы, заданные директивами *fastcgi_buffer_size* и *fastcgi_buffers*, часть ответа может быть записана во временный файл. Эта директива задает максимальный размер временного файла. Размер данных, сбрасываемых во временный файл за один раз, задается директивой *fastcgi_temp_file_write_size*.

0	отключает возможность буферизации ответов во временные файлы

Примечание

Данное ограничение не распространяется на ответы, которые будут *кэшированы* или сохранены на диске.

fastcgi_next_upstream

<i>Синтаксис</i>	fastcgi_next_upstream error timeout invalid_header http_500 http_503 http_403 http_404 http_429 non_idempotent off ...;
По умолчанию	fastcgi_next_upstream error timeout;
<i>Контекст</i>	http, server, location

Определяет, в каких случаях запрос будет передан следующему серверу:

error	произошла ошибка соединения с сервером, передачи ему запроса или чтения заголовка ответа сервера;
timeout	произошел таймаут во время соединения с сервером, передачи ему запроса или чтения заголовка ответа сервера;
invalid_header	сервер вернул пустой или неверный ответ;
http_500	сервер вернул ответ с кодом 500;
http_503	сервер вернул ответ с кодом 503;
http_403	сервер вернул ответ с кодом 403;
http_404	сервер вернул ответ с кодом 404;
http_429	сервер вернул ответ с кодом 429;
non_idempotent	обычно запросы с неидемпотентным методом (<i>POST</i> , <i>LOCK</i> , <i>PATCH</i>) не передаются на другой сервер, если запрос серверу группы уже был отправлен; включение параметра явно разрешает повторять подобные запросы;
off	запрещает передачу запроса следующему серверу.

Примечание

Необходимо понимать, что передача запроса следующему серверу возможна только при условии, что клиенту еще ничего не передавалось. То есть, если ошибка или таймаут возникли в середине передачи ответа, то исправить это уже невозможно.

Директива также определяет, что считается *неудачной попыткой* работы с сервером.

error	всегда считаются неудачными попытками, даже если они не указаны в директиве
timeout	
invalid_header	
http_500	считаются неудачными попытками, только если они указаны в директиве
http_503	
http_429	
http_403	никогда не считаются неудачными попытками
http_404	

Передача запроса следующему серверу может быть ограничена по *количеству попыток* и по *времени*.

fastcgi_next_upstream_timeout

<i>Синтаксис</i>	<code>fastcgi_next_upstream_timeout время;</code>
По умолчанию	<code>fastcgi_next_upstream_timeout 0;</code>
<i>Контекст</i>	http, server, location

Ограничивает время, в течение которого возможна передача запроса *следующему серверу*.

0	отключает это ограничение

fastcgi_next_upstream_tries

<i>Синтаксис</i>	<code>fastcgi_next_upstream_tries число;</code>
По умолчанию	<code>fastcgi_next_upstream_tries 0;</code>
<i>Контекст</i>	http, server, location

Ограничивает число допустимых попыток для передачи запроса *следующему серверу*.

0	отключает это ограничение

fastcgi_no_cache

<i>Синтаксис</i>	<code>fastcgi_no_cache строка ...;</code>
По умолчанию	—
<i>Контекст</i>	http, server, location

Задаёт условия, при которых ответ не будет сохраняться в кэш. Если значение хотя бы одного из строковых параметров непустое и не равно "0", то ответ не будет сохранен:

```
fastcgi_no_cache $cookie_nocache $arg_nocache$arg_comment;
fastcgi_no_cache $http_pragma $http_authorization;
```

Можно использовать совместно с директивой *fastcgi_cache_bypass*.

fastcgi_param

<i>Синтаксис</i>	<code>fastcgi_param параметр значение [if_not_empty];</code>
По умолчанию	—
<i>Контекст</i>	http, server, location

Задаёт параметр, который будет передаваться FastCGI-серверу. В качестве значения можно использовать текст, переменные и их комбинации. Директивы наследуются с предыдущего уровня конфигурации при условии, что на данном уровне не описаны свои директивы *fastcgi_param*.

Ниже приведен пример минимально необходимых параметров для PHP:

```
fastcgi_param SCRIPT_FILENAME /home/www/scripts/php$fastcgi_script_name;
fastcgi_param QUERY_STRING $query_string;
```

Параметр SCRIPT_FILENAME используется в PHP для определения имени скрипта, а в параметре QUERY_STRING передаются параметры запроса.

Если скрипты обрабатывают запросы POST, то нужны еще три параметра:

```
fastcgi_param REQUEST_METHOD $request_method;
fastcgi_param CONTENT_TYPE $content_type;
fastcgi_param CONTENT_LENGTH $content_length;
```

Если PHP был собран с параметром конфигурации `--enable-force-cgi-redirect`, то нужно передавать параметр `REDIRECT_STATUS` со значением "200":

```
fastcgi_param REDIRECT_STATUS 200;
```

Если директива указана с `if_not_empty`, такой параметр с пустым значением передаваться на сервер не будет:

```
fastcgi_param HTTPS $https if_not_empty;
```

fastcgi_pass

<i>Синтаксис</i>	<code>fastcgi_pass адрес;</code>
По умолчанию	—
<i>Контекст</i>	location, if в location

Задаёт адрес FastCGI-сервера. Адрес может быть указан в виде доменного имени или IP-адреса, и порта:

```
fastcgi_pass localhost:9000;
```

или в виде пути UNIX-сокета:

```
fastcgi_pass unix:/tmp/fastcgi.socket;
```

Если доменному имени соответствует несколько адресов, то все они будут использоваться по очереди (round-robin). И, кроме того, адрес может быть *группой серверов*.

В значении параметра можно использовать переменные. В этом случае, если адрес указан в виде доменного имени, имя ищется среди описанных *групп серверов* и если не найдено, то определяется с помощью *resolver*'а.

Примечание

Если `fastcgi_pass` стоит в `location` с косой чертой в конце префикса (например, `location /name/`), и при этом в директиве `auto_redirect` указано `default`, запросы без косой черты в конце будут перенаправляться (`/name -> /name/`).

fastcgi_pass_header

<i>Синтаксис</i>	<code>fastcgi_pass_header поле;</code>
По умолчанию	—
<i>Контекст</i>	http, server, location

Разрешает передавать от FastCGI-сервера клиенту *запрещенные для передачи* поля заголовка.

fastcgi_pass_request_body

<i>Синтаксис</i>	fastcgi_pass_request_body on off;
По умолчанию	—
<i>Контекст</i>	http, server, location

Позволяет запретить передачу исходного тела запроса на FastCGI-сервер. См. также директиву *fastcgi_pass_request_headers*.

fastcgi_pass_request_headers

<i>Синтаксис</i>	fastcgi_pass_request_headers on off;
По умолчанию	—
<i>Контекст</i>	http, server, location

Позволяет запретить передачу полей заголовка исходного запроса на FastCGI-сервер. См. также директиву *fastcgi_pass_request_body*.

fastcgi_read_timeout

<i>Синтаксис</i>	fastcgi_read_timeout <i>время</i> ;
По умолчанию	fastcgi_read_timeout 60s;
<i>Контекст</i>	http, server, location

Задаёт таймаут при чтении ответа FastCGI-сервера. Таймаут устанавливается не на всю передачу ответа, а только между двумя операциями чтения. Если по истечении этого времени FastCGI-сервер ничего не передаст, соединение закрывается.

fastcgi_request_buffering

<i>Синтаксис</i>	fastcgi_request_buffering on off;
По умолчанию	fastcgi_request_buffering on;
<i>Контекст</i>	http, server, location

Разрешает или запрещает использовать буферизацию тела запроса клиента.

on	тело запроса полностью <i>читается</i> от клиента перед отправкой запроса на FastCGI-сервер.
off	тело запроса отправляется на FastCGI-сервер сразу же по мере его поступления. В этом случае запрос не может быть передан <i>следующему серверу</i> , если Angie уже начал отправку тела запроса.

fastcgi_send_lowat

<i>Синтаксис</i>	<code>fastcgi_send_lowat размер;</code>
По умолчанию	<code>fastcgi_send_lowat 0;</code>
<i>Контекст</i>	http, server, location

При установке директивы в ненулевое значение Angie будет пытаться минимизировать число операций отправки на исходящих соединениях с FastCGI-сервером либо при помощи флага `NOTE_LOWAT` метода *кqueue*, либо при помощи параметра сокета `SO_SNDLOWAT`, с указанным размером.

Примечание

Эта директива игнорируется на Linux, Solaris и Windows.

fastcgi_send_timeout

<i>Синтаксис</i>	<code>fastcgi_send_timeout время;</code>
По умолчанию	<code>fastcgi_send_timeout 60s;</code>
<i>Контекст</i>	http, server, location

Задаёт таймаут при передаче запроса FastCGI-серверу. Таймаут устанавливается не на всю передачу запроса, а только между двумя операциями записи. Если по истечении этого времени FastCGI-сервер не примет новых данных, соединение закрывается.

fastcgi_socket_keepalive

<i>Синтаксис</i>	<code>fastcgi_socket_keepalive on off;</code>
По умолчанию	<code>fastcgi_socket_keepalive off;</code>
<i>Контекст</i>	http, server, location

Конфигурирует поведение "TCP keepalive" для исходящих соединений к FastCGI-серверу.

<code>""</code>	По умолчанию для сокета действуют настройки операционной системы.
<code>on</code>	для сокета включается параметр <code>SO_KEEPALIVE</code>

fastcgi_split_path_info

<i>Синтаксис</i>	<code>fastcgi_split_path_info regex;</code>
По умолчанию	—
<i>Контекст</i>	location

Задаёт регулярное выражение, выделяющее значение для переменной `$fastcgi_path_info`. Регулярное выражение должно иметь две группы захвата, из которых первая становится значением переменной `$fastcgi_script_name`, а вторая — значением переменной `$fastcgi_path_info`. Например, при таких настройках

```
location ~ ^(\.+\.php)(.*)$ {
    fastcgi_split_path_info      ^(\.+\.php)(.*)$;
    fastcgi_param SCRIPT_FILENAME /path/to/php$fastcgi_script_name;
    fastcgi_param PATH_INFO      $fastcgi_path_info;
```

и запросе `/show.php/article/0001` параметр `SCRIPT_FILENAME` будет равен `/path/to/php/show.php`,

а параметр `PATH_INFO` - `/article/0001`.

fastcgi_store

<i>Синтаксис</i>	<code>fastcgi_store on off строка;</code>
По умолчанию	<code>fastcgi_store off;</code>
<i>Контекст</i>	<code>http, server, location</code>

Разрешает сохранение на диск файлов.

<code>on</code>	сохраняет файлы в соответствии с путями, указанными в директивах <i>alias</i> или <i>root</i>
<code>off</code>	запрещает сохранение файлов

Кроме того, имя файла можно задать явно с помощью строки с переменными:

```
fastcgi_store /data/www$original_uri;
```

Время изменения файлов выставляется согласно полученному полю `Last-Modified` в заголовке ответа. Ответ сначала записывается во временный файл, а потом этот файл переименовывается. Временный файл и постоянное место хранения ответа могут располагаться на разных файловых системах. Однако нужно учитывать, что в этом случае вместо дешевой операции переименовывания в пределах одной файловой системы файл копируется с одной файловой системы на другую. Поэтому лучше, если сохраняемые файлы будут находиться на той же файловой системе, что и каталог с временными файлами, задаваемый директивой `fastcgi_temp_path` для данного `location`.

Директиву можно использовать для создания локальных копий статических неизменяемых файлов, например:

```
location /images/ {
    root          /data/www;
    error_page    404 = /fetch$uri;
}

location /fetch/ {
    internal;

    fastcgi_pass    backend:9000;
    ...

    fastcgi_store    on;
    fastcgi_store_access user:rw group:rw all:r;
```



```
fastcgi_temp_path    /data/temp;

alias                /data/www/;
}
```

fastcgi_store_access

<i>Синтаксис</i>	<code>fastcgi_store_access</code> пользователи:права ...;
По умолчанию	<code>fastcgi_store_access user:rw;</code>
<i>Контекст</i>	http, server, location

Задаёт права доступа для создаваемых файлов и каталогов, например,

```
fastcgi_store_access user:rw group:rw all:r;
```

Если заданы какие-либо права для group или all, то права для user указывать необязательно:

```
fastcgi_store_access group:rw all:r;
```

fastcgi_temp_file_write_size

<i>Синтаксис</i>	<code>fastcgi_temp_file_write_size</code> размер;
По умолчанию	<code>fastcgi_temp_file_write_size 8k 16k;</code>
<i>Контекст</i>	http, server, location

Ограничивает размер данных, сбрасываемых во временный файл за один раз, при включенной буферизации ответов FastCGI-сервера во временные файлы. По умолчанию размер ограничен двумя буферами, заданными директивами `fastcgi_buffer_size` и `fastcgi_buffers`. Максимальный размер временного файла задается директивой `fastcgi_max_temp_file_size`.

fastcgi_temp_path

<i>Синтаксис</i>	<code>fastcgi_temp_path</code> путь [уровень1 [уровень2 [уровень3]]]`;
По умолчанию	<code>fastcgi_temp_path fastcgi_temp;</code> (путь зависит от параметра сборки <code>--http-fastcgi-temp-path</code>)
<i>Контекст</i>	http, server, location

Задаёт имя каталога для хранения временных файлов с данными, полученными от FastCGI-серверов. В каталоге может использоваться иерархия подкаталогов до трех уровней. Например, при такой конфигурации

```
fastcgi_temp_path /spool/angie/fastcgi_temp 1 2;
```

временный файл будет следующего вида:

```
/spool/angie/fastcgi_temp/7/45/00000123457
```

См. также параметр `use_temp_path` директивы `fastcgi_cache_path`.

Параметры, передаваемые FastCGI-серверу

Поля заголовка HTTP-запроса передаются FastCGI-серверу в виде параметров. В приложениях и скриптах, запущенных в виде FastCGI-сервера, эти параметры обычно доступны в виде переменных среды. Например, поле заголовка **User-Agent** передается как параметр `HTTP_USER_AGENT`. Кроме полей заголовка HTTP-запроса можно передавать произвольные параметры с помощью директивы *fastcgi_param*.

Встроенные переменные

В модуле *http_fastcgi* есть встроенные переменные, которые можно использовать для формирования параметров с помощью директивы *fastcgi_param*:

`$fastcgi_script_name`

URI запроса или же, если URI заканчивается косой чертой, то URI запроса, дополненное именем индексного файла, задаваемого директивой *fastcgi_index*. Эту переменную можно использовать для задания параметров `SCRIPT_FILENAME` и `PATH_TRANSLATED`, используемых, в частности, для определения имени скрипта в PHP. Например, для запроса `/info/` и при использовании директив

```
fastcgi_index index.php;
fastcgi_param SCRIPT_FILENAME /home/www/scripts/php$fastcgi_script_name;
```

параметр `SCRIPT_FILENAME` будет равен `/home/www/scripts/php/info/index.php`.

При использовании директивы *fastcgi_split_path_info* переменная *\$fastcgi_script_name* равна значению первой группы захвата, задаваемой этой директивой.

`$fastcgi_path_info`

значение второй группы захвата, задаваемой директивой *fastcgi_split_path_info*. Эту переменную можно использовать для задания параметра `PATH_INFO`.

FLV

Обеспечивает серверную поддержку псевдо-стриминга для файлов Flash Video (FLV).

Он специальным образом обрабатывает запросы с аргументом **start** в строке запроса, посылая в ответ содержимое файла с запрошенного смещения в байтах, добавив перед ним FLV-заголовок.

При сборке из исходного кода модуль не собирается по умолчанию; его необходимо включить с помощью параметра сборки `--with-http_flv_module`. В пакетах и образах из наших репозиториях модуль включен в сборку.

Пример конфигурации

```
location ~ /\.flv$ {
    flv;
}
```

Директивы

flv

<i>Синтаксис</i>	flv;
По умолчанию	—
<i>Контекст</i>	location

Включает в содержащем location обработку этим модулем.

Geo

Создает переменные, значения которых зависят от IP-адреса клиента.

Пример конфигурации

```
geo $geo {
    default 0;

    127.0.0.1 2;
    192.168.1.0/24 1;
    10.1.0.0/16 1;

    ::1 2;
    2001:0db8::/32 1;
}
```

Директивы

geo

<i>Синтаксис</i>	geo [\$адрес] \$переменная { ... }
По умолчанию	—
<i>Контекст</i>	http

Описывает для указанной переменной зависимость значения от IP-адреса клиента. По умолчанию адрес берется из переменной *\$remote_addr*, но его также можно получить из другой переменной, например:

```
geo $arg_remote_addr $geo {
    ...;
}
```

Примечание

Поскольку переменные вычисляются только в момент использования, само по себе наличие даже большого числа объявлений переменных `geo` не влечет за собой никаких дополнительных расходов на обработку запросов.

Если значение переменной не представляет собой правильный IP-адрес, то используется адрес "255.255.255.255".

Адреса задаются либо префиксами в формате CIDR (включая одиночные адреса), либо в виде диапазонов.

Также поддерживаются следующие специальные параметры:

<code>delete</code>	удаляет описанную сеть
<code>default</code>	значение переменной, если адрес клиента не соответствует ни одному из заданных адресов. При задании адресов в формате CIDR вместо <code>default</code> можно использовать " <code>0.0.0.0/0</code> " и " <code>::/0</code> ". Если параметр <code>default</code> не указан, значением по умолчанию будет пустая строка.
<code>include</code>	включает файл с адресами и значениями. Включений может быть несколько.
<code>proxy</code>	задает доверенные адреса, при запросе с которых будет использоваться адрес в переданном поле заголовка запроса <code>X-Forwarded-For</code> . В отличие от обычных адресов, доверенные адреса проверяются последовательно.
<code>proxy_recursive</code>	включает рекурсивный поиск адреса. При выключенном рекурсивном поиске вместо исходного адреса клиента, совпадающего с одним из доверенных адресов, будет использоваться последний адрес, переданный в <code>X-Forwarded-For</code> . При включенном рекурсивном поиске вместо исходного адреса клиента, совпадающего с одним из доверенных адресов, будет использоваться последний не доверенный адрес, переданный в <code>X-Forwarded-For</code> .
<code>ranges</code>	указывает, что адреса задаются в виде диапазонов. Этот параметр должен быть первым. Для ускорения загрузки гео-базы нужно располагать адреса в порядке возрастания.

Пример:

```
geo $country {
    default      ZZ;
    include      conf/geo.conf;
    delete      127.0.0.0/16;
    proxy        192.168.100.0/24;
    proxy        2001:0db8::/32;

    127.0.0.0/24 US;
    127.0.0.1/32 RU;
    10.1.0.0/16  RU;
    192.168.1.0/24 UK;
}
```

В файле `conf/geo.conf` могут быть такие строки:

```
10.2.0.0/16    RU;
192.168.2.0/24 RU;
```

В качестве значения выбирается максимальное совпадение, например, для адреса 127.0.0.1 будет выбрано значение RU, а не US.

Пример описания диапазонов:

```
geo $country {
    ranges;
    default                ZZ;
    127.0.0.0-127.0.0.0    US;
    127.0.0.1-127.0.0.1    RU;
    127.0.0.2-127.0.0.255  US;
    10.1.0.0-10.1.255.255  RU;
    192.168.1.0-192.168.1.255 UK;
}
```

gRPC

Позволяет передавать запросы gRPC-серверу.

Примечание

Для работы этого модуля необходим модуль *HTTP2*.

Пример конфигурации

```
server {
    listen 9000;

    http2 on;

    location / {
        grpc_pass 127.0.0.1:9000;
    }
}
```

Директивы

grpc_bind

<i>Синтаксис</i>	<code>grpc_bind адрес [transparent] off;</code>
По умолчанию	—
<i>Контекст</i>	http, server, location

Задаёт локальный IP-адрес с необязательным портом, который будет использоваться в исходящих соединениях с gRPC-сервером. В значении параметра допустимо использование переменных. Специальное значение `off` отменяет действие унаследованной с предыдущего уровня конфигурации директивы `grpc_bind`, позволяя системе самостоятельно выбирать локальный IP-адрес и порт.

Параметр `transparent` позволяет задать нелокальный IP-адрес, который будет использоваться в исходящих соединениях с gRPC-сервером, например, реальный IP-адрес клиента:

```
grpc_bind $remote_addr transparent;
```

Для работы параметра обычно требуется запустить рабочие процессы Angie с привилегиями *суперпользователя*. В Linux это не требуется, так как если указан параметр *transparent*, то рабочие процессы наследуют *capability CAP_NET_RAW* из главного процесса.

Примечание

Необходимо настроить таблицу маршрутизации ядра для перехвата сетевого трафика с gRPC-сервера.

grpc_buffer_size

<i>Синтаксис</i>	grpc_buffer_size <i>размер</i> ;
По умолчанию	grpc_buffer_size 4k 8k;
<i>Контекст</i>	http, server, location

Задаёт размер буфера, в который будет читаться первая часть ответа, получаемого от gRPC-сервера. Ответ синхронно передается клиенту сразу же по мере его поступления.

grpc_connect_timeout

<i>Синтаксис</i>	grpc_connect_timeout <i>время</i> ;
По умолчанию	grpc_connect_timeout 60s;
<i>Контекст</i>	http, server, location

Задаёт таймаут для установления соединения с gRPC-сервером. Необходимо иметь в виду, что этот таймаут обычно не может превышать 75 секунд.

grpc_connection_drop

<i>Синтаксис</i>	grpc_connection_drop <i>время</i> on off;
По умолчанию	grpc_connection_drop off;
<i>Контекст</i>	http, server, location

Настраивает завершение всех соединений с проксируемым сервером, если он был удален из группы или помечен как постоянно недоступный в результате процесса *resolve* или команды *API DELETE*.

Соединение завершается, когда обрабатывается следующее событие чтения или записи для клиента или проксируемого сервера.

Установка *времени* включает *таймаут* до завершения соединения; при выборе значения *on* соединения завершаются немедленно.

grpc_hide_header

<i>Синтаксис</i>	grpc_hide_header поле;
По умолчанию	—
<i>Контекст</i>	http, server, location

По умолчанию Angie не передает клиенту поля заголовка **Date**, **Server** и **X-Accel-...** из ответа gRPC-сервера. Директива **grpc_hide_header** задает дополнительные поля, которые не будут передаваться. Если же передачу полей нужно разрешить, можно воспользоваться директивой *grpc_pass_header*.

grpc_ignore_headers

<i>Синтаксис</i>	grpc_ignore_headers поле ...;
По умолчанию	—
<i>Контекст</i>	http, server, location

Запрещает обработку некоторых полей заголовка из ответа gRPC-сервера. В директиве можно указать поля **X-Accel-Redirect** и **X-Accel-Charset**.

Если не запрещено, обработка этих полей заголовка заключается в следующем:

- **X-Accel-Redirect** производит *внутреннее перенаправление* на указанный URI;
- **X-Accel-Charset** задает желаемую *кодировку* ответа.

grpc_intercept_errors

<i>Синтаксис</i>	grpc_intercept_errors on off;
По умолчанию	grpc_intercept_errors off;
<i>Контекст</i>	http, server, location

Определяет, передавать ли клиенту ответы gRPC-сервера с кодом больше либо равным 300, или же перехватывать их и перенаправлять на обработку Angie с помощью директивы *error_page*.

grpc_next_upstream

<i>Синтаксис</i>	grpc_next_upstream error timeout invalid_header http_500 http_502 http_503 http_504 http_403 http_404 http_429 non_idempotent off ...;
По умолчанию	grpc_next_upstream error timeout;
<i>Контекст</i>	http, server, location

Определяет, в каких случаях запрос будет передан следующему в группе *upstream* серверу:

error	произошла ошибка соединения с сервером, передачи ему запроса или чтения заголовка ответа сервера;
timeout	произошел таймаут во время соединения с сервером, передачи ему запроса или чтения заголовка ответа сервера;
invalid_header	сервер вернул пустой или неверный ответ;
http_500	сервер вернул ответ с кодом 500;
http_502	сервер вернул ответ с кодом 502;
http_503	сервер вернул ответ с кодом 503;
http_504	сервер вернул ответ с кодом 504;
http_403	сервер вернул ответ с кодом 403;
http_404	сервер вернул ответ с кодом 404;
http_429	сервер вернул ответ с кодом 429;
non_idempotent	обычно запросы с неидемпотентным методом (<i>POST</i> , <i>LOCK</i> , <i>PATCH</i>) не передаются на другой сервер, если запрос серверу группы уже был отправлен; включение параметра явно разрешает повторять подобные запросы;
off	запрещает передачу запроса следующему серверу.

i Примечание

Необходимо понимать, что передача запроса следующему серверу возможна только при условии, что клиенту еще ничего не передавалось. То есть, если ошибка или таймаут возникли в середине передачи ответа клиенту, то действие директивы на такой запрос не распространяется.

Директива также определяет, что считается *неудачной попыткой* работы с сервером.

error, timeout, invalid_header	всегда считаются неудачными попытками, даже если они не указаны в директиве
http_500, http_502, http_503, http_504, http_429	считаются неудачными попытками, только если они указаны в директиве
http_403, http_404	никогда не считаются неудачными попытками

Передача запроса следующему серверу может быть ограничена по *количеству попыток* и по *времени*.

grpc_next_upstream_timeout

<i>Синтаксис</i>	grpc_next_upstream_timeout <i>время</i> ;
По умолчанию	grpc_next_upstream_timeout 0;
<i>Контекст</i>	http, server, location

Ограничивает время, в течение которого возможна передача запроса *следующему* серверу.

0	отключает это ограничение

grpc_next_upstream_tries

<i>Синтаксис</i>	grpc_next_upstream_tries <i>число</i> ;
По умолчанию	grpc_next_upstream_tries 0;
<i>Контекст</i>	http, server, location

Ограничивает число допустимых попыток для передачи запроса *следующему* серверу.

0	отключает это ограничение

grpc_pass

<i>Синтаксис</i>	grpc_pass <i>адрес</i> ;
По умолчанию	—
<i>Контекст</i>	location, if в location

Задаёт адрес gRPC-сервера. Адрес может быть указан в виде доменного имени или IP-адреса, и порта:

```
grpc_pass localhost:9000;
```

или в виде пути UNIX-сокета:

```
grpc_pass unix:/tmp/grpc.socket;
```

Также может использоваться схема `grpc://`:

```
grpc_pass grpc://127.0.0.1:9000;
```

Для использования gRPC по SSL необходимо использовать схему `grpcs://`:

```
grpc_pass grpcs://127.0.0.1:443;
```

Если доменному имени соответствует несколько адресов, то все они будут использоваться по очереди (round-robin). Кроме того, в качестве адреса можно указать *группу серверов*.

В значении параметра можно использовать переменные. В этом случае, если адрес указан в виде доменного имени, имя ищется среди описанных групп серверов и если не найдено, то определяется с помощью *resolver*'а.

Примечание

Если `grpc_pass` стоит в `location` с косой чертой в конце префикса (например, `location /name/`), и при этом в директиве `auto_redirect` указано `default`, запросы без косой черты в конце будут перенаправляться (`/name -> /name/`).

grpc_pass_header

<i>Синтаксис</i>	grpc_pass_header поле;
По умолчанию	—
<i>Контекст</i>	http, server, location

Разрешает передавать от gRPC-сервера клиенту *запрещенные для передачи* поля заголовка.

grpc_read_timeout

<i>Синтаксис</i>	grpc_read_timeout время;
По умолчанию	grpc_read_timeout 60s;
<i>Контекст</i>	http, server, location

Задаёт таймаут при чтении ответа gRPC-сервера. Таймаут устанавливается не на всю передачу ответа, а только между двумя операциями чтения. Если по истечении этого времени gRPC-сервер ничего не передаст, соединение закрывается.

grpc_send_timeout

<i>Синтаксис</i>	grpc_send_timeout время;
По умолчанию	grpc_send_timeout 60s;
<i>Контекст</i>	http, server, location

Задаёт таймаут при передаче запроса gRPC-серверу. Таймаут устанавливается не на всю передачу запроса, а только между двумя операциями записи. Если по истечении этого времени gRPC-сервер не примет новых данных, соединение закрывается.

grpc_set_header

<i>Синтаксис</i>	grpc_set_header поле значение;
По умолчанию	grpc_set_header Content-Length \$content_length;
<i>Контекст</i>	http, server, location

Позволяет переопределять или добавлять поля заголовка запроса, *передаваемые* проксируемому серверу. В качестве *значения* можно использовать текст, переменные и их комбинации. Директивы наследуются с предыдущего уровня конфигурации при условии, что на данном уровне не описаны свои директивы *grpc_set_header*.

Если значение поля заголовка — пустая строка, то поле вообще не будет передаваться gRPC-серверу:

```
grpc_set_header Accept-Encoding "";
```

grpc_socket_keepalive

<i>Синтаксис</i>	grpc_socket_keepalive on off;
По умолчанию	grpc_socket_keepalive off;
<i>Контекст</i>	http, server, location

Конфигурирует поведение "TCP keepalive" для исходящих соединений к проксируемому серверу.

" "	По умолчанию для сокета действуют настройки операционной системы.
on	для сокета включается параметр <i>SO_KEEPALIVE</i>

grpc_ssl_certificate

<i>Синтаксис</i>	grpc_ssl_certificate <i>файл</i> ;
По умолчанию	—
<i>Контекст</i>	http, server, location

Задаёт файл с сертификатом в формате PEM для аутентификации на gRPC SSL-сервере. В имени файла можно использовать переменные.

grpc_ssl_certificate_cache

<i>Синтаксис</i>	grpc_ssl_certificate_cache off; grpc_ssl_certificate_cache max= <i>N</i> [inactive= <i>time</i>] [valid= <i>time</i>];
Значение по умолчанию	grpc_ssl_certificate_cache off;
<i>Контекст</i>	http, server, location

Определяет кэш для хранения *SSL-сертификатов* и *секретных ключей*, заданных через переменные.

Директива поддерживает следующие параметры:

- **max** — устанавливает максимальное количество элементов в кэше. При переполнении кэша удаляются наименее недавно использованные (LRU) элементы.
- **inactive** — определяет время, после которого элемент будет удален, если к нему не было обращений. Значение по умолчанию — 10 секунд.
- **valid** — определяет время, в течение которого элемент кэша считается действительным и может использоваться повторно. Значение по умолчанию — 60 секунд. По истечении этого времени сертификаты перезагружаются или проходят повторную проверку.
- **off** — отключает кэш.

Пример:

```
grpc_ssl_certificate      $grpc_ssl_server_name.crt;
grpc_ssl_certificate_key  $grpc_ssl_server_name.key;
grpc_ssl_certificate_cache max=1000 inactive=20s valid=1m;
```

grpc_ssl_certificate_key

<i>Синтаксис</i>	grpc_ssl_certificate_key <i>файл</i> ;
По умолчанию	—
<i>Контекст</i>	http, server, location

Задаёт файл с секретным ключом в формате PEM для аутентификации на gRPC SSL-сервере.

Вместо файла можно указать значение "engine:*имя*:id", которое загружает ключ с указанным *id* из OpenSSL engine с заданным именем.

В имени файла можно использовать переменные.

grpc_ssl_ciphers

<i>Синтаксис</i>	grpc_ssl_ciphers <i>шифры</i> ;
По умолчанию	grpc_ssl_ciphers DEFAULT;
<i>Контекст</i>	http, server, location

Описывает разрешенные шифры для запросов к gRPC SSL-серверу. Шифры задаются в формате, поддерживаемом библиотекой OpenSSL.

Список шифров зависит от установленной версии OpenSSL. Полный список можно посмотреть с помощью команды `openssl ciphers`.

⚠ Предупреждение

Директива `grpc_ssl_ciphers` *не* настраивает шифры для TLS 1.3 при использовании OpenSSL. Для настройки шифров TLS 1.3 в OpenSSL используйте директиву `grpc_ssl_conf_command`, добавленную для расширенной конфигурации SSL.

- В LibreSSL шифры TLS 1.3 *можно* настраивать с помощью `grpc_ssl_ciphers`.
- В BoringSSL шифры TLS 1.3 настроить невозможно.

grpc_ssl_conf_command

<i>Синтаксис</i>	grpc_ssl_conf_command <i>имя значение</i> ;
По умолчанию	—
<i>Контекст</i>	http, server, location

Задаёт произвольные конфигурационные *команды* OpenSSL при установлении соединения с gRPC SSL-сервером.

ℹ Примечание

Директива поддерживается при использовании OpenSSL 1.0.2 и выше. Чтобы настроить шифры TLS 1.3 в OpenSSL, используйте команду `ciphersuites`.

На одном уровне может быть указано несколько директив *grpc_ssl_conf_command*. Директивы наследуются с предыдущего уровня конфигурации при условии, что на данном уровне не описаны свои директивы *grpc_ssl_conf_command*.

Предупреждение

Следует учитывать, что изменение настроек OpenSSL напрямую может привести к неожиданному поведению.

grpc_ssl_crl

<i>Синтаксис</i>	grpc_ssl_crl <i>файл</i> ;
По умолчанию	—
<i>Контекст</i>	http, server, location

Указывает файл с отозванными сертификатами (CRL) в формате PEM, используемыми при *проверке* сертификата gRPC SSL-сервера.

grpc_ssl_name

<i>Синтаксис</i>	grpc_ssl_name <i>имя</i> ;
По умолчанию	grpc_ssl_name `имя хоста из grpc_pass`;
<i>Контекст</i>	http, server, location

Позволяет переопределить имя сервера, используемое при *проверке* сертификата gRPC SSL-сервера, а также для *передачи его через SNI* при установлении соединения с gRPC SSL-сервером.

По умолчанию используется имя хоста из *grpc_pass*.

grpc_ssl_password_file

<i>Синтаксис</i>	grpc_ssl_password_file <i>файл</i> ;
По умолчанию	—
<i>Контекст</i>	http, server, location

Задаёт файл с паролями от *секретных ключей*, где каждый пароль указан на отдельной строке. Пароли применяются по очереди в момент загрузки ключа.

grpc_ssl_protocols

<i>Синтаксис</i>	grpc_ssl_protocols [SSLv2] [SSLv3] [TLSv1] [TLSv1.1] [TLSv1.2] [TLSv1.3];
По умолчанию	grpc_ssl_protocols TLSv1.2 TLSv1.3;
<i>Контекст</i>	http, server, location

Изменено в версии 1.2.0: Параметр TLSv1.3 добавлен к используемым по умолчанию.

Разрешает указанные протоколы для запросов к gRPC SSL-серверу.

grpc_ssl_server_name

<i>Синтаксис</i>	grpc_ssl_server_name on off;
По умолчанию	grpc_ssl_server_name off;
<i>Контекст</i>	http, server, location

Разрешает или запрещает передачу имени сервера, заданного директивой *grpc_ssl_name*, через расширение Server Name Indication протокола TLS (SNI, RFC 6066) при установлении соединения с SSL-сервером gRPC.

grpc_ssl_session_reuse

<i>Синтаксис</i>	grpc_ssl_session_reuse on off;
По умолчанию	grpc_ssl_session_reuse on;
<i>Контекст</i>	http, server, location

Определяет, использовать ли повторно SSL-сессии при работе с gRPC-сервером. Если в логах появляются ошибки "SSL3_GET_FINISHED:digest check failed", то можно попробовать выключить повторное использование сессий.

grpc_ssl_trusted_certificate

<i>Синтаксис</i>	grpc_ssl_trusted_certificate <i>файл</i> ;
По умолчанию	—
<i>Контекст</i>	http, server, location

Задаёт файл с доверенными сертификатами СА в формате PEM, используемыми при *проверке* сертификата gRPC SSL-сервера.

grpc_ssl_verify

<i>Синтаксис</i>	grpc_ssl_verify on off;
По умолчанию	grpc_ssl_verify off;
<i>Контекст</i>	http, server, location

Разрешает или запрещает проверку сертификата gRPC SSL-сервера.

grpc_ssl_verify_depth

<i>Синтаксис</i>	grpc_ssl_verify_depth число;
По умолчанию	grpc_ssl_verify_depth 1;
<i>Контекст</i>	http, server, location

Устанавливает глубину проверки в цепочке сертификатов gRPC SSL-сервера.

GunZIP

Фильтр, распаковывающий ответы с **Content-Encoding: gzip** для тех клиентов, которые не поддерживают метод сжатия "gzip". Модуль будет полезен, когда данные желательно хранить сжатыми для экономии места и сокращения затрат на ввод-вывод.

При сборке из исходного кода модуль не собирается по умолчанию; его необходимо включить с помощью параметра сборки `--with-http_gunzip_module`. В пакетах и образах из наших репозиториев модуль включен в сборку.

Пример конфигурации

```
location /storage/ {
    gunzip on;
    # ...
}
```

Директивы

gunzip

<i>Синтаксис</i>	gunzip on off;
По умолчанию	gunzip off;
<i>Контекст</i>	http, server, location

Разрешает или запрещает распаковку ответов, сжатых методом gzip, для тех клиентов, которые его не поддерживают. Если разрешено, то для определения, поддерживает ли клиент gzip, также учитываются следующие директивы: *gzip_http_version*, *gzip_proxied* и *gzip_disable*. См. также директиву *gzip_vary*.

gunzip_buffers

Синтаксис	gunzip_buffers <i>число</i> <i>размер</i> ;
По умолчанию	gunzip_buffers 32 4k 16 8k;
Контекст	http, server, location

Задаёт число и размер буферов, в которые будет разжиматься ответ. По умолчанию размер одного буфера равен размеру страницы. В зависимости от платформы это или 4К, или 8К.

GZip

Фильтр, сжимающий ответ методом gzip, что позволяет уменьшить размер передаваемых данных в 2 и более раз.

⚠ Предупреждение

При использовании протокола SSL/TLS сжатые ответы могут быть подвержены атакам BREACH.

Пример конфигурации

```
gzip on;
gzip_min_length 1000;
gzip_proxied expired no-cache no-store private auth;
gzip_types text/plain application/xml;
```

Для записи в лог достигнутого коэффициента сжатия можно использовать переменную `$gzip_ratio`.

Директивы

gzip

Синтаксис	gzip on off;
По умолчанию	gzip off;
Контекст	http, server, location, if в location

Разрешает или запрещает сжатие ответа методом gzip.

gzip_buffers

<i>Синтаксис</i>	gzip_buffers <i>число</i> <i>размер</i> ;
По умолчанию	gzip_buffers 32 4k 16 8k;
<i>Контекст</i>	http, server, location

Задаёт число и размер буферов, в которые будет сжиматься ответ. По умолчанию размер одного буфера равен размеру страницы. В зависимости от платформы это или 4К, или 8К.

gzip_comp_level

<i>Синтаксис</i>	gzip_comp_level <i>степень</i> ;
По умолчанию	gzip_comp_level 1;
<i>Контекст</i>	http, server, location

Устанавливает степень сжатия ответа методом gzip. Допустимые значения находятся в диапазоне от 1 до 9.

gzip_disable

<i>Синтаксис</i>	gzip_disable <i>regex</i> ...;
По умолчанию	—
<i>Контекст</i>	http, server, location

Запрещает сжатие ответа методом gzip для запросов с полями заголовка **User-Agent**, совпадающими с заданными регулярными выражениями.

Специальная маска **msie6** соответствует регулярному выражению "*MSIE [4-6].*", но работает быстрее. Из этой маски исключается "*MSIE 6.0; ... SV1*".

gzip_http_version

<i>Синтаксис</i>	gzip_http_version 1.0 1.1;
По умолчанию	gzip_http_version 1.1;
<i>Контекст</i>	http, server, location

Устанавливает минимальную HTTP-версию запроса, необходимую для сжатия ответа.

gzip_min_length

<i>Синтаксис</i>	gzip_min_length <i>длина</i> ;
По умолчанию	gzip_min_length 20;
<i>Контекст</i>	http, server, location

Устанавливает минимальную длину ответа, который будет сжиматься методом gzip. Длина определяется только из поля **Content-Length** заголовка ответа.

gzip_proxied

<i>Синтаксис</i>	gzip_proxied off expired no-cache no-store private no_last_modified no_etag auth any ...;
По умолчанию	gzip_proxied off;
<i>Контекст</i>	http, server, location

Разрешает или запрещает сжатие ответа методом gzip для проксированных запросов в зависимости от запроса и ответа. То, что запрос проксированный, определяется на основании наличия поля **Via** в заголовке запроса. В директиве можно указать одновременно несколько параметров:

off	запрещает сжатие для всех проксированных запросов, игнорируя остальные параметры;
expired	разрешает сжатие, если в заголовке ответа есть поле Expires со значением, запрещающим кэширование;
no-cache	разрешает сжатие, если в заголовке ответа есть поле Cache-Control с параметром "no-cache";
no-store	разрешает сжатие, если в заголовке ответа есть поле Cache-Control с параметром "no-store";
private	разрешает сжатие, если в заголовке ответа есть поле Cache-Control с параметром "private";
no_last_modified	разрешает сжатие, если в заголовке ответа нет поля Last-Modified ;
no_etag	разрешает сжатие, если в заголовке ответа нет поля ETag ;
auth	разрешает сжатие, если в заголовке запроса есть поле Authorization ;
any	разрешает сжатие для всех проксированных запросов.

gzip_types

<i>Синтаксис</i>	gzip_types <i>mime-тип ...</i> ;
По умолчанию	gzip_types text/html;
<i>Контекст</i>	http, server, location

Разрешает сжатие ответа методом gzip для указанных MIME-типов в дополнение к **text/html**. Специальное значение "*" соответствует любому MIME-типу. Ответы с типом **text/html** сжимаются всегда.

gzip_vary

<i>Синтаксис</i>	<code>gzip_vary on off;</code>
По умолчанию	<code>gzip_vary off;</code>
<i>Контекст</i>	http, server, location

Разрешает или запрещает выдавать в ответе поле заголовка "Vary: Accept-Encoding", если активны директивы `gzip`, `gzip_static` или `gunzip`.

Встроенные переменные

`$gzip_ratio`

достигнутый коэффициент сжатия — отношение размера исходного ответа к размеру сжатого.

GZip Static

Позволяет отдавать вместо обычного файла предварительно сжатый файл с таким же именем и с расширением ".gz".

При сборке из исходного кода модуль не собирается по умолчанию; его необходимо включить с помощью параметра сборки `--with-http_gzip_static_module`. В пакетах и образах из наших репозиториях модуль включен в сборку.

Пример конфигурации

```
gzip_static on;
gzip_proxied expired no-cache no-store private auth;
```

Директивы

gzip_static

<i>Синтаксис</i>	<code>gzip_static on off always;</code>
По умолчанию	<code>gzip_static off;</code>
<i>Контекст</i>	http, server, location

Разрешает (`on`) или запрещает (`off`) проверку готового сжатого файла. При использовании также учитываются директивы `gzip_http_version`, `gzip_proxied`, `gzip_disable` и `gzip_vary`.

Со значением `always` во всех случаях будет использоваться сжатый файл, без проверки поддержки на стороне клиента. Это полезно, если на диске все равно нет несжатых файлов, или используется модуль *GunZIP*.

Сжимать файлы можно с помощью программы `gzip` или совместимой с ней. Желательно, чтобы дата и время модификации исходного и сжатого файлов совпадали.

Headers

Позволяет выдавать поля заголовка **Expires** и **Cache-Control**, а также добавлять произвольные поля в заголовок ответа.

Пример конфигурации

```
expires 24h;
expires modified +24h;
expires @24h;
expires 0;
expires -1;
expires epoch;
expires $expires;
add_header Cache-Control private;
```

Директивы

add_header

<i>Синтаксис</i>	<code>add_header имя значение [always];</code>
По умолчанию	—
<i>Контекст</i>	http, server, location, if в location

Добавляет указанное поле в заголовок ответа при условии, что код ответа равен 200, 201 (1.3.10), 204, 206, 301, 302, 303, 304, 307 или 308. В значении параметра можно использовать переменные.

Директив `add_header` может быть несколько. Директивы наследуются с предыдущего уровня конфигурации при условии, что на данном уровне не описаны свои директивы `add_header`.

Если указан параметр **always**, то поле заголовка будет добавлено независимо от кода ответа.

add_trailer

<i>Синтаксис</i>	<code>add_trailer имя значение [always];</code>
По умолчанию	—
<i>Контекст</i>	http, server, location, if в location

Добавляет указанное поле в конец ответа при условии, что код ответа равен 200, 201, 206, 301, 302, 303, 307 или 308. В значении можно использовать переменные.

Директив `add_trailer` может быть несколько. Директивы наследуются с предыдущего уровня конфигурации при условии, что на данном уровне не описаны свои директивы `add_trailer`.

Если указан параметр **always**, то указанное поле будет добавлено независимо от кода ответа.

expires

<i>Синтаксис</i>	<code>expires [modified] <i>время</i>;</code> <code>expires epoch max off;</code>
По умолчанию	<code>expires off;</code>
<i>Контекст</i>	http, server, location, if в location

Разрешает или запрещает добавлять или менять поля **Expires** и **Cache-Control** в заголовке ответа при условии, что код ответа равен 200, 201, 204, 206, 301, 302, 303, 304, 307 или 308. В качестве параметра можно задать положительное или отрицательное *время*.

Время в поле **Expires** получается как сумма текущего времени и времени, заданного в директиве. Если используется параметр **modified**, то время получается как сумма времени модификации файла и времени, заданного в директиве.

Кроме того, с помощью префикса "@" можно задать время суток:

```
expires @15h30m;
```

Содержимое поля **Cache-Control** зависит от знака заданного времени:

- отрицательное время — "Cache-Control: no-cache".
- положительное или равное нулю время — "Cache-Control: max-age=`t`", где *t* это время в секундах, заданное в директиве.

epoch	задает время "Thu, 01 Jan 1970 00:00:01 GMT" (1 января 1970 00:00:01 GMT) для поля Expires и "no-cache" для поля Cache-Control .
max	задает время "Thu, 31 Dec 2037 23:55:55 GMT" (31 декабря 2037 23:55:55 GMT) для поля Expires и 10 лет для поля Cache-Control .
off	запрещает добавлять или менять поля Expires и Cache-Control в заголовке ответа.

В значении последнего параметра можно использовать переменные:

```
map $sent_http_content_type $expires {
    default      off;
    application/pdf 42d;
    ~image/      max;
}

expires $expires;
```

Index

Обслуживает запросы, оканчивающиеся косой чертой (/). Такие запросы также могут обслуживаться модулями *AutoIndex* и *Random Index*.

Пример конфигурации

```
location / {
    index index.$geo.html index.html;
}
```

Директивы

index

<i>Синтаксис</i>	<code>index файл ...;</code>
По умолчанию	<code>index index.html;</code>
<i>Контекст</i>	http, server, location

Определяет файлы, которые будут использоваться в качестве индекса. В имени файла можно использовать переменные. Наличие файлов проверяется в порядке их перечисления. В конце списка может стоять файл с абсолютным путем. Пример:

```
index index.$geo.html index.0.html /index.html;
```

Необходимо иметь в виду, что при использовании индексного файла делается внутреннее перенаправление и запрос может быть обработан уже в другом `location`. Например, в такой конфигурации:

```
location = / {
    index index.html;
}

location / {
#    ...
}
```

запрос "/" будет фактически обработан во втором `location` как `/index.html`.

Limit Conn

Позволяет ограничить число соединений по заданному ключу, в частности, число соединений с одного IP-адреса.

Учитываются не все соединения, а лишь те, в которых имеются запросы, обрабатываемые сервером, и заголовок запроса уже прочитан.

Пример конфигурации

```
http {
    limit_conn_zone $binary_remote_addr zone=addr:10m;

    ...

    server {
        ...
    }
}
```

```
location /download/ {
    limit_conn addr 1;
}
```

Директивы

limit_conn

<i>Синтаксис</i>	limit_conn зона число;
По умолчанию	—
<i>Контекст</i>	http, server, location

Задаёт зону разделяемой памяти и максимально допустимое число соединений для одного значения ключа. При превышении этого числа в ответ на запрос сервер вернёт *ошибку*. Например, директивы

```
limit_conn_zone $binary_remote_addr zone=addr:10m;

server {
    location /download/ {
        limit_conn addr 1;
    }
}
```

разрешают одновременно обрабатывать не более одного соединения с одного IP-адреса.

Примечание

В HTTP/2 и HTTP/3 каждый параллельный запрос считается отдельным соединением.

Директив *limit_conn* может быть несколько. Например, следующая конфигурация ограничивает число соединений с сервером с одного клиентского IP-адреса и в то же время ограничивает общее число соединений с виртуальным сервером:

```
limit_conn_zone $binary_remote_addr zone=perip:10m;
limit_conn_zone $server_name zone=perserver:10m;

server {
    ...
    limit_conn perip 10;
    limit_conn perserver 100;
}
```

Директивы наследуются с предыдущего уровня конфигурации при условии, что на данном уровне не описаны свои директивы *limit_conn*.

limit_conn_dry_run

<i>Синтаксис</i>	limit_conn_dry_run on off;
По умолчанию	limit_conn_dry_run off;
<i>Контекст</i>	http, server, location

Включает режим пробного запуска. В данном режиме число соединений не ограничивается, однако в *зоне разделяемой памяти* текущее число избыточных соединений учитывается как обычно.

limit_conn_log_level

<i>Синтаксис</i>	limit_conn_log_level info notice warn error;
По умолчанию	limit_conn_log_level error;
<i>Контекст</i>	http, server, location

Задаёт желаемый уровень записи в лог случаев ограничения числа соединений.

limit_conn_status

<i>Синтаксис</i>	limit_conn_status код;
По умолчанию	limit_conn_status 503;
<i>Контекст</i>	http, server, location

Позволяет переопределить код ответа, используемый при отклонении запросов.

limit_conn_zone

<i>Синтаксис</i>	limit_conn_zone <i>ключ</i> zone = <i>название:размер</i> ;
По умолчанию	—
<i>Контекст</i>	http

Задаёт параметры зоны разделяемой памяти, которая хранит состояние для разных значений ключа. Состояние в частности содержит текущее число соединений. В качестве ключа можно использовать текст, переменные и их комбинации. Запросы с пустым значением ключа не учитываются.

Пример использования:

```
limit_conn_zone $binary_remote_addr zone=addr:10m;
```

Здесь в качестве ключа используется IP-адрес клиента. Обратите внимание, что вместо переменной \$remote_addr использована переменная \$binary_remote_addr.

Длина значения переменной \$remote_addr может колебаться от 7 до 15 байт, при этом размер хранимого состояния составляет либо 32, либо 64 байта на 32-битных платформах и всегда 64 байта на 64-битных.

Длина значения переменной `$binary_remote_addr` всегда равна 4 байтам для IPv4-адресов или 16 байтам для IPv6-адресов. При этом размер состояния всегда равен 32 или 64 байтам на 32-битных платформах и 64 байтам на 64-битных.

В зоне размером 1 мегабайт может разместиться около 32 тысяч состояний размером 32 байта или 16 тысяч состояний размером 64 байта. При переполнении зоны в ответ на последующие запросы сервер будет возвращать *ошибку*.

Встроенные переменные

`$limit_conn_status`

хранит результат ограничения числа соединений: *PASSED*, *REJECTED* или *REJECTED_DRY_RUN*

Limit Req

Позволяет ограничить скорость обработки запросов по заданному ключу или, как частный случай, скорость обработки запросов, поступающих с одного IP-адреса. Ограничение обеспечивается с помощью метода "leaky bucket".

Пример конфигурации

```
http {
    limit_req_zone $binary_remote_addr zone=one:10m rate=1r/s;

    ...

    server {

        ...

        location /search/ {
            limit_req zone=one burst=5;
        }
    }
}
```

Директивы

limit_req

<i>Синтаксис</i>	<code>limit_req zone=название [burst=число] [nodelay delay=число];</code>
По умолчанию	—
Контекст	http, server, location

Задаёт зону разделяемой памяти (*zone*) и максимальный размер всплеска запросов (*burst*). Если скорость поступления запросов превышает описанную в зоне, то их обработка задерживается так, чтобы запросы обрабатывались с заданной скоростью. Избыточные запросы задерживаются до тех пор, пока их число не превысит максимальный размер всплеска. При превышении запрос завершается с ошибкой. По умолчанию максимальный размер всплеска равен нулю. Например, директивы

```
limit_req_zone $binary_remote_addr zone=one:10m rate=1r/s;

server {
    location /search/ {
        limit_req zone=one burst=5;
    }
}
```

позволяют в среднем не более 1 запроса в секунду со всплесками не более 5 запросов.

Если же избыточные запросы в пределах лимита всплесков задерживать не требуется, то следует использовать параметр `nodelay`:

```
limit_req zone=one burst=5 nodelay;
```

Параметр `delay` задает лимит, по достижении которого избыточные запросы задерживаются. Значение по умолчанию равно нулю и означает, что задерживаются все избыточные запросы.

Директив `limit_req` может быть несколько. Например, следующая конфигурация ограничивает скорость обработки запросов, поступающих с одного IP-адреса, и в то же время ограничивает скорость обработки запросов одним виртуальным сервером:

```
limit_req_zone $binary_remote_addr zone=perip:10m rate=1r/s;
limit_req_zone $server_name zone=perserver:10m rate=10r/s;

server {
    ...
    limit_req zone=perip burst=5 nodelay;
    limit_req zone=perserver burst=10;
}
```

Директивы наследуются с предыдущего уровня конфигурации при условии, что на данном уровне не описаны свои директивы `limit_req`.

limit_req_dry_run

<i>Синтаксис</i>	<code>limit_req_dry_run on off;</code>
По умолчанию	<code>limit_req_dry_run off;</code>
<i>Контекст</i>	<code>http, server, location</code>

Включает режим пробного запуска. В данном режиме скорость обработки запросов не ограничивается, однако в *зоне разделяемой памяти* текущее число избыточных запросов учитывается как обычно.

limit_req_log_level

<i>Синтаксис</i>	<code>limit_req_log_level info notice warn error;</code>
По умолчанию	<code>limit_req_log_level error;</code>
<i>Контекст</i>	<code>http, server, location</code>

Задаёт желаемый уровень записи в лог случаев отказа в обработке запросов при превышении скорости и случаев задержек при обработке запроса. Задержки записываются в лог с уровнем на единицу

меньшим, чем отказы, например, если указано `limit_req_log_level notice`, то задержки будут записываться в лог на уровне `info`.

limit_req_status

<i>Синтаксис</i>	<code>limit_req_status код;</code>
По умолчанию	<code>limit_req_status 503;</code>
<i>Контекст</i>	<code>http, server, location</code>

Позволяет переопределить код ответа, используемый при отклонении запросов.

limit_req_zone

<i>Синтаксис</i>	<code>limit_req_zone ключ zone=название:размер rate=скорость;</code>
По умолчанию	—
<i>Контекст</i>	<code>http</code>

Задаёт параметры зоны разделяемой памяти, которая хранит состояние для разных значений ключа. Состояние в частности хранит текущее число избыточных запросов. В качестве ключа можно использовать текст, переменные и их комбинации. Запросы с пустым значением ключа не учитываются.

Пример использования:

```
limit_req_zone $binary_remote_addr zone=one:10m rate=1r/s;
```

В данном случае состояния хранятся в зоне `one` размером 10 мегабайт, и средняя скорость обработки запросов для этой зоны не может превышать 1 запроса в секунду.

В качестве ключа используется IP-адрес клиента. Обратите внимание, что вместо переменной `$remote_addr` используется переменная `$binary_remote_addr`.

Длина значения переменной `$binary_remote_addr` всегда равна 4 байтам для IPv4-адресов или 16 байтам для IPv6-адресов. При этом размер состояния всегда равен 64 байтам на 32-битных платформах и 128 байтам на 64-битных платформах.

В зоне размером 1 мегабайт может разместиться около 16 тысяч состояний размером 64 байта или около 8 тысяч состояний размером 128 байт.

При переполнении зоны удаляется наименее востребованное состояние. Если и это не позволяет создать новое состояние, запрос завершается с *ошибкой*.

Скорость `rate` задается в запросах в секунду (r/s). Если же нужна скорость меньше одного запроса в секунду, то она задается в запросах в минуту (r/m), например, ползапроса в секунду — это 30r/m.

Встроенные переменные

`$limit_req_status`

хранит результат ограничения скорости поступления запросов: `PASSED`, `DELAYED`, `REJECTED`, `DELAYED_DRY_RUN` или `REJECTED_DRY_RUN`

Log

Модуль записывает логи запросов в указанном формате.

Логи записываются в контексте `location`, где заканчивается обработка. Это может быть `location`, отличный от первоначального, если в процессе обработки запроса происходит *внутреннее перенаправление*.

Пример конфигурации

```
log_format compression '$remote_addr - $remote_user [$time_local] '
                        '$request' $status $bytes_sent '
                        '$http_referer' '$http_user_agent' '$gzip_ratio';

access_log /spool/logs/angie-access.log compression buffer=32k;
```

Директивы

`access_log`

<i>Синтаксис</i>	<code>access_log <i>путь</i> [<i>формат</i> [<i>buffer=размер</i>] [<i>gzip=степень</i>]] [<i>flush=время</i>] [<i>if=условие</i>];</code> <code>access_log off;</code>
По умолчанию	<code>access_log logs/access.log combined;</code> (путь зависит от параметра сборки <code>--http-log-path</code>)
<i>Контекст</i>	<code>http</code> , <code>server</code> , <code>location</code> , <code>if</code> в <code>location</code> , <code>limit_except</code>

Задаёт *путь*, *формат* и настройки буферизованной записи в лог. На одном уровне конфигурации может использоваться несколько логов. Запись в *syslog* настраивается указанием префикса *"syslog:"* в первом параметре. Специальное значение *off* отменяет все директивы *access_log* для текущего уровня. Если формат не указан, то используется предопределённый формат *"combined"*.

Если задан размер буфера с помощью параметра *buffer* или указан параметр *gzip*, то запись будет буферизованной.

Предупреждение

Размер буфера должен быть не больше размера атомарной записи в дисковый файл. Для FreeBSD этот размер неограничен.

При включенной буферизации данные записываются в файл:

- если очередная строка лога не помещается в буфер;
- если данные в буфере находятся дольше интервала времени, заданного параметром *flush*;

- при переоткрытии лог-файла или завершении рабочего процесса.

Если задан параметр `gzip`, то буфер будет сжиматься перед записью в файл. Степень сжатия может быть задана в диапазоне от 1 (быстрее, но хуже сжатие) до 9 (медленнее, но лучше сжатие). По умолчанию используются буфер размером *64K* байт и степень сжатия *1*. Данные сжимаются атомарными блоками, и в любой момент времени лог-файл может быть распакован или прочитан с помощью утилиты `zcat`.

Пример:

```
access_log /path/to/log.gz combined gzip flush=5m;
```

Примечание

Для поддержки gzip-сжатия логов Angie должен быть собран с библиотекой *zlib*.

В пути файла можно использовать переменные, но такие логи имеют некоторые ограничения:

- *пользователь*, с правами которого работают рабочие процессы, должен иметь права на создание файлов в каталоге с такимилогами;
- не работает буферизация;
- файл открывается для каждой записи в лог и сразу же после записи закрывается. Следует однако иметь в виду, что поскольку дескрипторы часто используемых файлов могут храниться в кэше, то при ротации логов в течение времени, заданного параметром *valid* директивы *open_log_file_cache*, запись может продолжаться в старый файл.
- при каждой записи в лог проверяется существование корневого каталога для запроса — если этот каталог не существует, то лог не создается. Поэтому *root* и *access_log* нужно описывать на одном уровне конфигурации:

```
server {
    root      /spool/vhost/data/$host;
    access_log /spool/vhost/logs/$host;
    ...
}
```

Параметр `if` включает условную запись в лог. Запрос не будет записываться в лог, если результатом вычисления условия является "0" или пустая строка. В следующем примере запросы с кодами ответа 2xx и 3xx не будут записываться в лог:

```
map $status $loggable {
    ~^[23] 0;
    default 1;
}

access_log /path/to/access.log combined if=$loggable;
```

log_format

<i>Синтаксис</i>	<code>log_format имя [escape= default json none] строка ...;</code>

По умолчанию	<code>log_format combined "...";</code>

<i>Контекст</i>	<code>http</code>

Задаёт формат лога.

Параметр `escape` позволяет задать экранирование символов `json` или `default` в переменных, по умолчанию используется `default`. Значение `none` отключает экранирование символов.

При использовании `default` символы `"`, `\`, а также символы со значениями меньше 32 или больше 126 экранируются как `"\xXX"`. Если значение переменной не найдено, то в качестве значения в лог будет записываться дефис `-`.

При использовании `json` экранируются все символы, недопустимые в JSON строках: символы `"` и `\` экранируются как `"\"` и `\"`, символы со значениями меньше 32 экранируются как `"\n"`, `"\r"`, `"\t"`, `"\b"`, `"\f"` или `"\u00XX"`.

Строки заголовка, переданные клиенту, начинаются с префикса `sent_http_`, например, `$sent_http_content_range`.

В конфигурации всегда существует предопределенный формат `combined`:

```
log_format combined '$remote_addr - $remote_user [$time_local] '
                    '"$request" $status $body_bytes_sent '
                    '"$http_referer" "$http_user_agent";'
```

open_log_file_cache

<i>Синтаксис</i>	<code>open_log_file_cache max=N [inactive=время] [min_uses=N] [valid=время];</code> <code>open_log_file_cache off;</code>
По умолчанию	<code>open_log_file_cache off;</code>
<i>Контекст</i>	<code>http, server, location</code>

Задаёт кэш, в котором хранятся дескрипторы файлов часто используемых логов, имена которых заданы с использованием переменных. Параметры:

<code>max</code>	Задаёт максимальное число дескрипторов в кэше; при переполнении кэша наименее востребованные (LRU) дескрипторы закрываются.
<code>inactive</code>	Задаёт время, после которого кэшированный дескриптор закрывается, если к нему не было обращений в течение этого времени. По умолчанию — 10 секунд.
<code>min_uses</code>	Задаёт минимальное число использований файла в течение времени, заданного параметром <code>inactive</code> , после которого дескриптор файла будет оставаться открытым в кэше. По умолчанию — 1.
<code>valid</code>	Указывает, через какое время нужно проверять, что файл ещё существует под тем же именем. По умолчанию — 60 секунд.
<code>off</code>	Запрещает кэширование.

Пример использования:

```
open_log_file_cache max=1000 inactive=20s valid=1m min_uses=2;
```

Map

Создает переменные, значения которых зависят от значений других переменных.

Пример конфигурации

```
map $http_host $name {
    hostnames;

    default      0;

    example.com  1;
    *.example.com 1;
    example.org   2;
    *.example.org 2;
    .example.net  3;
    wap.*         4;
}

map $http_user_agent $mobile {
    default      0;
    "~Opera Mini" 1;
}
```

Директивы

map

<i>Синтаксис</i>	<code>map строка \$переменная { ... }</code>
По умолчанию	—
<i>Контекст</i>	http

Создает новую переменную. Ее значение зависит от первого параметра, заданного строкой с переменными, например:

```
set $var1 "foo";
set $var2 "bar";

map $var1$var2 $new_variable {
    default "foobar_value";
}
```

Здесь переменная `$new_variable` будет иметь значение, составленное из двух переменных `$var1` и `$var2`, или значение по умолчанию, если эти переменные не определены.

Примечание

Поскольку переменные вычисляются только в момент использования, само по себе наличие даже большого числа объявлений переменных "map" не влечет за собой никаких дополнительных расходов на обработку запросов.

Параметры внутри блока **map** задают соответствие между исходными и результирующими значениями.

Исходные значения задаются строками или регулярными выражениями.

Строки проверяются без учета регистра.

Перед регулярным выражением ставится символ `~`, если при сравнении следует учитывать регистр символов, либо символы `~*`, если регистр символов учитывать не нужно. Регулярное выражение может содержать именованные и позиционные группы захвата, которые могут затем использоваться в других директивах совместно с результирующей переменной.

Если исходное значение совпадает с именем одного из специальных параметров, описанных ниже, перед ним следует поставить символ `\`.

В качестве результирующего значения можно указать текст, переменную и их комбинации.

Также поддерживаются следующие специальные параметры:

default <i>значение</i>	задает результирующее значение, если исходное значение не совпадает ни с одним из перечисленных. Если параметр <i>default</i> не указан, результирующим значением по умолчанию будет пустая строка.
hostnames	указывает, что в качестве исходных значений можно использовать маску для первой или последней части имени хоста. Этот параметр следует указывать перед списком значений.

Например,

```
*.example.com 1;
example.*      1;
```

Вместо двух записей

```
example.com    1;
*.example.com  1;
```

можно использовать одну:

```
.example.com 1;
```

include <i>файл</i>	включает файл со значениями. Включений может быть несколько.
volatile	указывает, что переменная не кэшируется.

Если исходному значению соответствует несколько из указанных вариантов, например, одновременно подходят и маска, и регулярное выражение, будет выбран первый подходящий вариант в следующем порядке приоритета:

1. Строковое значение без маски.
2. Самое длинное строковое значение с маской в начале, например `*.example.com`.
3. Самое длинное строковое значение с маской в конце, например `mail.*`.
4. Первое подходящее регулярное выражение (в порядке следования в конфигурационном файле).
5. Значение по умолчанию (*default*).

map_hash_bucket_size

<i>Синтаксис</i>	map_hash_bucket_size <i>размер</i> ;
По умолчанию	map_hash_bucket_size 32 64 128;
<i>Контекст</i>	http

Задаёт размер корзины в хэш-таблицах для переменных *map*. Значение по умолчанию зависит от размера строки кэша процессора. Подробнее настройка хэш-таблиц обсуждается *отдельно*.

map_hash_max_size

<i>Синтаксис</i>	map_hash_max_size <i>размер</i> ;
По умолчанию	map_hash_max_size 2048;
<i>Контекст</i>	http

Задаёт максимальный размер хэш-таблиц для переменных *map*. Подробнее настройка хэш-таблиц обсуждается *отдельно*.

Memcached

Позволяет получать ответ из сервера memcached. Ключ задается в переменной *\$memcached_key*. Ответ в memcached должен быть предварительно помещен внешним по отношению к Angie способом.

Пример конфигурации

```
server {
    location / {
        set             $memcached_key "$uri?$args";
        memcached_pass  host:11211;
        error_page      404 502 504 = @fallback;
    }

    location @fallback {
        proxy_pass      http://backend;
    }
}
```

Директивы

memcached_bind

<i>Синтаксис</i>	memcached_bind <i>адрес</i> [transparent] off;
По умолчанию	—
<i>Контекст</i>	http, server, location

Задаёт локальный IP-адрес с необязательным портом, который будет использоваться в исходящих соединениях с сервером memcached. В значении параметра допустимо использование переменных. Специальное значение `off` отменяет действие унаследованной с предыдущего уровня конфигурации директивы `memcached_bind`, позволяя системе самостоятельно выбирать локальный IP-адрес и порт.

Параметр `transparent` позволяет задать нелокальный IP-адрес, который будет использоваться в исходящих соединениях с проксируемым сервером, например, реальный IP-адрес клиента:

```
memcached_bind $remote_addr transparent;
```

Для работы параметра обычно требуется запустить рабочие процессы Angie с привилегиями *суперпользователя*. В Linux это не требуется, так как если указан параметр `transparent`, то рабочие процессы наследуют *capability CAP_NET_RAW* из главного процесса.

Примечание

Необходимо настроить таблицу маршрутизации ядра для перехвата сетевого трафика с сервера memcached.

memcached_buffer_size

<i>Синтаксис</i>	<code>memcached_buffer_size размер;</code>
По умолчанию	<code>memcached_buffer_size 4k 8k;</code>
<i>Контекст</i>	<code>http, server, location</code>

Задаёт размер буфера, в который будет читаться первая часть ответа, получаемого от сервера memcached. Ответ синхронно передается клиенту сразу же по мере его поступления.

memcached_connect_timeout

<i>Синтаксис</i>	<code>memcached_connect_timeout время;</code>
По умолчанию	<code>memcached_connect_timeout 60s;</code>
<i>Контекст</i>	<code>http, server, location</code>

Задаёт таймаут для установления соединения с сервером memcached. Необходимо иметь в виду, что этот таймаут обычно не может превышать 75 секунд.

memcached_gzip_flag

<i>Синтаксис</i>	<code>memcached_gzip_flag флаг;</code>
По умолчанию	—
<i>Контекст</i>	<code>http, server, location</code>

Включает проверку указанного флага в ответе сервера memcached и установку поля `Content-Encoding` заголовка ответа в "gzip", если этот флаг установлен.

memcached_next_upstream

<i>Синтаксис</i>	memcached_next_upstream error timeout invalid_response not_found off ...;
По умолчанию	memcached_next_upstream error timeout;
<i>Контекст</i>	http, server, location

Определяет, в каких случаях запрос будет передан следующему в группе *upstream* серверу:

error	произошла ошибка соединения с сервером, передачи ему запроса или чтения заголовка ответа сервера;
timeout	произошел таймаут во время соединения с сервером, передачи ему запроса или чтения заголовка ответа сервера;
invalid_response	сервер вернул пустой или неверный ответ;
not_found	сервер не нашел ответ;
off	запрещает передачу запроса следующему серверу.

Примечание

Необходимо понимать, что передача запроса следующему серверу возможна только при условии, что клиенту еще ничего не передавалось. То есть, если ошибка или таймаут возникли в середине передачи ответа клиенту, то действие директивы на такой запрос не распространяется.

Директива также определяет, что считается *неудачной попыткой* работы с сервером.

error, timeout, invalid_response	Всегда считаются неудачными попытками, даже если они не указаны в директиве.
not_found	Никогда не считается неудачными попытками.

Передача запроса следующему серверу может быть ограничена по *количеству попыток* и по *времени*.

memcached_next_upstream_timeout

<i>Синтаксис</i>	memcached_next_upstream_timeout <i>время</i> ;
По умолчанию	memcached_next_upstream_timeout 0;
<i>Контекст</i>	http, server, location

Ограничивает время, в течение которого возможна передача запроса *следующему* серверу.

0	отключает это ограничение

memcached_next_upstream_tries

<i>Синтаксис</i>	memcached_next_upstream_tries <i>число</i> ;
По умолчанию	memcached_next_upstream_tries 0;
<i>Контекст</i>	http, server, location

Ограничивает число допустимых попыток для передачи запроса *следующему* серверу.

0	отключает это ограничение

memcached_pass

<i>Синтаксис</i>	memcached_pass <i>адрес</i> ;
По умолчанию	—
<i>Контекст</i>	location, if в location

Задаёт адрес сервера memcached. Адрес может быть указан в виде доменного имени или IP-адреса, и порта:

```
memcached_pass localhost:11211;
```

или в виде пути UNIX-сокета:

```
memcached_pass unix:/tmp/memcached.socket;
```

Если доменному имени соответствует несколько адресов, то все они будут использоваться по очереди (round-robin). Кроме того, в качестве адреса можно указать *группу серверов*.

Примечание

Если memcached_pass стоит в location с косой чертой в конце префикса (например, location /name/), и при этом в директиве *auto_redirect* указано default, запросы без косой черты в конце будут перенаправляться (/name -> /name/).

memcached_read_timeout

<i>Синтаксис</i>	memcached_read_timeout <i>время</i> ;
По умолчанию	memcached_read_timeout 60s;
<i>Контекст</i>	http, server, location

Задаёт таймаут при чтении ответа сервера memcached. Таймаут устанавливается не на всю передачу ответа, а только между двумя операциями чтения. Если по истечении этого времени сервер memcached ничего не передаст, соединение закрывается.

memcached_send_timeout

<i>Синтаксис</i>	memcached_send_timeout <i>время</i> ;
По умолчанию	memcached_send_timeout 60s;
<i>Контекст</i>	http, server, location

Задаёт таймаут при передаче запроса серверу memcached. Таймаут устанавливается не на всю передачу запроса, а только между двумя операциями записи. Если по истечении этого времени сервер memcached не примет новых данных, соединение закрывается.

memcached_socket_keepalive

<i>Синтаксис</i>	memcached_socket_keepalive on off;
По умолчанию	memcached_socket_keepalive off;
<i>Контекст</i>	http, server, location

Конфигурирует поведение "TCP keepalive" для исходящих соединений к проксируемому серверу.

" "	По умолчанию для сокета действуют настройки операционной системы.
on	для сокета включается параметр <i>SO_KEEPALIVE</i>

Встроенные переменные

\$memcached_key

Задаёт ключ для получения ответа из сервера memcached.

Mirror

Позволяет зеркалировать исходный запрос при помощи создания фоновых зеркалирующих подзапросов. Ответы на зеркалирующие подзапросы игнорируются.

Пример конфигурации

```
location / {
    mirror /mirror;
    proxy_pass http://backend;
}

location = /mirror {
    internal;
    proxy_pass http://test_backend$request_uri;
}
```

Директивы

mirror

<i>Синтаксис</i>	<code>mirror uri off;</code>
По умолчанию	<code>mirror off;</code>
<i>Контекст</i>	<code>http, server, location</code>

Задаёт URI, на который будет зеркалироваться исходный запрос. На одном уровне конфигурации может быть задано несколько зеркал.

mirror_request_body

<i>Синтаксис</i>	<code>mirror_request_body on off;</code>
По умолчанию	<code>mirror_request_body on;</code>
<i>Контекст</i>	<code>http, server, location</code>

Определяет, зеркалировать ли тело запроса клиента. Если включено, то тело запроса клиента будет прочитано перед созданием зеркалирующих подзапросов. В этом случае небуферизованное проксирование тела запроса клиента, задаваемое директивами *proxy_request_buffering*, *fastcgi_request_buffering*, *scgi_request_buffering* и *uwsgi_request_buffering*, будет отключено.

```
location / {
    mirror /mirror;
    mirror_request_body off;
    proxy_pass http://backend;
}

location = /mirror {
    internal;
    proxy_pass http://log_backend;
    proxy_pass_request_body off;
    proxy_set_header Content-Length "";
    proxy_set_header X-Original-URI $request_uri;
}
```

MP4

Обеспечивает серверную поддержку псевдо-стриминга для файлов в формате MP4. Такие файлы обычно имеют расширения `.mp4`, `.m4v` и `.m4a`.

Псевдо-стриминг работает в паре с совместимым медиаплеером. Плеер посылает серверу HTTP-запрос с указанием точки времени старта в аргументе `start` строки запроса (время задается в секундах), а сервер в ответ посылает поток, у которого начальная позиция соответствует запрошенному времени, например:

```
http://example.com/elephants_dream.mp4?start=238.88
```

Это позволяет в любой момент времени выполнить произвольное позиционирование, а также начать воспроизведение с середины временной шкалы.

В форматах, основанных на H.264, метаданные, необходимые для поддержки позиционирования, хранятся в так называемом "moov-атоме". Это часть файла, которая содержит индексную информацию для всего файла.

До начала воспроизведения плееру необходимо прочитать метаданные. Для этого он отправляет специальный запрос с аргументом **start=0**. Многие кодирующие программы добавляют метаданные в конец файла. Это неоптимально для псевдо-стриминга, поскольку плееру потребуется загрузить файл целиком прежде чем начать воспроизведение. Если метаданные находятся в начале файла, Angie достаточно начать отправлять в ответ содержимое файла. Если же метаданные находятся в конце файла, потребуется прочитать весь файл и подготовить новый поток, в котором метаданные предшествуют медийным данным. Это требует дополнительного процессорного времени, памяти и дискового ввода/вывода, поэтому лучше заранее **подготовить** исходный файл для псевдо-стриминга, нежели делать это для каждого запроса.

Модуль также поддерживает аргумент **end** HTTP-запроса, задающий время окончания воспроизведения потока. Аргумент **end** задается совместно с аргументом **start** или самостоятельно:

```
http://example.com/elephants_dream.mp4?start=238.88&end=555.55
```

Для запроса с ненулевыми аргументами **start** или **end** Angie считывает из файла метаданные, готовит поток с запрошенным диапазоном и отправляет его клиенту. Это тоже требует дополнительных ресурсов, как указано выше.

Если аргумент **start** указывает на видеокادر, не являющийся ключевым, то начало такого видео может воспроизводиться с ошибками. В этом случае к запрашиваемому видео *могут* быть добавлены ближайший к точке **start** ключевой кадр и все промежуточные кадры между ними. При воспроизведении эти кадры будут скрыты при помощи edit-листа.

Если запрос, обрабатываемый этим модулем, не содержит аргументов **start** и **end**, дополнительные ресурсы не тратятся, а файл отправляется непосредственно как статический ресурс. Некоторые плееры также поддерживают запросы с указанием диапазона запрашиваемых байт (byte-range requests), для них этот модуль не требуется.

При сборке из исходного кода модуль не собирается по умолчанию; его необходимо включить с помощью параметра сборки **--with-http_mp4_module**. В пакетах и образах из наших репозиторий модуль включен в сборку.

Предупреждение

Если ранее использовался сторонний модуль **mp4**, следует его отключить.

Схожая поддержка псевдо-стриминга для FLV-файлов обеспечивается модулем *FLV*.

Пример конфигурации

```
location /video/ {
    mp4;
    mp4_buffer_size    1m;
    mp4_max_buffer_size 5m;
}
```

Директивы

mp4

<i>Синтаксис</i>	mp4;
По умолчанию	—
<i>Контекст</i>	location

Включает в содержащем location обработку этим модулем.

mp4_buffer_size

<i>Синтаксис</i>	mp4_buffer_size <i>размер</i> ;
По умолчанию	mp4_buffer_size 512K;
<i>Контекст</i>	http, server, location

Задаёт начальный размер буфера, используемого при обработке MP4-файлов.

mp4_max_buffer_size

<i>Синтаксис</i>	mp4_max_buffer_size <i>размер</i> ;
По умолчанию	mp4_max_buffer_size 10M;
<i>Контекст</i>	http, server, location

В ходе обработки метаданных может понадобиться буфер большего размера. Его размер не может превышать указанного, иначе Angie вернёт серверную ошибку 500 (Internal Server Error) и запишет в лог следующее сообщение:

```
"/some/movie/file.mp4" mp4 moov atom is too large: 12583268, you may want to increase
mp4_max_buffer_size
```

mp4_limit_rate

<i>Синтаксис</i>	mp4_limit_rate on off <i>коэффициент</i> ;
По умолчанию	mp4_limit_rate off;
<i>Контекст</i>	http, server, location

Ограничивает скорость передачи запрошенного MP4-файла клиенту. Предельное значение вычисляется умножением заданного *коэффициента* на средний битрейт файла.

- Специальное значение **off** отключает ограничение скорости.
- Специальное значение **on** соответствует *коэффициенту* 1.1.
- Начало действия ограничения регулируется значением *mp4_limit_rate_after*.

Запросы ограничиваются индивидуально: если клиент открыл два соединения, общая скорость удваивается. В связи с этим обратите внимание на *limit_conn* и сопутствующие директивы.

mp4_limit_rate_after

<i>Синтаксис</i>	mp4_limit_rate_after <i>время</i> ;
По умолчанию	mp4_limit_rate_after 60s;
<i>Контекст</i>	http, server, location

Задаёт (в пересчёте на *время воспроизведения*) объём медиаданных, после передачи которого скорость будет ограничена директивой *mp4_limit_rate*.

mp4_start_key_frame

<i>Синтаксис</i>	mp4_start_key_frame on off;
По умолчанию	mp4_start_key_frame off;
<i>Контекст</i>	http, server, location

Включает режим, в котором видео всегда начинается с ключевого видеокadra. Если аргумент start не указывает на ключевой кадр, то первоначальные кадры будут скрыты при помощи mp4 edit-листа. Edit-листы поддерживаются большинством плееров и браузеров включая Chrome, Safari, QuickTime и ffmpeg, частично поддерживаются в Firefox.

Prometheus

Собирает *статистику* Angie, исходя из определенных в конфигурации шаблонов, и возвращает сформированные на основании этих шаблонов метрики в *формате Prometheus*.

Предупреждение

Чтобы собирать статистику, включите в нужных контекстах зону разделяемой памяти с помощью:

- директивы *zone* в *http_upstream* или *stream_upstream*;
- директивы *status_zone*;
- параметра *status_zone* в директиве *resolver*.

Пример конфигурации

Три метрики для сбора статистики запросов к серверным зонам разделяемой памяти, объединенные в шаблон *custom* и опубликованные по пути */p8s*:

```
http {
    prometheus_template custom {
        'angie_http_server_zones_requests_total{zone="$1"}' $p8s_value
        path=~^/http/server_zones/([^/]+)/requests/total$
        type=counter;

        'angie_http_server_zones_requests_processing{zone="$1"}' $p8s_value
        path=~^/http/server_zones/([^/]+)/requests/processing$
    }
}
```

```

        type=gauge;

        'angie_http_server_zones_requests_discarded{zone="$1"}' $p8s_value
        path=~^/http/server_zones/([~/]+)/requests/discarded$
        type=counter;
    }

    # ...

    server {

        listen 80;

        location =/p8s {
            prometheus custom;
        }

        # ...

    }
}

```

В состав Angie входит вспомогательный файл `prometheus_all.conf`, куда включен набор общепотребимых метрик, сведенных в шаблон `all`:

Содержимое файла (Angie)

```

prometheus_template all {

angie_connections_accepted $p8s_value
    path=/connections/accepted
    type=counter
    'help=The total number of accepted client connections.';

angie_connections_dropped $p8s_value
    path=/connections/dropped
    type=counter
    'help=The total number of dropped client connections.';

angie_connections_active $p8s_value
    path=/connections/active
    type=gauge
    'help=The current number of active client connections.';

angie_connections_idle $p8s_value
    path=/connections/idle
    type=gauge
    'help=The current number of idle client connections.';

'angie_slabs_pages_used{zone="$1"}' $p8s_value
    path=~^/slabs/([~/]+)/pages/used$
    type=gauge
    'help=The number of currently used memory pages in a slab zone.';

'angie_slabs_pages_free{zone="$1"}' $p8s_value

```

```

path=~~/slabs/([~/]+)/pages/free$
type=gauge
'help=The number of currently free memory pages in a slab zone.';

'angie_slabs_pages_slots_used{zone="$1",size="$2"}' $p8s_value
path=~~/slabs/([~/]+)/slots/([~/]+)/used$
type=gauge
'help=The number of currently used memory slots of a specific size in a slab zone.
↪';

'angie_slabs_pages_slots_free{zone="$1",size="$2"}' $p8s_value
path=~~/slabs/([~/]+)/slots/([~/]+)/free$
type=gauge
'help=The number of currently free memory slots of a specific size in a slab zone.
↪';

'angie_slabs_pages_slots_reqs{zone="$1",size="$2"}' $p8s_value
path=~~/slabs/([~/]+)/slots/([~/]+)/reqs$
type=counter
'help=The total number of attempts to allocate a memory slot of a specific size
↪in a slab zone.';

'angie_slabs_pages_slots_fails{zone="$1",size="$2"}' $p8s_value
path=~~/slabs/([~/]+)/slots/([~/]+)/fails$
type=counter
'help=The number of unsuccessful attempts to allocate a memory slot of a specific
↪size in a slab zone.';

'angie_resolvers_queries{zone="$1",type="$2"}' $p8s_value
path=~~/resolvers/([~/]+)/queries/([~/]+)/$
type=counter
'help=The number of queries of a specific type to resolve in a resolver zone.';

'angie_resolvers_sent{zone="$1",type="$2"}' $p8s_value
path=~~/resolvers/([~/]+)/sent/([~/]+)/$
type=counter
'help=The number of sent DNS queries of a specific type to resolve in a resolver
↪zone.';

'angie_resolvers_responses{zone="$1",status="$2"}' $p8s_value
path=~~/resolvers/([~/]+)/responses/([~/]+)/$
type=counter
'help=The number of resolution results with a specific status in a resolver zone.
↪';

'angie_http_server_zones_ssl_handshaked{zone="$1"}' $p8s_value
path=~~/http/server_zones/([~/]+)/ssl/handshaked$
type=counter
'help=The total number of successful SSL handshakes in an HTTP server zone.';

'angie_http_server_zones_ssl_reuses{zone="$1"}' $p8s_value
path=~~/http/server_zones/([~/]+)/ssl/reuses$
type=counter
'help=The total number of session reuses during SSL handshakes in an HTTP server
↪';

```

```

↪zone.';

'angie_http_server_zones_ssl_timedout{zone="$1"}' $p8s_value
  path=~~/http/server_zones/([~/]+)/ssl/timedout$
  type=counter
  'help=The total number of timed-out SSL handshakes in an HTTP server zone.';

'angie_http_server_zones_ssl_failed{zone="$1"}' $p8s_value
  path=~~/http/server_zones/([~/]+)/ssl/failed$
  type=counter
  'help=The total number of failed SSL handshakes in an HTTP server zone.';

'angie_http_server_zones_requests_total{zone="$1"}' $p8s_value
  path=~~/http/server_zones/([~/]+)/requests/total$
  type=counter
  'help=The total number of client requests received in an HTTP server zone.';

'angie_http_server_zones_requests_processing{zone="$1"}' $p8s_value
  path=~~/http/server_zones/([~/]+)/requests/processing$
  type=gauge
  'help=The number of client requests currently being processed in an HTTP server_
↪zone.';

'angie_http_server_zones_requests_discarded{zone="$1"}' $p8s_value
  path=~~/http/server_zones/([~/]+)/requests/discarded$
  type=counter
  'help=The total number of client requests completed in an HTTP server zone_
↪without sending a response.';

'angie_http_server_zones_responses{zone="$1",code="$2"}' $p8s_value
  path=~~/http/server_zones/([~/]+)/responses/([~/]+)/$
  type=counter
  'help=The number of responses with a specific status in an HTTP server zone.';

'angie_http_server_zones_data_received{zone="$1"}' $p8s_value
  path=~~/http/server_zones/([~/]+)/data/received$
  type=counter
  'help=The total number of bytes received from clients in an HTTP server zone.';

'angie_http_server_zones_data_sent{zone="$1"}' $p8s_value
  path=~~/http/server_zones/([~/]+)/data/sent$
  type=counter
  'help=The total number of bytes sent to clients in an HTTP server zone.';

'angie_http_location_zones_requests_total{zone="$1"}' $p8s_value
  path=~~/http/location_zones/([~/]+)/requests/total$
  type=counter
  'help=The total number of client requests in an HTTP location zone.';

'angie_http_location_zones_requests_discarded{zone="$1"}' $p8s_value
  path=~~/http/location_zones/([~/]+)/requests/discarded$
  type=counter
  'help=The total number of client requests completed in an HTTP location zone_

```

```

↪without sending a response.';

'angie_http_location_zones_responses{zone="$1",code="$2"}' $p8s_value
  path=~^/http/location_zones/([~/]+)/responses/([~/]+)$
  type=counter
  'help=The number of responses with a specific status in an HTTP location zone.';

'angie_http_location_zones_data_received{zone="$1"}' $p8s_value
  path=~^/http/location_zones/([~/]+)/data/received$
  type=counter
  'help=The total number of bytes received from clients in an HTTP location zone.';

'angie_http_location_zones_data_sent{zone="$1"}' $p8s_value
  path=~^/http/location_zones/([~/]+)/data/sent$
  type=counter
  'help=The total number of bytes sent to clients in an HTTP location zone.';

'angie_http_upstreams_peers_state{upstream="$1",peer="$2"}' $p8st_all_ups_state
  path=~^/http/upstreams/([~/]+)/peers/([~/]+)/state$
  type=gauge
  'help=The current state of an upstream peer in "HTTP": 1 - up, 2 - down, 3 - ↪
↪unavailable, or 4 - recovering.';

'angie_http_upstreams_peers_selected_current{upstream="$1",peer="$2"}' $p8s_value
  path=~^/http/upstreams/([~/]+)/peers/([~/]+)/selected/current$
  type=gauge
  'help=The number of requests currently being processed by an upstream peer in
↪"HTTP".';

'angie_http_upstreams_peers_selected_total{upstream="$1",peer="$2"}' $p8s_value
  path=~^/http/upstreams/([~/]+)/peers/([~/]+)/selected/total$
  type=counter
  'help=The total number of attempts to use an upstream peer in "HTTP".';

'angie_http_upstreams_peers_responses{upstream="$1",peer="$2",code="$3"}' $p8s_value
  path=~^/http/upstreams/([~/]+)/peers/([~/]+)/responses/([~/]+)$
  type=counter
  'help=The number of responses with a specific status received from an upstream ↪
↪peer in "HTTP".';

'angie_http_upstreams_peers_data_sent{upstream="$1",peer="$2"}' $p8s_value
  path=~^/http/upstreams/([~/]+)/peers/([~/]+)/data/sent$
  type=counter
  'help=The total number of bytes sent to an upstream peer in "HTTP".';

'angie_http_upstreams_peers_data_received{upstream="$1",peer="$2"}' $p8s_value
  path=~^/http/upstreams/([~/]+)/peers/([~/]+)/data/received$
  type=counter
  'help=The total number of bytes received from an upstream peer in "HTTP".';

```

```
'angie_http_upstreams_peers_health_fails{upstream="$1",peer="$2"}' $p8s_value
  path=~~/http/upstreams/([~/]+)/peers/([~/]+)/health/fails$
  type=counter
  'help=The total number of unsuccessful attempts to communicate with an upstream
↪peer in "HTTP".';

'angie_http_upstreams_peers_health_unavailable{upstream="$1",peer="$2"}' $p8s_value
  path=~~/http/upstreams/([~/]+)/peers/([~/]+)/health/unavailable$
  type=counter
  'help=The number of times when an upstream peer in "HTTP" became "unavailable"
↪due to reaching the max_fails limit.';

'angie_http_upstreams_peers_health_downtime{upstream="$1",peer="$2"}' $p8s_value
  path=~~/http/upstreams/([~/]+)/peers/([~/]+)/health/downtime$
  type=counter
  'help=The total time (in milliseconds) that an upstream peer in "HTTP" was
↪"unavailable".';

'angie_http_upstreams_keepalive{upstream="$1"}' $p8s_value
  path=~~/http/upstreams/([~/]+)/keepalive$
  type=gauge
  'help=The number of currently cached keepalive connections for an HTTP upstream.';

'angie_http_caches_responses{zone="$1",status="$2"}' $p8s_value
  path=~~/http/caches/([~/]+)/([~/]+)/responses$
  type=counter
  'help=The total number of responses processed in an HTTP cache zone with a
↪specific cache status.';

'angie_http_caches_bytes{zone="$1",status="$2"}' $p8s_value
  path=~~/http/caches/([~/]+)/([~/]+)/bytes$
  type=counter
  'help=The total number of bytes processed in an HTTP cache zone with a specific
↪cache status.';

'angie_http_caches_responses_written{zone="$1",status="$2"}' $p8s_value
  path=~~/http/caches/([~/]+)/([~/]+)/responses_written$
  type=counter
  'help=The total number of responses written to an HTTP cache zone with a specific
↪cache status.';

'angie_http_caches_bytes_written{zone="$1",status="$2"}' $p8s_value
  path=~~/http/caches/([~/]+)/([~/]+)/bytes_written$
  type=counter
  'help=The total number of bytes written to an HTTP cache zone with a specific
↪cache status.';

'angie_http_caches_size{zone="$1"}' $p8s_value
  path=~~/http/caches/([~/]+)/size$
  type=gauge
  'help=The current size (in bytes) of cached responses in an HTTP cache zone.';

'angie_http_caches_shards_size{zone="$1",path="$2"}' $p8s_value
```

```

path=~http/caches/([~/]+)/shards/([~/]+)/size$
type=gauge
'help=The current size (in bytes) of cached responses in a shard path of an HTTP
↪cache zone.';

'angie_http_limit_conns{zone="$1",status="$2"}' $p8s_value
path=~http/limit_conns/([~/]+)/([~/]+)/$
type=counter
'help=The number of requests processed by an HTTP limit_conn zone with a specific
↪result.';

'angie_http_limit_reqs{zone="$1",status="$2"}' $p8s_value
path=~http/limit_reqs/([~/]+)/([~/]+)/$
type=counter
'help=The number of requests processed by an HTTP limit_reqs zone with a specific
↪result.';

'angie_stream_server_zones_ssl_handshaked{zone="$1"}' $p8s_value
path=~stream/server_zones/([~/]+)/ssl/handshaked$
type=counter
'help=The total number of successful SSL handshakes in a stream server zone.';

'angie_stream_server_zones_ssl_reuses{zone="$1"}' $p8s_value
path=~stream/server_zones/([~/]+)/ssl/reuses$
type=counter
'help=The total number of session reuses during SSL handshakes in a stream server
↪zone.';

'angie_stream_server_zones_ssl_timedout{zone="$1"}' $p8s_value
path=~stream/server_zones/([~/]+)/ssl/timedout$
type=counter
'help=The total number of timed-out SSL handshakes in a stream server zone.';

'angie_stream_server_zones_ssl_failed{zone="$1"}' $p8s_value
path=~stream/server_zones/([~/]+)/ssl/failed$
type=counter
'help=The total number of failed SSL handshakes in a stream server zone.';

'angie_stream_server_zones_connections_total{zone="$1"}' $p8s_value
path=~stream/server_zones/([~/]+)/connections/total$
type=counter
'help=The total number of client connections received in a stream server zone.';

'angie_stream_server_zones_connections_processing{zone="$1"}' $p8s_value
path=~stream/server_zones/([~/]+)/connections/processing$
type=gauge
'help=The number of client connections currently being processed in a stream
↪server zone.';

'angie_stream_server_zones_connections_discarded{zone="$1"}' $p8s_value
path=~stream/server_zones/([~/]+)/connections/discarded$
type=counter
'help=The total number of client connections completed in a stream server zone
↪without establishing a session.';

```

```
'angie_stream_server_zones_connections_passed{zone="$1"}' $p8s_value
  path=~~/stream/server_zones/([~/]+)/connections/passed$
  type=counter
  'help=The total number of client connections in a stream server zone passed for
↳ handling to a different listening socket.';

'angie_stream_server_zones_sessions{zone="$1",status="$2"}' $p8s_value
  path=~~/stream/server_zones/([~/]+)/sessions/([~/]+)/$
  type=counter
  'help=The number of sessions finished with a specific status in a stream server
↳ zone.';

'angie_stream_server_zones_data_received{zone="$1"}' $p8s_value
  path=~~/stream/server_zones/([~/]+)/data/received$
  type=counter
  'help=The total number of bytes received from clients in a stream server zone.';

'angie_stream_server_zones_data_sent{zone="$1"}' $p8s_value
  path=~~/stream/server_zones/([~/]+)/data/sent$
  type=counter
  'help=The total number of bytes sent to clients in a stream server zone.';

'angie_stream_upstreams_peers_state{upstream="$1",peer="$2"}' $p8st_all_ups_state
  path=~~/stream/upstreams/([~/]+)/peers/([~/]+)/state$
  type=gauge
  'help=The current state of an upstream peer in "stream": 1 - up, 2 - down, 3 -
↳ unavailable, or 4 - recovering.';

'angie_stream_upstreams_peers_selected_current{upstream="$1",peer="$2"}' $p8s_value
  path=~~/stream/upstreams/([~/]+)/peers/([~/]+)/selected/current$
  type=gauge
  'help=The number of sessions currently being processed by an upstream peer in
↳ "stream".';

'angie_stream_upstreams_peers_selected_total{upstream="$1",peer="$2"}' $p8s_value
  path=~~/stream/upstreams/([~/]+)/peers/([~/]+)/selected/total$
  type=counter
  'help=The total number of attempts to use an upstream peer in "stream".';

'angie_stream_upstreams_peers_data_sent{upstream="$1",peer="$2"}' $p8s_value
  path=~~/stream/upstreams/([~/]+)/peers/([~/]+)/data/sent$
  type=counter
  'help=The total number of bytes sent to an upstream peer in "stream".';

'angie_stream_upstreams_peers_data_received{upstream="$1",peer="$2"}' $p8s_value
  path=~~/stream/upstreams/([~/]+)/peers/([~/]+)/data/received$
  type=counter
  'help=The total number of bytes received from an upstream peer in "stream".';

'angie_stream_upstreams_peers_health_fails{upstream="$1",peer="$2"}' $p8s_value
```



```

    path=~~/stream/upstreams/([^/]+)/peers/([^/]+)/health/fails$
    type=counter
    'help=The total number of unsuccessful attempts to communicate with an upstream
↪peer in "stream".';

'angie_stream_upstreams_peers_health_unavailable{upstream="$1",peer="$2"}' $p8s_value
    path=~~/stream/upstreams/([^/]+)/peers/([^/]+)/health/unavailable$
    type=counter
    'help=The number of times when an upstream peer in "stream" became "unavailable"
↪due to reaching the max_fails limit.';

'angie_stream_upstreams_peers_health_downtime{upstream="$1",peer="$2"}' $p8s_value
    path=~~/stream/upstreams/([^/]+)/peers/([^/]+)/health/downtime$
    type=counter
    'help=The total time (in milliseconds) that an upstream peer in "stream" was
↪"unavailable".';
}

map $p8s_value $p8st_all_ups_state {
    volatile;
    "up"          1;
    "down"        2;
    "unavailable" 3;
    "recovering"  4;
#   "unhealthy"   5;
#   "checking"    6;
#   "draining"    7;
    "busy"        8;
    default       0;
}

```

Содержимое файла (Angie PRO)

```

prometheus_template all {

angie_connections_accepted $p8s_value
    path=/connections/accepted
    type=counter
    'help=The total number of accepted client connections.';

angie_connections_dropped $p8s_value
    path=/connections/dropped
    type=counter
    'help=The total number of dropped client connections.';

angie_connections_active $p8s_value
    path=/connections/active
    type=gauge
    'help=The current number of active client connections.';

angie_connections_idle $p8s_value
    path=/connections/idle
    type=gauge
    'help=The current number of idle client connections.';
}

```

```
'angie_slabs_pages_used{zone="$1"}' $p8s_value
  path=~~/slabs/([~/]+)/pages/used$
  type=gauge
  'help=The number of currently used memory pages in a slab zone.';

'angie_slabs_pages_free{zone="$1"}' $p8s_value
  path=~~/slabs/([~/]+)/pages/free$
  type=gauge
  'help=The number of currently free memory pages in a slab zone.';

'angie_slabs_pages_slots_used{zone="$1",size="$2"}' $p8s_value
  path=~~/slabs/([~/]+)/slots/([~/]+)/used$
  type=gauge
  'help=The number of currently used memory slots of a specific size in a slab zone.
→';

'angie_slabs_pages_slots_free{zone="$1",size="$2"}' $p8s_value
  path=~~/slabs/([~/]+)/slots/([~/]+)/free$
  type=gauge
  'help=The number of currently free memory slots of a specific size in a slab zone.
→';

'angie_slabs_pages_slots_reqs{zone="$1",size="$2"}' $p8s_value
  path=~~/slabs/([~/]+)/slots/([~/]+)/reqs$
  type=counter
  'help=The total number of attempts to allocate a memory slot of a specific size
→in a slab zone.';

'angie_slabs_pages_slots_fails{zone="$1",size="$2"}' $p8s_value
  path=~~/slabs/([~/]+)/slots/([~/]+)/fails$
  type=counter
  'help=The number of unsuccessful attempts to allocate a memory slot of a specific
→size in a slab zone.';

'angie_resolvers_queries{zone="$1",type="$2"}' $p8s_value
  path=~~/resolvers/([~/]+)/queries/([~/]+)/$
  type=counter
  'help=The number of queries of a specific type to resolve in a resolver zone.';

'angie_resolvers_sent{zone="$1",type="$2"}' $p8s_value
  path=~~/resolvers/([~/]+)/sent/([~/]+)/$
  type=counter
  'help=The number of sent DNS queries of a specific type to resolve in a resolver
→zone.';

'angie_resolvers_responses{zone="$1",status="$2"}' $p8s_value
  path=~~/resolvers/([~/]+)/responses/([~/]+)/$
  type=counter
  'help=The number of resolution results with a specific status in a resolver zone.
→';

'angie_http_server_zones_ssl_handshaked{zone="$1"}' $p8s_value
  path=~~/http/server_zones/([~/]+)/ssl/handshaked$
  type=counter
```

```

    'help=The total number of successful SSL handshakes in an HTTP server zone.';

'angie_http_server_zones_ssl_reuses{zone="$1"}' $p8s_value
    path=~~/http/server_zones/([~/]+)/ssl/reuses$
    type=counter
    'help=The total number of session reuses during SSL handshakes in an HTTP server_
↪zone.';

'angie_http_server_zones_ssl_timedout{zone="$1"}' $p8s_value
    path=~~/http/server_zones/([~/]+)/ssl/timedout$
    type=counter
    'help=The total number of timed-out SSL handshakes in an HTTP server zone.';

'angie_http_server_zones_ssl_failed{zone="$1"}' $p8s_value
    path=~~/http/server_zones/([~/]+)/ssl/failed$
    type=counter
    'help=The total number of failed SSL handshakes in an HTTP server zone.';

'angie_http_server_zones_requests_total{zone="$1"}' $p8s_value
    path=~~/http/server_zones/([~/]+)/requests/total$
    type=counter
    'help=The total number of client requests received in an HTTP server zone.';

'angie_http_server_zones_requests_processing{zone="$1"}' $p8s_value
    path=~~/http/server_zones/([~/]+)/requests/processing$
    type=gauge
    'help=The number of client requests currently being processed in an HTTP server_
↪zone.';

'angie_http_server_zones_requests_discarded{zone="$1"}' $p8s_value
    path=~~/http/server_zones/([~/]+)/requests/discarded$
    type=counter
    'help=The total number of client requests completed in an HTTP server zone_
↪without sending a response.';

'angie_http_server_zones_responses{zone="$1",code="$2"}' $p8s_value
    path=~~/http/server_zones/([~/]+)/responses/([~/]+)/$
    type=counter
    'help=The number of responses with a specific status in an HTTP server zone.';

'angie_http_server_zones_data_received{zone="$1"}' $p8s_value
    path=~~/http/server_zones/([~/]+)/data/received$
    type=counter
    'help=The total number of bytes received from clients in an HTTP server zone.';

'angie_http_server_zones_data_sent{zone="$1"}' $p8s_value
    path=~~/http/server_zones/([~/]+)/data/sent$
    type=counter
    'help=The total number of bytes sent to clients in an HTTP server zone.';

'angie_http_location_zones_requests_total{zone="$1"}' $p8s_value
    path=~~/http/location_zones/([~/]+)/requests/total$
    type=counter

```

```
'help=The total number of client requests in an HTTP location zone.';

'angie_http_location_zones_requests_discarded{zone="$1"}' $p8s_value
path=~^/http/location_zones/([~/]+)/requests/discarded$
type=counter
'help=The total number of client requests completed in an HTTP location zone,
↳without sending a response.';

'angie_http_location_zones_responses{zone="$1",code="$2"}' $p8s_value
path=~^/http/location_zones/([~/]+)/responses/([~/]+)/$
type=counter
'help=The number of responses with a specific status in an HTTP location zone.';

'angie_http_location_zones_data_received{zone="$1"}' $p8s_value
path=~^/http/location_zones/([~/]+)/data/received$
type=counter
'help=The total number of bytes received from clients in an HTTP location zone.';

'angie_http_location_zones_data_sent{zone="$1"}' $p8s_value
path=~^/http/location_zones/([~/]+)/data/sent$
type=counter
'help=The total number of bytes sent to clients in an HTTP location zone.';

'angie_http_upstreams_peers_backup{upstream="$1",peer="$2"}' $p8st_all_ups_backup
path=~^/http/upstreams/([~/]+)/peers/([~/]+)/backup$
type=gauge
'help=The HTTP upstream peer backup group level.';

'angie_http_upstreams_peers_state{upstream="$1",peer="$2"}' $p8st_all_ups_state
path=~^/http/upstreams/([~/]+)/peers/([~/]+)/state$
type=gauge
'help=The current state of an upstream peer in "HTTP": 1 - up, 2 - down, 3 -
↳unavailable, 4 - recovering, 5 - unhealthy, 6 - checking, or 7 - draining.';

'angie_http_upstreams_peers_selected_current{upstream="$1",peer="$2"}' $p8s_value
path=~^/http/upstreams/([~/]+)/peers/([~/]+)/selected/current$
type=gauge
'help=The number of requests currently being processed by an upstream peer in
↳"HTTP".';

'angie_http_upstreams_peers_selected_total{upstream="$1",peer="$2"}' $p8s_value
path=~^/http/upstreams/([~/]+)/peers/([~/]+)/selected/total$
type=counter
'help=The total number of attempts to use an upstream peer in "HTTP".';

'angie_http_upstreams_peers_responses{upstream="$1",peer="$2",code="$3"}' $p8s_value
path=~^/http/upstreams/([~/]+)/peers/([~/]+)/responses/([~/]+)/$
type=counter
'help=The number of responses with a specific status received from an upstream
↳peer in "HTTP".';
```

```
'angie_http_upstreams_peers_data_sent{upstream="$1",peer="$2"}' $p8s_value
  path=~~/http/upstreams/([~/]+)/peers/([~/]+)/data/sent$
  type=counter
  'help=The total number of bytes sent to an upstream peer in "HTTP".';

'angie_http_upstreams_peers_data_received{upstream="$1",peer="$2"}' $p8s_value
  path=~~/http/upstreams/([~/]+)/peers/([~/]+)/data/received$
  type=counter
  'help=The total number of bytes received from an upstream peer in "HTTP".';

'angie_http_upstreams_peers_health_fails{upstream="$1",peer="$2"}' $p8s_value
  path=~~/http/upstreams/([~/]+)/peers/([~/]+)/health/fails$
  type=counter
  'help=The total number of unsuccessful attempts to communicate with an upstream
↪peer in "HTTP".';

'angie_http_upstreams_peers_health_unavailable{upstream="$1",peer="$2"}' $p8s_value
  path=~~/http/upstreams/([~/]+)/peers/([~/]+)/health/unavailable$
  type=counter
  'help=The number of times when an upstream peer in "HTTP" became "unavailable"
↪due to reaching the max_fails limit.';

'angie_http_upstreams_peers_health_downtime{upstream="$1",peer="$2"}' $p8s_value
  path=~~/http/upstreams/([~/]+)/peers/([~/]+)/health/downtime$
  type=counter
  'help=The total time (in milliseconds) that an upstream peer in "HTTP" was
↪"unavailable".';

'angie_http_upstreams_peers_health_header_time{upstream="$1",peer="$2"}' $p8s_value
  path=~~/http/upstreams/([~/]+)/peers/([~/]+)/health/header_time$
  type=gauge
  'help=Average time (in milliseconds) to receive the response headers from an
↪upstream peer in "HTTP".';

'angie_http_upstreams_peers_health_response_time{upstream="$1",peer="$2"}' $p8s_value
  path=~~/http/upstreams/([~/]+)/peers/([~/]+)/health/response_time$
  type=gauge
  'help=Average time (in milliseconds) to receive the complete response from an
↪upstream peer in "HTTP".';

'angie_http_upstreams_peers_health_probes_count{upstream="$1",peer="$2"}' $p8s_value
  path=~~/http/upstreams/([~/]+)/peers/([~/]+)/health/probes/count$
  type=counter
  'help=The total number of probes for this peer.';

'angie_http_upstreams_peers_health_probes_fails{upstream="$1",peer="$2"}' $p8s_value
  path=~~/http/upstreams/([~/]+)/peers/([~/]+)/health/probes/fails$
  type=counter
  'help=The total number of failed probes for this peer.';

'angie_http_upstreams_keepalive{upstream="$1"}' $p8s_value
  path=~~/http/upstreams/([~/]+)/keepalive$
  type=gauge
  'help=The number of currently cached keepalive connections for an HTTP upstream.';
```

```
'angie_http_upstreams_backup_switch_active{upstream="$1"}' $p8s_value
  path=~~/http/upstreams/([~/]+)/backup_switch/active$
  type=gauge
  'help=The currently active HTTP upstream servers backup group level.';

'angie_http_upstreams_queue_queued{upstream="$1"}' $p8s_value
  path=~~/http/upstreams/([~/]+)/queue/queued$
  type=counter
  'help=The total number of queued requests for an HTTP upstream.';

'angie_http_upstreams_queue_waiting{upstream="$1"}' $p8s_value
  path=~~/http/upstreams/([~/]+)/queue/waiting$
  type=gauge
  'help=The number of requests currently waiting in an HTTP upstream queue.';

'angie_http_upstreams_queue_dropped{upstream="$1"}' $p8s_value
  path=~~/http/upstreams/([~/]+)/queue/dropped$
  type=counter
  'help=The total number of requests dropped from an HTTP upstream queue because
↳the client had prematurely closed the connection.';

'angie_http_upstreams_queue_timedout{upstream="$1"}' $p8s_value
  path=~~/http/upstreams/([~/]+)/queue/timedout$
  type=counter
  'help=The total number of requests timed out from an HTTP upstream queue.';

'angie_http_upstreams_queue_overflows{upstream="$1"}' $p8s_value
  path=~~/http/upstreams/([~/]+)/queue/overflows$
  type=counter
  'help=The total number of requests rejected by an HTTP upstream queue because the
↳size limit had been reached.';

'angie_http_caches_responses{zone="$1",status="$2"}' $p8s_value
  path=~~/http/caches/([~/]+)/([~/]+)/responses$
  type=counter
  'help=The total number of responses processed in an HTTP cache zone with a
↳specific cache status.';

'angie_http_caches_bytes{zone="$1",status="$2"}' $p8s_value
  path=~~/http/caches/([~/]+)/([~/]+)/bytes$
  type=counter
  'help=The total number of bytes processed in an HTTP cache zone with a specific
↳cache status.';

'angie_http_caches_responses_written{zone="$1",status="$2"}' $p8s_value
  path=~~/http/caches/([~/]+)/([~/]+)/responses_written$
  type=counter
  'help=The total number of responses written to an HTTP cache zone with a specific
↳cache status.';

'angie_http_caches_bytes_written{zone="$1",status="$2"}' $p8s_value
  path=~~/http/caches/([~/]+)/([~/]+)/bytes_written$
  type=counter
```

```
'help=The total number of bytes written to an HTTP cache zone with a specific
↪cache status.';

'angie_http_caches_size{zone="$1"}' $p8s_value
  path=~~/http/caches/([~/]+)/size$
  type=gauge
  'help=The current size (in bytes) of cached responses in an HTTP cache zone.';

'angie_http_caches_shards_size{zone="$1",path="$2"}' $p8s_value
  path=~~/http/caches/([~/]+)/shards/([~/]+)/size$
  type=gauge
  'help=The current size (in bytes) of cached responses in a shard path of an HTTP
↪cache zone.';

'angie_http_limit_conns{zone="$1",status="$2"}' $p8s_value
  path=~~/http/limit_conns/([~/]+)/([~//])$
  type=counter
  'help=The number of requests processed by an HTTP limit_conn zone with a specific
↪result.';

'angie_http_limit_reqs{zone="$1",status="$2"}' $p8s_value
  path=~~/http/limit_reqs/([~/]+)/([~//])$
  type=counter
  'help=The number of requests processed by an HTTP limit_reqs zone with a specific
↪result.';

'angie_stream_server_zones_ssl_handshaked{zone="$1"}' $p8s_value
  path=~~/stream/server_zones/([~/]+)/ssl/handshaked$
  type=counter
  'help=The total number of successful SSL handshakes in a stream server zone.';

'angie_stream_server_zones_ssl_reuses{zone="$1"}' $p8s_value
  path=~~/stream/server_zones/([~/]+)/ssl/reuses$
  type=counter
  'help=The total number of session reuses during SSL handshakes in a stream server
↪zone.';

'angie_stream_server_zones_ssl_timedout{zone="$1"}' $p8s_value
  path=~~/stream/server_zones/([~/]+)/ssl/timedout$
  type=counter
  'help=The total number of timed-out SSL handshakes in a stream server zone.';

'angie_stream_server_zones_ssl_failed{zone="$1"}' $p8s_value
  path=~~/stream/server_zones/([~/]+)/ssl/failed$
  type=counter
  'help=The total number of failed SSL handshakes in a stream server zone.';

'angie_stream_server_zones_connections_total{zone="$1"}' $p8s_value
  path=~~/stream/server_zones/([~/]+)/connections/total$
  type=counter
  'help=The total number of client connections received in a stream server zone.';
```

```
'angie_stream_server_zones_connections_processing{zone="$1"}' $p8s_value
  path=~~/stream/server_zones/([~/]+)/connections/processing$
  type=gauge
  'help=The number of client connections currently being processed in a stream
↪server zone.';

'angie_stream_server_zones_connections_discarded{zone="$1"}' $p8s_value
  path=~~/stream/server_zones/([~/]+)/connections/discarded$
  type=counter
  'help=The total number of client connections completed in a stream server zone
↪without establishing a session.';

'angie_stream_server_zones_connections_passed{zone="$1"}' $p8s_value
  path=~~/stream/server_zones/([~/]+)/connections/passed$
  type=counter
  'help=The total number of client connections in a stream server zone passed for
↪handling to a different listening socket.';

'angie_stream_server_zones_sessions{zone="$1",status="$2"}' $p8s_value
  path=~~/stream/server_zones/([~/]+)/sessions/([~/]+)/$
  type=counter
  'help=The number of sessions finished with a specific status in a stream server
↪zone.';

'angie_stream_server_zones_data_received{zone="$1"}' $p8s_value
  path=~~/stream/server_zones/([~/]+)/data/received$
  type=counter
  'help=The total number of bytes received from clients in a stream server zone.';

'angie_stream_server_zones_data_sent{zone="$1"}' $p8s_value
  path=~~/stream/server_zones/([~/]+)/data/sent$
  type=counter
  'help=The total number of bytes sent to clients in a stream server zone.';

'angie_stream_upstreams_peers_backup{upstream="$1",peer="$2"}' $p8st_all_ups_backup
  path=~~/stream/upstreams/([~/]+)/peers/([~/]+)/backup$
  type=gauge
  'help=The "stream" upstream peer backup group level.';

'angie_stream_upstreams_peers_state{upstream="$1",peer="$2"}' $p8st_all_ups_state
  path=~~/stream/upstreams/([~/]+)/peers/([~/]+)/state$
  type=gauge
  'help=The current state of an upstream peer in "stream": 1 - up, 2 - down, 3 -
↪unavailable, 4 - recovering, 5 - unhealthy, 6 - checking, or 7 - draining.';

'angie_stream_upstreams_peers_selected_current{upstream="$1",peer="$2"}' $p8s_value
  path=~~/stream/upstreams/([~/]+)/peers/([~/]+)/selected/current$
  type=gauge
  'help=The number of sessions currently being processed by an upstream peer in
↪"stream".';

'angie_stream_upstreams_peers_selected_total{upstream="$1",peer="$2"}' $p8s_value
```



```

path=~~/stream/upstreams/([~/]+)/peers/([~/]+)/selected/total$
type=counter
'help=The total number of attempts to use an upstream peer in "stream".';

'angie_stream_upstreams_peers_data_sent{upstream="$1",peer="$2"}' $p8s_value
path=~~/stream/upstreams/([~/]+)/peers/([~/]+)/data/sent$
type=counter
'help=The total number of bytes sent to an upstream peer in "stream".';

'angie_stream_upstreams_peers_data_received{upstream="$1",peer="$2"}' $p8s_value
path=~~/stream/upstreams/([~/]+)/peers/([~/]+)/data/received$
type=counter
'help=The total number of bytes received from an upstream peer in "stream".';

'angie_stream_upstreams_peers_data_pkt_sent{upstream="$1",peer="$2"}' $p8s_value
path=~~/stream/upstreams/([~/]+)/peers/([~/]+)/data/pkt_sent$
type=counter
'help=The total number of packets sent to an upstream peer in "stream".';

'angie_stream_upstreams_peers_data_pkt_received{upstream="$1",peer="$2"}' $p8s_value
path=~~/stream/upstreams/([~/]+)/peers/([~/]+)/data/pkt_received$
type=counter
'help=The total number of packets received from an upstream peer in "stream".';

'angie_stream_upstreams_peers_health_fails{upstream="$1",peer="$2"}' $p8s_value
path=~~/stream/upstreams/([~/]+)/peers/([~/]+)/health/fails$
type=counter
'help=The total number of unsuccessful attempts to communicate with an upstream
↪peer in "stream".';

'angie_stream_upstreams_peers_health_unavailable{upstream="$1",peer="$2"}' $p8s_value
path=~~/stream/upstreams/([~/]+)/peers/([~/]+)/health/unavailable$
type=counter
'help=The number of times when an upstream peer in "stream" became "unavailable"
↪due to reaching the max_fails limit.';

'angie_stream_upstreams_peers_health_downtime{upstream="$1",peer="$2"}' $p8s_value
path=~~/stream/upstreams/([~/]+)/peers/([~/]+)/health/downtime$
type=counter
'help=The total time (in milliseconds) that an upstream peer in "stream" was
↪"unavailable".';

'angie_stream_upstreams_peers_health_connect_time{upstream="$1",peer="$2"}' $p8s_value
path=~~/stream/upstreams/([~/]+)/peers/([~/]+)/health/connect_time$
type=gauge
'help=Average time (in milliseconds) to connect to an upstream peer in "stream".';

'angie_stream_upstreams_peers_health_first_byte_time{upstream="$1",peer="$2"}' $p8s_
↪value
path=~~/stream/upstreams/([~/]+)/peers/([~/]+)/health/first_byte_time$
type=gauge
'help=Average time (in milliseconds) to receive the first byte from an upstream
↪peer in "stream".';

'angie_stream_upstreams_peers_health_last_byte_time{upstream="$1",peer="$2"}' $p8s_

```

```

↪value
    path=~~/stream/upstreams/([~/]+)/peers/([~/]+)/health/last_byte_time$
    type=gauge
    'help=Average time (in milliseconds) of the whole communication session with an
↪upstream peer in "stream".';

'angie_stream_upstreams_peers_health_probes_count{upstream="$1",peer="$2"}' $p8s_value
    path=~~/stream/upstreams/([~/]+)/peers/([~/]+)/health/probes/count$
    type=counter
    'help=The total number of probes for this peer.';

'angie_stream_upstreams_peers_health_probes_fails{upstream="$1",peer="$2"}' $p8s_value
    path=~~/stream/upstreams/([~/]+)/peers/([~/]+)/health/probes/fails$
    type=counter
    'help=The total number of failed probes for this peer.';

'angie_stream_upstreams_backup_switch_active{upstream="$1"}' $p8s_value
    path=~~/stream/upstreams/([~/]+)/backup_switch/active$
    type=gauge
    'help=The currently active "stream" upstream servers backup group level.';
}

map $p8s_value $p8st_all_ups_state {
    volatile;
    "up"          1;
    "down"        2;
    "unavailable" 3;
    "recovering"  4;
    "unhealthy"   5;
    "checking"    6;
    "draining"    7;
    "busy"        8;
    default       0;
}

map $p8s_value $p8st_all_ups_backup {
    volatile;
    "false"       0;
    "true"        1;
    default       $p8s_value;
}

```

Его использование:

```

http {

    include prometheus_all.conf;

    # ...

    server {

        listen 80;

        location =/p8s {
            prometheus all;
        }
    }
}

```

```
    }

    # ...

}
}
```

```
$ curl localhost/p8s

# Angie Prometheus template "all"
...
```

Директивы

prometheus

Синтаксис	<code>prometheus имя_шаблона;</code>
По умолчанию	—
Контекст	<code>location</code>

Указывает для контекста `location` шаблон-обработчик, заданный директивой `prometheus template`. При запросе этот `location` вычисляет и возвращает метрики шаблона в формате Prometheus.

```
location =/p8s {
    prometheus custom;
}
```

```
$ curl localhost/p8s

# Angie Prometheus template "custom"
...
```

prometheus_template

Синтаксис	<code>prometheus_template имя_шаблона { ... }</code>
По умолчанию	—
Контекст	<code>http</code>

Определяет именованный шаблон метрик, собираемых и экспортируемых Angie, для использования с директивой `prometheus`.

Примечание

В состав Angie также входит готовый шаблон `all`, куда включен набор наиболее общепотребимых метрик.

Может содержать произвольное число определений метрик, каждая из которых имеет следующую структуру: `<имя_метрики> <переменная> [path=<строка_сопоставления>] [type=<тип>] [help=<справка>]`.

имя_метрики	Задаёт имя метрики, под которым она будет добавлена в формате Prometheus в ответ на запрос. Может содержать необязательную часть с метками (...), например: <pre>http_requests_total{method="\$1",code="\$2"}</pre> В значениях меток можно использовать переменные Angie; если <i>строка_сопоставления</i> задана регулярным выражением, то можно также использовать определённые в этом выражении группы захвата. Такие переменные и группы вычисляются при получении значения метрики, которое задаёт <i>переменная</i>.
переменная	Задаёт имя переменной, которая будет вычислена и добавлена как значение метрики в ответ на запрос. Если переменной нет или результат вычисления пуст (""), метрика не добавляется.

Метрика вычисляется со значением, которое задаёт *переменная*; при успехе вычисления метрика добавляется в ответ, например:

```
'angie_time{version="$angie_version"}' $msec;
```

```
$ curl localhost/p8s
```

```
angie_time{version="|version|"} 1695119820.562
```

path=строка_сопо	Сопоставляется со всеми конечными путями метрик в поддереве /status API-интерфейса Angie, позволяя добавить в ответ сразу несколько экземпляров метрики.

При сопоставлении пути берутся с начальной косой чертой, но без конечной, например `/angie/generation`; при этом регистр символов не учитывается. Есть два способа сопоставления:

path=точное_соот	Проверяется посимвольным сравнением.
path=~регулярное	Проверяется при помощи библиотеки PCRE; может задавать группы захвата для использования в метках поля <i>имя метрики</i> .

Если *строка_сопоставления* соответствует какому-либо пути, то значение метрики Angie по нему заносится в переменную `$p8s_value`, которую при заданном `path=` можно использовать в поле *переменная*.

В случае регулярного выражения совпадающих путей может быть несколько; метрика добавляется в ответ для *каждого* совпадения. В сочетании с группами захвата это позволяет получить ряд метрик с одним именем и разными метками, например:

```
'angie_slabs_slots_free{zone="$1",size="$2"}' $p8s_value
path=~^/slabs/([~/]+)/slots/([~/]+)/free$;
```

Это определение добавляет метрики для всех зон и всех размеров, которые есть сейчас в конфигурации:

```
angie_slabs_slots_free{zone="one",size="8"} 502
angie_slabs_slots_free{zone="one",size="16"} 249
angie_slabs_slots_free{zone="one",size="32"} 122
angie_slabs_slots_free{zone="one",size="128"} 22
```

```
angie_slabs_slots_free{zone="one",size="512"} 4
angie_slabs_slots_free{zone="two",size="8"} 311
...
```

Если совпадений нет (при любом способе сопоставления), то метрика не добавляется.

Примечание

Параметр `path=` доступен только при сборке Angie с модулем *API*.

<code>type=тип,</code> <code>help=справка</code>	Задают соответственно тип и справочную строку метрики в формате Prometheus , которые добавляются с ней в ответ без изменений и проверок.

Встроенные переменные

В модуле `http_prometheus` есть встроенная переменная, получающая значение при сопоставлении путей метрик из раздела */status* API-интерфейса Angie с параметром *строка_сопоставления* метрик, заданных директивой *prometheus_template*.

`$p8s_value`

Если *строка_сопоставления* метрики, заданной в *prometheus_template*, соответствует какому-либо пути, то значение метрики Angie, находящееся по этому пути, заносится в переменную `$p8s_value`. Она предназначена для использования в поле *переменная* в определениях метрик, которые вычисляются на основе параметра `path=`.

Значения метрик Angie, заносимые в переменную `$p8s_value`, не всегда отвечают потребностям формата Prometheus. Тогда можно воспользоваться директивой *map*, например для преобразования строк в числа:

```
map $p8s_value $ups_state_n {
    up           0;
    unavailable  1;
    down         2;
    default      3;
}

prometheus_template main {
    'angie_http_upstreams_state{upstream="$1",peer="$2"}' $ups_state_n
    path=~^/http/upstreams/([~/]+)/peers/([~/]+)/state$;
}
```

Если у метрики Angie логическое значение, то есть `true` или `false`, переменная получает значение "1" или "0" соответственно; если значение метрики — `null`, то переменная будет равна "(null)". Для дат используется целочисленный формат эпохи UNIX.

Proxy

Позволяет передавать запросы другому (проксируемому) серверу.

Пример конфигурации

```
location / {
    proxy_pass      http://localhost:8000;
    proxy_set_header Host      $host;
    proxy_set_header X-Real-IP $remote_addr;

    proxy_cache      cache_zone;
    proxy_cache_valid 200 302 10m;
    proxy_cache_valid 404      1m;
}
```

Директивы

proxy_bind

<i>Синтаксис</i>	<code>proxy_bind адрес [transparent] off;</code>
По умолчанию	—
<i>Контекст</i>	http, server, location

Задаёт локальный IP-адрес с необязательным портом, который будет использоваться в исходящих соединениях с проксируемым сервером. В значении параметра допустимо использование переменных. Специальное значение **off** отменяет действие унаследованной с предыдущего уровня конфигурации директивы *proxy_bind*, позволяя системе самостоятельно выбирать локальный IP-адрес и порт.

Параметр **transparent** позволяет задать нелокальный IP-адрес, который будет использоваться в исходящих соединениях с проксируемым сервером, например, реальный IP-адрес клиента:

```
proxy_bind $remote_addr transparent;
```

Для работы параметра обычно требуется запустить рабочие процессы Angie с привилегиями *superпользователя*. В Linux это не требуется, так как если указан параметр *transparent*, то рабочие процессы наследуют *capability CAP_NET_RAW* из главного процесса.

Примечание

Необходимо настроить таблицу маршрутизации ядра для перехвата сетевого трафика с проксируемого сервера.

proxy_buffer_size

<i>Синтаксис</i>	proxy_buffer_size <i>размер</i> ;
По умолчанию	proxy_buffer_size 4k 8k;
<i>Контекст</i>	http, server, location

Задаёт размер буфера, в который будет читаться первая часть ответа, получаемого от проксируемого сервера. В этой части ответа обычно находится небольшой заголовок ответа. По умолчанию размер одного буфера равен размеру страницы памяти. В зависимости от платформы это или 4К, или 8К, однако его можно сделать меньше.

proxy_buffering

<i>Синтаксис</i>	proxy_buffering on off;
По умолчанию	proxy_buffering on;
<i>Контекст</i>	http, server, location

on	Angie принимает ответ проксируемого сервера как можно быстрее, сохраняя его в буферы, заданные директивами <i>proxy_buffer_size</i> и <i>proxy_buffers</i> . Если ответ не вмещается целиком в память, то его часть может быть записана на диск во <i>временный файл</i> . Запись во временные файлы контролируется директивами <i>proxy_max_temp_file_size</i> и <i>proxy_temp_file_write_size</i> .
off	Ответ синхронно передается клиенту сразу же по мере его поступления. Angie не пытается считать весь ответ проксируемого сервера. Максимальный размер данных, который Angie может принять от сервера за один раз, задается директивой <i>proxy_buffer_size</i> .

Буферизация может быть также включена или выключена путем передачи значения "yes" или "no" в поле X-Accel-Buffering заголовка ответа. Эту возможность можно запретить с помощью директивы *proxy_ignore_headers*.

Angie не кэширует ответы, если параметр proxy_buffering отключен (off).

proxy_buffers

<i>Синтаксис</i>	proxy_buffers <i>число размер</i> ;
По умолчанию	proxy_buffers 8 4k 8k;
<i>Контекст</i>	http, server, location

Задаёт число и размер буферов для одного соединения, в которые будет читаться ответ, получаемый от проксируемого сервера.

По умолчанию размер одного буфера равен размеру страницы. В зависимости от платформы это или 4К, или 8К.

proxy_busy_buffers_size

<i>Синтаксис</i>	proxy_busy_buffers_size <i>размер</i> ;
По умолчанию	proxy_busy_buffers_size 8k 16k;
<i>Контекст</i>	http, server, location

При включенной *буферизации* ответов проксируемого сервера, ограничивает суммарный размер буферов, которые могут быть заняты для отправки ответа клиенту, пока ответ еще не прочитан целиком. Оставшиеся буферы тем временем могут использоваться для чтения ответа и, при необходимости, буферизации части ответа во временный файл.

По умолчанию размер ограничен величиной двух буферов, заданных директивами *proxy_buffer_size* и *proxy_buffers*.

proxy_cache

<i>Синтаксис</i>	proxy_cache <i>зона</i> off [path= <i>путь</i>];
По умолчанию	proxy_cache off;
<i>Контекст</i>	http, server, location

Задаёт зону разделяемой памяти для кэширования. Одну и ту же зону можно использовать в нескольких местах конфигурации. В значении параметра допускаются переменные.

off	запрещает кэширование, унаследованное с предыдущего уровня конфигурации.

В Angie PRO можно указать несколько директив *proxy_cache_path*, использующих одно и то же значение *keys_zone*, чтобы включить *шардинг кэша*. При этом следует задать параметр *path* директивы *proxy_cache*, использующей это значение *keys_zone*:

path= <i>путь</i>	Значение вычисляется в момент <i>кэширования</i> ответа от бэкенда и предполагает использование переменных, в том числе содержащих информацию из ответа. Если ответ берется из кэша, то <i>путь</i> не вычисляется заново; таким образом, кэшированный ответ сохраняет изначальный <i>путь</i> , пока не будет удален из кэша.

Это позволяет выбирать нужный путь кэша, применяя директивы *map* или скрипты к ответам от бэкенда. Пример для Content-Type:

```
proxy_cache_path /cache/one keys_zone=zone:10m;
proxy_cache_path /cache/two keys_zone=zone;

map $upstream_http_content_type $cache {
    ~^text/ one;
    default two;
}

server {
    ...
```



```
location / {
    proxy_pass http://backend;
    proxy_cache zone path=/cache/$cache;
    proxy_cache_valid 200 10m;
}
```

Здесь есть два пути кэша и отображение переменной, чтобы их различать. Если Content-Type начинается с text/, будет выбран первый путь, иначе — второй.

Примечание

При использовании *proxy_cache* обычно необходимо также задать директиву *proxy_cache_valid*, чтобы явно указать время действия кэшированных ответов. Если она не задана, Angie не использует значения по умолчанию, а определяет время хранения ответа в кэше на основе HTTP-заголовков ответа от сервера в следующем порядке приоритета:

1. Заголовок X-Accel-Expires (наивысший приоритет).
2. Заголовок Cache-Control с параметрами max-age или s-maxage.
3. Заголовок Expires.

Если ни один заголовок не содержит допустимого значения или их нет вовсе, ответ не будет кэшироваться, так как определить срок его действия нельзя.

proxy_cache_background_update

<i>Синтаксис</i>	proxy_cache_background_update on off;
По умолчанию	proxy_cache_background_update off;
<i>Контекст</i>	http, server, location

Позволяет запустить фоновый подзапрос для обновления просроченного элемента кэша, в то время как клиенту возвращается устаревший кэшированный ответ.

Предупреждение

Использование устаревшего кэшированного ответа в момент его обновления должно быть *разрешено*.

proxy_cache_bypass

<i>Синтаксис</i>	proxy_cache_bypass ...;
По умолчанию	—
<i>Контекст</i>	http, server, location

Задаёт условия, при которых ответ не будет браться из кэша. Если значение хотя бы одного из строковых параметров непустое и не равно "0", то ответ не берется из кэша:

```
proxy_cache_bypass $cookie_nocache $arg_nocache$arg_comment;  
proxy_cache_bypass $http_pragma $http_authorization;
```

Можно использовать совместно с директивой *proxy_no_cache*.

proxy_cache_convert_head

<i>Синтаксис</i>	proxy_cache_convert_head on off;
По умолчанию	proxy_cache_convert_head on;
<i>Контекст</i>	http, server, location

Разрешает или запрещает преобразование метода "HEAD" в "GET" для кэширования. Если преобразование выключено, то необходимо, чтобы *ключ кэширования* включал в себя *\$request_method*.

proxy_cache_key

<i>Синтаксис</i>	proxy_cache_key строка;
По умолчанию	proxy_cache_key \$scheme\$proxy_host\$request_uri;
<i>Контекст</i>	http, server, location

Задаёт ключ для кэширования, например,

```
proxy_cache_key "$host$request_uri $cookie_user";
```

По умолчанию значение директивы близко к строке

```
proxy_cache_key $scheme$proxy_host$uri$is_args$args;
```

proxy_cache_lock

<i>Синтаксис</i>	proxy_cache_lock on off;
По умолчанию	proxy_cache_lock off;
<i>Контекст</i>	http, server, location

Если включено, одновременно только одному запросу будет позволено заполнить новый элемент кэша, идентифицируемый согласно директиве *proxy_cache_key*, передав запрос на проксируемый сервер. Остальные запросы этого же элемента будут либо ожидать появления ответа в кэше, либо освобождения блокировки этого элемента, в течение времени, заданного директивой *proxy_cache_lock_timeout*.

proxy_cache_lock_age

<i>Синтаксис</i>	proxy_cache_lock_age <i>время</i> ;
По умолчанию	proxy_cache_lock_age 5s;
<i>Контекст</i>	http, server, location

Если последний запрос, переданный на проксируемый сервер для заполнения нового элемента кэша, не завершился за указанное время, на проксируемый сервер может быть передан еще один запрос.

proxy_cache_lock_timeout

<i>Синтаксис</i>	proxy_cache_lock_timeout <i>время</i> ;
По умолчанию	proxy_cache_lock_timeout 5s;
<i>Контекст</i>	http, server, location

Задаёт таймаут для *proxy_cache_lock*. По истечении указанного времени запрос будет передан на проксируемый сервер, однако ответ не будет кэширован.

proxy_cache_max_range_offset

<i>Синтаксис</i>	proxy_cache_max_range_offset <i>число</i> ;
По умолчанию	—
<i>Контекст</i>	http, server, location

Задаёт смещение в байтах для запросов с указанием диапазона запрашиваемых байт (byte-range requests). Если диапазон находится за указанным смещением, range-запрос будет передан на проксируемый сервер и ответ не будет кэширован.

proxy_cache_methods

<i>Синтаксис</i>	proxy_cache_methods GET HEAD POST ...;
По умолчанию	proxy_cache_methods GET HEAD;
<i>Контекст</i>	http, server, location

Если метод запроса клиента указан в этой директиве, то ответ будет кэширован. Методы "GET" и "HEAD" всегда добавляются в список, но тем не менее рекомендуется перечислять их явно. См. также директиву *proxy_no_cache*.

proxy_cache_min_uses

<i>Синтаксис</i>	<code>proxy_cache_min_uses число;</code>
По умолчанию	<code>proxy_cache_min_uses 1;</code>
<i>Контекст</i>	http, server, location

Задаёт число запросов, после которого ответ будет кэширован.

Предупреждение

Метаданные кэша хранятся в разделяемой памяти. Ручное удаление файлов кэша не сбрасывает счетчики и может привести к непредсказуемому поведению. Для полного сброса остановите сервер, удалите директорию кэша и запустите снова.

Примечание

Сторонние модули очистки кэша (например, external-cache-purge) удаляют только файлы, но не сбрасывают счетчик `proxy_cache_min_uses`. Директива предназначена для защиты кэша от загрязнения редкими запросами, и сброс счетчика при очистке может негативно повлиять на производительность.

proxy_cache_path

<i>Синтаксис</i>	<code>proxy_cache_path путь [levels=уровни] [use_temp_path=on off] keys_zone=имя:размер[:file=файл] [inactive=время] [max_size=размер] [min_free=размер] [manager_files=число] [manager_sleep=время] [manager_threshold=время] [loader_files=число] [loader_sleep=время] [loader_threshold=время];</code>
По умолчанию	—
<i>Контекст</i>	http

Задаёт *путь* и другие параметры кэша. Данные кэша хранятся в файлах. Именем файла в кэше является результат функции MD5 от *ключа кэширования*.

levels	задаёт уровни иерархии кэша: можно задать от 1 до 3 уровней, на каждом уровне допускаются значения 1 или 2.

Например, при использовании

```
proxy_cache_path /data/angie/cache levels=1:2 keys_zone=one:10m;
```

имена файлов в кэше будут такого вида:

```
/data/angie/cache/c/29/b7f54b2df7773722d382f4809d65029c
```

Кэшируемый ответ сначала записывается во временный файл, а потом этот файл переименовывается. Временные файлы и кэш могут располагаться на разных файловых системах. Однако нужно

учитывать, что в этом случае вместо дешевой операции переименовывания в пределах одной файловой системы файл копируется с одной файловой системы на другую. Поэтому лучше, если кэш будет находиться на той же файловой системе, что и каталог с временными файлами.

<code>use_temp_path=on</code>	определяет каталог, который будет использоваться для временных файлов
<code> off</code>	
<code>on</code>	Если параметр не задан или установлен в значение <code>on</code> , будет использоваться каталог, задаваемый директивой <code>proxy_temp_path</code> для данного <code>location</code>
<code>off</code>	временные файлы будут располагаться непосредственно в каталоге кэша
<code>keys_zone</code>	Задаёт имя и размер зоны разделяемой памяти для хранения всех активных ключей и информации о данных. Метаданные кэша хранятся в разделяемой памяти. Зоны размером в 1 мегабайт достаточно для хранения около 8000 ключей. Когда с <code>keys_zone</code> задан необязательный <i>файл</i>, Angie сбрасывает содержимое этой зоны на диск при завершении работы основного процесса и пытается восстановить ее по тому же адресу памяти при следующем запуске или после обновления бинарных файлов, чтобы обеспечить более надежную сохранность данных и сократить время загрузки кэша. Если восстановление области невозможно из-за изменения ее размера, несовместимости версий бинарных файлов или по другим причинам, Angie запишет в журнал предупреждение (<code>failed to restore zone at address</code>) и не будет использовать механизм восстановления зоны. Вместо этого несовместимый файл будет переименован в <code>.old</code>; вы можете либо удалить его, либо восстановить его имя и вернуть Angie к конфигурации и версии, в которых этот файл был изначально создан.
 Предупреждение Убедитесь, что путь к <i>файлу</i> указан правильно и имеет необходимые права доступа для использования Angie, а также защищен от несанкционированного доступа; относительные пути основаны на префиксе.	
<code>inactive</code>	Если к данным кэша не обращаются в течение времени, заданного этим параметром, то данные удаляются, независимо от их свежести. По умолчанию <code>inactive</code> равен 10 минутам.

Примечание

В Angie PRO можно задавать несколько директив `proxy_cache_path` с одним и тем же значением `keys_zone`. Размер зоны разделяемой памяти можно указывать только в первой из них. Выбор между директивами будет производиться на основе параметра `path` соответствующей директивы `proxy_cache`.

Специальный процесс **менеджера кэша** следит за максимальным размером кэша, а также за минимальным объемом свободного места на файловой системе с кэшем, и удаляет наименее востребованные данные при превышении максимального размера кэша или недостаточном объеме свободного места. Удаление данных происходит итерациями.

max_size	максимальное пороговое значение размера кэша
min_free	минимальное пороговое значение объема свободного места на файловой системе с кэшем
manager_files	максимальное количество удаляемых элементов кэша за одну итерацию По умолчанию: 100.
manager_threshold	ограничивает время работы одной итерации По умолчанию: 200 миллисекунд.
manager_sleep	время, в течение которого выдерживается пауза между итерациями По умолчанию: 50 миллисекунд.

Через минуту после запуска Angie активируется специальный процесс **загрузчика кэша**. Он сканирует файловую систему в поисках ранее закэшированных данных и загружает эту информацию в область кэша. Этот процесс выполняется итеративно; каждая итерация обрабатывает ограниченное количество элементов, заданное параметром **loader_files**, следит за тем, чтобы не превышать **loader_threshold**, затем делает короткую паузу, заданную **loader_sleep**, перед переходом к следующей партии. Итерации продолжаются до тех пор, пока загрузчик не обработает все существующие записи кэша на диске:

loader_files	максимальное количество элементов кэша к загрузке в одну итерацию По умолчанию: 100
loader_threshold	ограничивает время работы одной итерации По умолчанию: 200 миллисекунд
loader_sleep	время, в течение которого выдерживается пауза между итерациями По умолчанию: 50 миллисекунд

Примечание

Указание *файла* для параметра **keys_zone** не влияет на работу загрузчика кэша.

proxy_cache_revalidate

<i>Синтаксис</i>	proxy_cache_revalidate on off;
По умолчанию	proxy_cache_revalidate off;
<i>Контекст</i>	http, server, location

Разрешает ревалидацию просроченных элементов кэша при помощи условных запросов с полями заголовка **If-Modified-Since** и **If-None-Match**.

proxy_cache_use_stale

<i>Синтаксис</i>	proxy_cache_use_stale error timeout invalid_header updating http_500 http_502 http_503 http_504 http_403 http_404 http_429 off ...;
По умолчанию	proxy_cache_use_stale off;
<i>Контекст</i>	http, server, location

Определяет, в каких случаях можно использовать устаревший кэшированный ответ. Параметры директивы совпадают с параметрами директивы *proxy_next_upstream*.

error	Позволяет использовать устаревший кэшированный ответ при невозможности выбора проксируемого сервера для обработки запроса.
updating	Дополнительный параметр, разрешает использовать устаревший кэшированный ответ, если на данный момент он уже обновляется. Это позволяет минимизировать число обращений к проксируемым серверам при обновлении кэшированных данных.

Использование устаревшего кэшированного ответа может также быть разрешено непосредственно в заголовке ответа на определенное количество секунд после того, как ответ устарел:

- Расширение `stale-while-revalidate` поля заголовка `Cache-Control` разрешает использовать устаревший кэшированный ответ, если на данный момент он уже обновляется.
- Расширение `stale-if-error` поля заголовка `Cache-Control` разрешает использовать устаревший кэшированный ответ в случае ошибки.

Примечание

Такой способ менее приоритетен, чем задание параметров директивы.

Чтобы минимизировать число обращений к проксируемым серверам при заполнении нового элемента кэша, можно воспользоваться директивой `proxy_cache_lock`.

proxy_cache_valid

<i>Синтаксис</i>	<code>proxy_cache_valid [код ...] время;</code>
По умолчанию	—
<i>Контекст</i>	<code>http, server, location</code>

Задаёт время кэширования для разных кодов ответа. Например, директивы

```
proxy_cache_valid 200 302 10m;
proxy_cache_valid 404 1m;
```

задают время кэширования 10 минут для ответов с кодами 200 и 302 и 1 минуту для ответов с кодом 404.

Если указано только время кэширования,

```
proxy_cache_valid 5m;
```

то кэшируются только ответы 200, 301 и 302.

Кроме того, можно кэшировать любые ответы с помощью параметра `any`:

```
proxy_cache_valid 200 302 10m;
proxy_cache_valid 301 1h;
proxy_cache_valid any 1m;
```

Примечание

Параметры кэширования могут также быть заданы непосредственно в заголовке ответа. Такой способ приоритетнее, чем задание времени кэширования с помощью директивы.

- Поле заголовка **X-Accel-Expires** задает время кэширования ответа в секундах. Значение *0* запрещает кэшировать ответ. Если значение начинается с префикса *@*, оно задает абсолютное время в секундах с начала эпохи, до которого ответ может быть кэширован.
- Если в заголовке нет поля **X-Accel-Expires**, параметры кэширования определяются по полям заголовка **Expires** или **Cache-Control**.
- Ответ, в заголовке которого есть поле **Set-Cookie**, не будет кэшироваться.
- Ответ, в заголовке которого есть поле **Vary** со специальным значением *"*"*, не будет кэшироваться. Ответ, в заголовке которого есть поле **Vary** с другим значением, будет кэширован с учетом соответствующих полей заголовка запроса.

Обработка одного или более из этих полей заголовка может быть отключена при помощи директивы *proxy_ignore_headers*.

proxy_connect_timeout

<i>Синтаксис</i>	<code>proxy_connect_timeout время;</code>
По умолчанию	<code>proxy_connect_timeout 60s;</code>
<i>Контекст</i>	<code>http, server, location</code>

Задаёт таймаут для установления соединения с проксируемым сервером. Необходимо иметь в виду, что этот таймаут обычно не может превышать 75 секунд.

proxy_connection_drop

<i>Синтаксис</i>	<code>proxy_connection_drop время on off;</code>
По умолчанию	<code>proxy_connection_drop off;</code>
<i>Контекст</i>	<code>http, server, location</code>

Настраивает завершение всех соединений с проксируемым сервером, если он был удален из группы или помечен как постоянно недоступный в результате процесса *resolve* или команды *API DELETE*.

Соединение завершается, когда обрабатывается следующее событие чтения или записи для клиента или проксируемого сервера.

Установка *времени* включает *таймаут* до завершения соединения; при выборе значения *on* соединения завершаются немедленно.

proxy_cookie_domain

<i>Синтаксис</i>	<code>proxy_cookie_domain off;</code> <code>proxy_cookie_domain домен замена;</code>
По умолчанию	<code>proxy_cookie_domain off;</code>
<i>Контекст</i>	<code>http, server, location</code>

Задаёт текст, который нужно изменить в атрибуте **domain** полей **Set-Cookie** заголовка ответа проксируемого сервера. Предположим, проксируемый сервер вернул поле заголовка **Set-Cookie** с атрибутом *"domain=localhost"*. Директива


```
proxy_cookie_domain localhost example.org;
```

перепишет данный атрибут в виде "domain=example.org".

Точка в начале строк *домен* и *замена*, а равно как и в атрибуте *domain* игнорируется. Регистр значения не имеет.

В строках *домен* и *замена* можно использовать переменные:

```
proxy_cookie_domain www.$host $host;
```

Директиву также можно задать при помощи регулярных выражений. При этом *домен* должен начинаться с символа "~". Регулярное выражение может содержать именованные и позиционные группы захвата, а *замена* ссылаться на них:

```
proxy_cookie_domain ~\.(?P<sl_domain>[-0-9a-z]+\.[a-z]+)$ $sl_domain;
```

На одном уровне может быть указано несколько директив *proxy_cookie_domain*:

```
proxy_cookie_domain localhost example.org;
proxy_cookie_domain ~\.[a-z]+\.[a-z]+)$ $1;
```

Если к cookie могут быть применены несколько директив, будет выбрана первая из них.

Параметр *off* отменяет действие унаследованных с предыдущего уровня конфигурации директив *proxy_cookie_domain*.

proxy_cookie_flags

<i>Синтаксис</i>	<code>proxy_cookie_flags off cookie [флаг ...];</code>
По умолчанию	<code>proxy_cookie_flags off;</code>
<i>Контекст</i>	http, server, location

Задаёт один или несколько флагов для cookie. В качестве cookie можно использовать текст, переменные и их комбинации. В качестве флага можно использовать текст, переменные и их комбинации.

Параметры *secure*, *httponly*, *samesite=strict*, *samesite=lax*, *samesite=none* добавляют соответствующие флаги.

Параметры *nosecure*, *nohttponly*, *nosamesite* удаляют соответствующие флаги.

Cookie также можно задать при помощи регулярных выражений. При этом cookie должен начинаться с символа "~".

На одном уровне конфигурации может быть указано несколько директив *proxy_cookie_flags*:

```
proxy_cookie_flags one httponly;
proxy_cookie_flags ~ nsecure samesite=strict;
```

Если к cookie могут быть применены несколько директив, будет выбрана первая из них. В данном примере флаг *httponly* добавляется к cookie *one*, для остальных cookie добавляется флаг *samesite=strict* и удаляется флаг *secure*.

Параметр *off* отменяет действие всех директив *proxy_cookie_flags* на данном уровне.

proxy_cookie_path

<i>Синтаксис</i>	<code>proxy_cookie_path off;</code> <code>proxy_cookie_path <i>путь</i> <i>замена</i>;</code>
По умолчанию	<code>proxy_cookie_path off;</code>
<i>Контекст</i>	http, server, location

Задаёт текст, который нужно изменить в атрибуте path полей **Set-Cookie** заголовка ответа проксируемого сервера. Предположим, проксируемый сервер вернул поле заголовка **Set-Cookie** с атрибутом "path=/two/some/uri/". Директива

```
proxy_cookie_path /two/ /;
```

перепишет данный атрибут в виде "path=/some/uri/".

В строках *путь* и *замена* можно использовать переменные:

```
proxy_cookie_path $uri /some$uri;
```

Директиву также можно задать при помощи регулярных выражений. При этом *путь* должен начинаться либо с символа "~", если при сравнении следует учитывать регистр символов, либо с символов "~*", если регистр символов учитывать не нужно. Регулярное выражение может содержать именованные и позиционные группы захвата, а *замена* ссылаться на них:

```
proxy_cookie_path ~*/user/([~/]+) /u/$1;
```

На одном уровне может быть указано несколько директив *proxy_cookie_path*:

```
proxy_cookie_path /one/ /;  
proxy_cookie_path / /two/;
```

Если к cookie могут быть применены несколько директив, будет выбрана первая из них.

Параметр **off** отменяет действие унаследованных с предыдущего уровня конфигурации директив *proxy_cookie_path*.

proxy_force_ranges

<i>Синтаксис</i>	<code>proxy_force_ranges off;</code>
По умолчанию	<code>proxy_force_ranges off;</code>
<i>Контекст</i>	http, server, location

Включает поддержку диапазонов запрашиваемых байт (byte-range) для кэшированных и некэшированных ответов проксируемого сервера вне зависимости от наличия поля **Accept-Ranges** в заголовках этих ответов.

proxy_headers_hash_bucket_size

<i>Синтаксис</i>	proxy_headers_hash_bucket_size <i>размер</i> ;
По умолчанию	proxy_headers_hash_bucket_size 64;
<i>Контекст</i>	http, server, location

Задаёт размер корзины для хэш-таблиц, используемых директивами *proxy_hide_header* и *proxy_set_header*. Подробнее настройка хэш-таблиц обсуждается *отдельно*.

proxy_headers_hash_max_size

<i>Синтаксис</i>	proxy_headers_hash_max_size <i>размер</i> ;
По умолчанию	proxy_headers_hash_max_size 512;
<i>Контекст</i>	http, server, location

Задаёт максимальный размер хэш-таблиц, используемых директивами *proxy_hide_header* и *proxy_set_header*. Подробнее настройка хэш-таблиц обсуждается *отдельно*.

proxy_hide_header

<i>Синтаксис</i>	proxy_hide_header <i>поле</i> ;
По умолчанию	—
<i>Контекст</i>	http, server, location

По умолчанию Angie не передаёт клиенту поля заголовка Date, Server, X-Pad и X-Accel-... из ответа проксированного сервера. Директива *proxy_hide_header* задаёт дополнительные поля, которые не будут передаваться. Если же передачу полей нужно разрешить, можно воспользоваться директивой *proxy_pass_header*.

proxy_http_version

<i>Синтаксис</i>	proxy_http_version 1.0 1.1 3;
По умолчанию	proxy_http_version 1.0;
<i>Контекст</i>	http, server, location, if в location, limit_except

Задаёт версию протокола HTTP для проксирования. По умолчанию используется версия 1.0. Для работы *постоянных соединений* рекомендуется версия 1.1 или выше.

proxy_http3_hq

<i>Синтаксис</i>	proxy_http3_hq on off;
По умолчанию	proxy_http3_hq off;
<i>Контекст</i>	http, server

Отключает или включает особый режим согласования **hq-interop**, используемый в функциональных тестах *QUIC*, на которые полагается Angie.

Предупреждение

Включайте этот режим только для запуска специализированных тестов, которым в явной форме необходим данный режим.

proxy_http3_max_concurrent_streams

<i>Синтаксис</i>	proxy_http3_max_concurrent_streams число;
По умолчанию	proxy_http3_max_concurrent_streams 128;
<i>Контекст</i>	http, server

Инициализирует настройки HTTP/3 и QUIC, а также задает максимальное число параллельных HTTP/3-потоков в *соединении*. Требуется включения *постоянных соединений*.

proxy_http3_max_table_capacity

<i>Синтаксис</i>	proxy_http3_max_table_capacity число;
Значение по умолчанию	proxy_http3_max_table_capacity 4096;
<i>Контекст</i>	http, server, location

Определяет емкость *динамической таблицы* для прокси-соединений.

Примечание

Похожая директива *http3_max_table_capacity* задает это значение для серверных соединений. Чтобы избежать ошибок, использование динамической таблицы отключается при включенном кэшировании в режиме проксирования.

proxy_http3_stream_buffer_size

<i>Синтаксис</i>	proxy_http3_stream_buffer_size <i>размер</i> ;
По умолчанию	proxy_http3_stream_buffer_size 64k;
<i>Контекст</i>	http, server

Задаёт *размер* буфера, используемого для чтения и записи *QUIC-потоков*.

proxy_ignore_client_abort

<i>Синтаксис</i>	proxy_ignore_client_abort on off;
По умолчанию	proxy_ignore_client_abort off;
<i>Контекст</i>	http, server, location

Определяет, закрывать ли соединение с проксируемым сервером в случае, если клиент закрыл соединение, не дождавись ответа.

proxy_ignore_headers

<i>Синтаксис</i>	proxy_ignore_headers <i>поле ...</i> ;
По умолчанию	—
<i>Контекст</i>	http, server, location

Запрещает обработку некоторых полей заголовка из ответа проксированного сервера. В директиве можно указать поля X-Accel-Redirect, X-Accel-Expires, X-Accel-Limit-Rate, X-Accel-Buffering, X-Accel-Charset, Expires, Cache-Control, Set-Cookie и Vary.

Если не запрещено, обработка этих полей заголовка заключается в следующем:

- X-Accel-Expires, Expires, Cache-Control, Set-Cookie и Vary задают *параметры кэширования* ответа;
- X-Accel-Redirect производит *внутреннее перенаправление* на указанный URI;
- X-Accel-Limit-Rate задаёт *ограничение скорости* передачи ответа клиенту;
- X-Accel-Buffering включает или выключает *буферизацию* ответа;
- X-Accel-Charset задаёт желаемую *кодировку* ответа.

proxy_intercept_errors

<i>Синтаксис</i>	proxy_intercept_errors on off;
По умолчанию	proxy_intercept_errors off;
<i>Контекст</i>	http, server, location

Определяет, передавать ли клиенту проксированные ответы с кодом больше либо равным 300, или же перехватывать их и перенаправлять на обработку Angie с помощью директивы *error_page*.

proxy_limit_rate

<i>Синтаксис</i>	proxy_limit_rate <i>скорость</i> ;
По умолчанию	proxy_limit_rate 0;
<i>Контекст</i>	http, server, location

Ограничивает скорость чтения ответа от проксируемого сервера. *Скорость* задается в байтах в секунду; можно использовать переменные.

0	отключает ограничение скорости

Примечание

Ограничение устанавливается на запрос, поэтому, если Angie одновременно откроет два соединения к проксируемому серверу, суммарная скорость будет вдвое выше заданного ограничения. Ограничение работает только в случае, если включена *буферизация* ответов проксируемого сервера.

proxy_max_temp_file_size

<i>Синтаксис</i>	proxy_max_temp_file_size <i>размер</i> ;
По умолчанию	proxy_max_temp_file_size 1024m;
<i>Контекст</i>	http, server, location

Если включена *буферизация* ответов проксируемого сервера, и ответ не помещается целиком в буферы, заданные директивами *proxy_buffer_size* и *proxy_buffers*, часть ответа может быть записана во временный файл. Эта директива задает максимальный размер временного файла. Размер данных, сбрасываемых во временный файл за один раз, задается директивой *proxy_temp_file_write_size*.

0	отключает возможность буферизации ответов во временные файлы

Примечание

Данное ограничение не распространяется на ответы, которые будут *кэшированы* или сохранены на диске.

proxy_method

<i>Синтаксис</i>	proxy_method <i>метод</i> ;
По умолчанию	—
<i>Контекст</i>	http, server, location

Задаёт HTTP-метод, который будет использоваться в передаваемых на проксируемый сервер запросах вместо метода из клиентского запроса. В значении параметра допустимо использование переменных.

proxy_next_upstream

<i>Синтаксис</i>	proxy_next_upstream error timeout invalid_header http_500 http_502 http_503 http_504 http_403 http_404 http_429 non_idempotent off ...;
По умолчанию	proxy_next_upstream error timeout;
<i>Контекст</i>	http, server, location

Определяет, в каких случаях запрос будет передан следующему в группе *upstream* серверу:

error	произошла ошибка соединения с сервером, передачи ему запроса или чтения заголовка ответа сервера;
timeout	произошел таймаут во время соединения с сервером, передачи ему запроса или чтения заголовка ответа сервера;
invalid_header	сервер вернул пустой или неверный ответ;
http_500	сервер вернул ответ с кодом 500;
http_502	сервер вернул ответ с кодом 502;
http_503	сервер вернул ответ с кодом 503;
http_504	сервер вернул ответ с кодом 504;
http_403	сервер вернул ответ с кодом 403;
http_404	сервер вернул ответ с кодом 404;
http_429	сервер вернул ответ с кодом 429;
non_idempotent	обычно запросы с <i>неидемпотентным</i> методом (<i>POST</i> , <i>LOCK</i> , <i>PATCH</i>) не передаются на другой сервер, если запрос серверу группы уже был отправлен; включение параметра явно разрешает повторять подобные запросы;
off	запрещает передачу запроса следующему серверу.

Примечание

Необходимо понимать, что передача запроса следующему серверу возможна только при условии, что клиенту еще ничего не передавалось. То есть, если ошибка или таймаут возникли в середине передачи ответа клиенту, то действие директивы на такой запрос не распространяется.

Директива также определяет, что считается *неудачной попыткой* работы с сервером.

error	всегда считаются неудачными попытками, даже если они не указаны в директиве
timeout	
invalid_header	
http_500	считаются неудачными попытками, только если они указаны в директиве
http_502	
http_503	
http_504	
http_429	
http_403	никогда не считаются неудачными попытками
http_404	

Передача запроса следующему серверу может быть ограничена по *количеству попыток* и по *времени*.

proxy_next_upstream_timeout

<i>Синтаксис</i>	proxy_next_upstream_timeout <i>время</i> ;
По умолчанию	proxy_next_upstream_timeout 0;
<i>Контекст</i>	http, server, location

Ограничивает время, в течение которого возможна передача запроса *следующему* серверу.

0	отключает это ограничение

proxy_next_upstream_tries

<i>Синтаксис</i>	proxy_next_upstream_tries <i>число</i> ;
По умолчанию	proxy_next_upstream_tries 0;
<i>Контекст</i>	http, server, location

Ограничивает число допустимых попыток для передачи запроса *следующему* серверу.

0	отключает это ограничение

proxy_no_cache

<i>Синтаксис</i>	proxy_no_cache <i>строка ...</i> ;
По умолчанию	—
<i>Контекст</i>	http, server, location

Задаёт условия, при которых ответ не будет сохраняться в кэш. Если значение хотя бы одного из строковых параметров непустое и не равно "0", то ответ не будет сохранен:

```
proxy_no_cache $cookie_nocache $arg_nocache$arg_comment;
proxy_no_cache $http_pragma $http_authorization;
```

Можно использовать совместно с директивой *proxy_cache_bypass*.

proxy_pass

<i>Синтаксис</i>	proxy_pass <i>uri</i> ;
По умолчанию	—
<i>Контекст</i>	location, if в location, limit_except

Задаёт протокол и адрес проксируемого сервера, а также необязательный URI, на который должен отображаться location. В качестве протокола можно указать **http** или **https**. Адрес может быть указан в виде доменного имени или IP-адреса, и необязательного порта:


```
proxy_pass http://localhost:8000/uri/;
```

или в виде пути UNIX-сокета, который указывается после слова `unix` и заключается в двоеточия:

```
proxy_pass http://unix:/tmp/backend.socket:/uri/;
```

Если доменному имени соответствует несколько адресов, то все они будут использоваться по очереди (round-robin). Кроме того, в качестве адреса можно указать *группу серверов*.

В значении параметра можно использовать переменные. В этом случае, если адрес указан в виде доменного имени, имя ищется среди описанных групп серверов и если не найдено, то определяется с помощью *resolver*'а.

URI запроса передается на сервер так:

- Если директива `proxy_pass` указана с **URI**, то при передаче запроса серверу часть *нормализованного* URI запроса, соответствующая location, заменяется на URI, указанный в директиве:

```
location /name/ {
    proxy_pass http://127.0.0.1/remote/;
}
```

- Если директива `proxy_pass` указана **без URI**, то при обработке первоначального запроса на сервер передается URI запроса в том же виде, в каком его прислал клиент, а при обработке измененного URI - нормализованный URI запроса целиком:

```
location /some/path/ {
    proxy_pass http://127.0.0.1;
}
```

В ряде случаев часть URI запроса, подлежащую замене, выделить невозможно:

- Если location задан регулярным выражением, а также в именованных location.

В этих случаях `proxy_pass` следует указывать без URI.

- Если внутри проксируемого location с помощью директивы *rewrite* изменяется URI, и именно с этой конфигурацией будет обрабатываться запрос (*break*):

```
location /name/ {
    rewrite /name/([~/]+) /users?name=$1 break;
    proxy_pass http://127.0.0.1;
}
```

В этом случае URI, указанный в директиве, игнорируется, и на сервер передается измененный URI запроса целиком.

- При использовании переменных в *proxy_pass*:

```
location /name/ {
    proxy_pass http://127.0.0.1$request_uri;
}
```

В этом случае если в директиве указан URI, он передается на сервер как есть, заменяя URI первоначального запроса.

Проксирование WebSocket требует особой *настройки*.

Примечание

Если `proxy_pass` стоит в `location` с косой чертой в конце префикса (например, `location /name/`), и при этом в директиве `auto_redirect` указано `default`, запросы без косой черты в конце будут перенаправляться (`/name -> /name/`).

proxy_pass_header

<i>Синтаксис</i>	<code>proxy_pass_header поле ...;</code>
По умолчанию	—
<i>Контекст</i>	<code>http, server, location</code>

Разрешает передавать от проксируемого сервера клиенту *запрещенные для передачи* поля заголовка.

proxy_pass_request_body

<i>Синтаксис</i>	<code>proxy_pass_request_body on off;</code>
По умолчанию	<code>proxy_pass_request_body on;</code>
<i>Контекст</i>	<code>http, server, location</code>

Позволяет запретить передачу исходного тела запроса на проксируемый сервер.

```
location /x-accel-redirect-here/ {
    proxy_method GET;
    proxy_pass_request_body off;
    proxy_set_header Content-Length "";

    proxy_pass ...;
}
```

См. также директивы `proxy_set_header` и `proxy_pass_request_headers`.

proxy_pass_request_headers

<i>Синтаксис</i>	<code>proxy_pass_request_headers on off;</code>
По умолчанию	<code>proxy_pass_request_headers on;</code>
<i>Контекст</i>	<code>http, server, location</code>

Позволяет запретить передачу полей заголовка исходного запроса на проксируемый сервер.

```
location /x-accel-redirect-here/ {
    proxy_method GET;
    proxy_pass_request_headers off;
    proxy_pass_request_body off;

    proxy_pass ...;
}
```

См. также директивы `proxy_set_header` и `proxy_pass_request_body`.

proxy_pass_trailers

<i>Синтаксис</i>	<code>proxy_pass_trailers on off;</code>
По умолчанию	<code>proxy_pass_trailers off;</code>
<i>Контекст</i>	<code>http, server, location</code>

Разрешает передачу полей трейлеров от проксируемого сервера клиенту.

Секция трейлеров в HTTP/1.1 включается явно.

```
location / {
    proxy_http_version 1.1;
    proxy_set_header Connection "te";
    proxy_set_header TE "trailers";
    proxy_pass_trailers on;

    proxy_pass ...;
}
```

proxy_quic_active_connection_id_limit

<i>Синтаксис</i>	<code>proxy_quic_active_connection_id_limit число;</code>
По умолчанию	<code>proxy_quic_active_connection_id_limit 2;</code>
<i>Контекст</i>	<code>http, server</code>

Задаёт значение транспортного параметра *QUIC* `active_connection_id_limit`. Это максимальное число активных идентификаторов соединений, поддерживаемых для одного сервера.

proxy_quic_gso

<i>Синтаксис</i>	<code>proxy_quic_gso on off;</code>
По умолчанию	<code>proxy_quic_gso off;</code>
<i>Контекст</i>	<code>http, server</code>

Отключает или включает отправку данных в оптимизированном пакетном режиме *QUIC*, использующем программное снижение нагрузки путем сегментации (`generic segmentation offload`).

proxy_quic_host_key

<i>Синтаксис</i>	proxy_quic_host_key <i>файл</i> ;
По умолчанию	—
<i>Контекст</i>	http, server

Задаёт *файл* с секретным ключом, применяемым в *QUIC* при шифровании токенов Stateless Reset и Address Validation. По умолчанию случайный ключ создается при каждом перезапуске. Токены, созданные при помощи старых ключей, не принимаются.

proxy_read_timeout

<i>Синтаксис</i>	proxy_read_timeout <i>время</i> ;
По умолчанию	proxy_read_timeout 60s;
<i>Контекст</i>	http, server, location

Задаёт таймаут при чтении ответа проксируемого сервера. Таймаут устанавливается не на всю передачу ответа, а только между двумя операциями чтения. Если по истечении этого времени проксируемый сервер ничего не передаст, соединение закрывается.

proxy_redirect

<i>Синтаксис</i>	proxy_redirect default; proxy_redirect off; proxy_redirect <i>перенаправление</i> <i>замена</i> ;
По умолчанию	proxy_redirect default;
<i>Контекст</i>	http, server, location

Задаёт текст, который нужно изменить в полях заголовка "Location" и "Refresh" в ответе проксируемого сервера.

Предположим, проксируемый сервер вернул поле заголовка:

```
Location: http://localhost:8000/two/some/uri/
```

Директива

```
proxy_redirect http://localhost:8000/two/ http://frontend/one/;
```

перепишет эту строку в виде:

```
Location: http://frontend/one/some/uri/
```

В заменяемой строке можно не указывать имя сервера:

```
proxy_redirect http://localhost:8000/two/ /;
```

тогда будут подставлены основное имя сервера и порт, если он отличен от 80.

Стандартная замена, задаваемая параметром **default**, использует параметры директив *location* и *proxy_pass*. Поэтому две нижеприведенные конфигурации одинаковы:

```
location /one/ {
    proxy_pass      http://upstream:port/two/;
    proxy_redirect  default;
```

```
location /one/ {
    proxy_pass      http://upstream:port/two/;
    proxy_redirect  http://upstream:port/two/ /one/;
```

Предупреждение

Параметр `default` недопустим, если в `proxy_pass` используются переменные.

В строке *замена* можно использовать переменные:

```
proxy_redirect http://localhost:8000/ http://$host:$server_port/;
```

В строке *перенаправление* тоже можно использовать переменные:

```
proxy_redirect http://$proxy_host:8000/ /;
```

Директиву также можно задать при помощи регулярных выражений. При этом *перенаправление* должно начинаться либо с символа "~", если при сравнении следует учитывать регистр символов, либо с символов "~*", если регистр символов учитывать не нужно. Регулярное выражение может содержать именованные и позиционные группы захвата, а *замена* ссылаться на них:

```
proxy_redirect ~^(http://[~:]+):\d+(/.+)$ $1$2;
proxy_redirect ~*/user/([~/]+)/(.)$ http://$1.example.com/$2;
```

На одном уровне может быть указано несколько директив `proxy_redirect`:

```
proxy_redirect default;
proxy_redirect http://localhost:8000/ /;
proxy_redirect http://www.example.com/ /;
```

Если к полям заголовка в ответе проксируемого сервера могут быть применены несколько директив, будет выбрана первая из них.

Параметр `off` отменяет действие унаследованных с предыдущего уровня конфигурации директив `proxy_redirect`.

С помощью этой директивы можно также добавлять имя хоста к относительным перенаправлениям, выдаваемым проксируемым сервером:

```
proxy_redirect / /;
```

proxy_request_buffering

<i>Синтаксис</i>	<code>proxy_request_buffering on off;</code>
По умолчанию	<code>proxy_request_buffering on;</code>
<i>Контекст</i>	<code>http, server, location</code>

Разрешает или запрещает использовать буферизацию тела запроса клиента.

on	тело запроса полностью <i>читается</i> от клиента перед отправкой запроса на проксируемый сервер.
off	тело запроса отправляется на проксируемый сервер сразу же по мере его поступления. В этом случае запрос не может быть передан <i>следующему серверу</i> , если Angie уже начал отправку тела запроса.

Если для отправки тела исходного запроса используется HTTP/1.1 и передача данных частями (chunked transfer encoding), то тело запроса буферизуется независимо от значения директивы, если для проксирования также не *включен* HTTP/1.1.

proxy_send_lowat

<i>Синтаксис</i>	proxy_send_lowat <i>размер</i> ;
По умолчанию	proxy_send_lowat 0;
<i>Контекст</i>	http, server, location

При установке директивы в ненулевое значение Angie будет пытаться минимизировать число операций отправки на исходящих соединениях с проксируемым сервером либо при помощи флага NOTE_LOWAT метода *kqueue*, либо при помощи параметра сокета SO_SNDLOWAT, с указанным размером.

Примечание

Эта директива игнорируется на Linux, Solaris и Windows.

proxy_send_timeout

<i>Синтаксис</i>	proxy_send_timeout <i>время</i> ;
По умолчанию	proxy_send_timeout 60s;
<i>Контекст</i>	http, server, location

Задаёт таймаут при передаче запроса проксируемому серверу. Таймаут устанавливается не на всю передачу запроса, а только между двумя операциями записи. Если по истечении этого времени проксируемый сервер не примет новых данных, соединение закрывается.

proxy_set_body

<i>Синтаксис</i>	proxy_set_body <i>значение</i> ;
По умолчанию	—
<i>Контекст</i>	http, server, location

Позволяет переопределить тело запроса, передаваемое на проксируемый сервер. В качестве значения можно использовать текст, переменные и их комбинации.

proxy_set_header

<i>Синтаксис</i>	proxy_set_header поле значение;
По умолчанию	proxy_set_header Host \$proxy_host;
<i>Контекст</i>	http, server, location

Позволяет переопределять или добавлять поля заголовка запроса, *передаваемые* проксируемому серверу. В качестве *значения* можно использовать текст, переменные и их комбинации. Директивы наследуются с предыдущего уровня конфигурации при условии, что на данном уровне не описаны свои директивы *proxy_set_header*. По умолчанию переопределяются только два поля:

```
proxy_set_header Host $proxy_host;
proxy_set_header Connection close;
```

Если включено кэширование, поля заголовка If-Modified-Since , If-Unmodified-Since , If-None-Match , If-Match , Range и If-Range исходного запроса не передаются на проксируемый сервер.

Неизмененное поле заголовка запроса "Host" можно передать так:

```
proxy_set_header Host $http_host;
```

Однако, если это поле отсутствует в заголовке запроса клиента, то ничего передаваться не будет. В этом случае лучше воспользоваться переменной *\$host* - ее значение равно имени сервера в поле "Host" заголовка запроса, или же основному имени сервера, если поля нет:

```
proxy_set_header Host $host;
```

Кроме того, можно передать имя сервера вместе с портом проксируемого сервера:

```
proxy_set_header Host $host:$proxy_port;
```

Если значение поля заголовка — пустая строка, то поле вообще не будет передаваться проксируемому серверу:

```
proxy_set_header Accept-Encoding "";
```

proxy_socket_keepalive

<i>Синтаксис</i>	proxy_socket_keepalive on off;
По умолчанию	proxy_socket_keepalive off;
<i>Контекст</i>	http, server, location

Конфигурирует поведение "TCP keepalive" для исходящих соединений к проксируемому серверу.

""	По умолчанию для сокета действуют настройки операционной системы.
on	для сокета включается параметр <i>SO_KEEPALIVE</i>

proxy_ssl_certificate

<i>Синтаксис</i>	proxy_ssl_certificate <i>файл</i> [<i>файл</i>];
По умолчанию	—
<i>Контекст</i>	http, server, location

Задаёт файл с сертификатом в формате PEM для аутентификации на проксируемом HTTPS-сервере. В имени файла можно использовать переменные.

При включённом *proxy_ssl_nTLS* директива принимает два аргумента вместо одного:

```
location /proxy {
    proxy_ssl_nTLS on;

    proxy_ssl_certificate      sign.crt enc.crt;
    proxy_ssl_certificate_key  sign.key enc.key;

    proxy_ssl_ciphers "ECC-SM2-WITH-SM4-SM3:ECDSA-SM2-WITH-SM4-SM3:RSA";

    proxy_pass https://backend:443;
}
```

proxy_ssl_certificate_cache

<i>Синтаксис</i>	proxy_ssl_certificate_cache off; proxy_ssl_certificate_cache max= <i>N</i> [<i>inactive=time</i>] [<i>valid=time</i>];
Значение по умолчанию	proxy_ssl_certificate_cache off;
<i>Контекст</i>	http, server, location

Определяет кэш для хранения *SSL-сертификатов* и *секретных ключей*, заданных через переменные.

Директива поддерживает следующие параметры:

- **max** — устанавливает максимальное количество элементов в кэше. При переполнении кэша удаляются наименее недавно использованные (LRU) элементы.
- **inactive** — определяет время, после которого элемент будет удален, если к нему не было обращений. Значение по умолчанию — 10 секунд.
- **valid** — определяет время, в течение которого элемент кэша считается действительным и может использоваться повторно. Значение по умолчанию — 60 секунд. По истечении этого времени сертификаты перезагружаются или проходят повторную проверку.
- **off** — отключает кэш.

Пример:

```
proxy_ssl_certificate      $proxy_ssl_server_name.crt;
proxy_ssl_certificate_key  $proxy_ssl_server_name.key;
proxy_ssl_certificate_cache max=1000 inactive=20s valid=1m;
```


proxy_ssl_certificate_key

<i>Синтаксис</i>	proxy_ssl_certificate_key <i>файл</i> [<i>файл</i>];
По умолчанию	—
<i>Контекст</i>	http, server, location

Задаёт файл с секретным ключом в формате PEM для аутентификации на проксируемом HTTPS-сервере.

Вместо файла можно указать значение "engine:*имя*:*id*", которое загружает ключ с указанным *id* из OpenSSL engine с заданным именем.

В имени файла можно использовать переменные.

При включённом *proxy_ssl_ntls* директива принимает два аргумента вместо одного:

```
location /proxy {
    proxy_ssl_ntls on;

    proxy_ssl_certificate      sign.crt enc.crt;
    proxy_ssl_certificate_key  sign.key enc.key;

    proxy_ssl_ciphers "ECC-SM2-WITH-SM4-SM3:ECDHE-SM2-WITH-SM4-SM3:RSA";

    proxy_pass https://backend:443;
}
```

proxy_ssl_ciphers

<i>Синтаксис</i>	proxy_ssl_ciphers <i>шифры</i> ;
По умолчанию	proxy_ssl_ciphers DEFAULT;
<i>Контекст</i>	http, server, location

Описывает разрешённые шифры для запросов к проксируемому HTTPS-серверу. Шифры задаются в формате, поддерживаемом библиотекой OpenSSL.

Список шифров зависит от установленной версии OpenSSL. Полный список можно посмотреть с помощью команды `openssl ciphers`.

Предупреждение

Директива `proxy_ssl_ciphers` не настраивает шифры для TLS 1.3 при использовании OpenSSL. Для настройки шифров TLS 1.3 в OpenSSL используйте директиву `proxy_ssl_conf_command`, добавленную для расширенной конфигурации SSL.

- В LibreSSL шифры TLS 1.3 можно настраивать с помощью `proxy_ssl_ciphers`.
- В BoringSSL шифры TLS 1.3 настроить невозможно.

proxy_ssl_conf_command

<i>Синтаксис</i>	proxy_ssl_conf_command <i>имя значение</i> ;
По умолчанию	—
<i>Контекст</i>	http, server, location

Задаёт произвольные конфигурационные команды OpenSSL при установлении соединения с проксируемым HTTPS-сервером.

Примечание

Директива поддерживается при использовании OpenSSL 1.0.2 и выше. Чтобы настроить шифры TLS 1.3 в OpenSSL, используйте команду `ciphersuites`.

На одном уровне может быть указано несколько директив `proxy_ssl_conf_command`. Директивы наследуются с предыдущего уровня конфигурации при условии, что на данном уровне не описаны свои директивы `proxy_ssl_conf_command`.

Предупреждение

Следует учитывать, что изменение настроек OpenSSL напрямую может привести к неожиданному поведению.

proxy_ssl_crl

<i>Синтаксис</i>	proxy_ssl_crl <i>файл</i> ;
По умолчанию	—
<i>Контекст</i>	http, server, location

Указывает файл с отозванными сертификатами (CRL) в формате PEM, используемыми при *проверке* сертификата проксируемого HTTPS-сервера.

proxy_ssl_name

<i>Синтаксис</i>	proxy_ssl_name <i>имя</i> ;
По умолчанию	proxy_ssl_name \$proxy_host;
<i>Контекст</i>	http, server, location

Позволяет переопределить имя сервера, используемое при *проверке* сертификата проксируемого HTTPS-сервера, а также для *передачи его через SNI* при установлении соединения с проксируемым HTTPS-сервером.

По умолчанию используется имя хоста из URL'а, заданного директивой `proxy_pass`.

proxy_ssl_ntls

<i>Синтаксис</i>	proxy_ssl_ntls on off;
По умолчанию	proxy_ssl_ntls off;
<i>Контекст</i>	http, server

Включает клиентскую поддержку NTLS при использовании TLS библиотеки [TongSuo](#).

```
location /proxy {
    proxy_ssl_ntls on;

    proxy_ssl_certificate      sign.crt enc.crt;
    proxy_ssl_certificate_key  sign.key enc.key;

    proxy_ssl_ciphers "ECC-SM2-WITH-SM4-SM3:ECDHE-SM2-WITH-SM4-SM3:RSA";

    proxy_pass https://backend:443;
}
```

Примечание

Angie необходимо собрать с использованием параметра конфигурации `--with-ntls`, с соответствующей SSL библиотекой с поддержкой NTLS

```
./configure --with-openssl=../Tongsuo-8.3.0 \
            --with-openssl-opt=enable-ntls \
            --with-ntls
```

proxy_ssl_password_file

<i>Синтаксис</i>	proxy_ssl_password_file <i>файл</i> ;
По умолчанию	—
<i>Контекст</i>	http, server, location

Задаёт файл с паролями от *секретных ключей*, где каждый пароль указан на отдельной строке. Пароли применяются по очереди в момент загрузки ключа.

proxy_ssl_protocols

<i>Синтаксис</i>	proxy_ssl_protocols [SSLv2] [SSLv3] [TLSv1] [TLSv1.1] [TLSv1.2] [TLSv1.3];
По умолчанию	proxy_ssl_protocols TLSv1.2 TLSv1.3;
<i>Контекст</i>	http, server, location

Изменено в версии 1.2.0: Параметр `TLSv1.3` добавлен к используемым по умолчанию.

Разрешает указанные протоколы для запросов к проксируемому HTTPS-серверу.

proxy_ssl_server_name

<i>Синтаксис</i>	proxy_ssl_server_name on off;
По умолчанию	proxy_ssl_server_name off;
<i>Контекст</i>	http, server, location

Разрешает или запрещает передачу имени сервера, заданного директивой *proxy_ssl_name*, через расширение **Server Name Indication** протокола TLS (SNI, RFC 6066) при установлении соединения с проксируемым HTTPS-сервером.

proxy_ssl_session_reuse

<i>Синтаксис</i>	proxy_ssl_session_reuse on off;
По умолчанию	proxy_ssl_session_reuse on;
<i>Контекст</i>	http, server, location

Определяет, использовать ли повторно SSL-сессии при работе с проксируемым сервером. Если в логах появляются ошибки "*SSL3_GET_FINISHED:digest check failed*", то можно попробовать включить повторное использование сессий.

proxy_ssl_trusted_certificate

<i>Синтаксис</i>	proxy_ssl_trusted_certificate <i>файл</i> ;
По умолчанию	—
<i>Контекст</i>	http, server, location

Задаёт файл с доверенными сертификатами СА в формате PEM, используемыми при *проверке* сертификата проксируемого HTTPS-сервера.

proxy_ssl_verify

<i>Синтаксис</i>	proxy_ssl_verify on off;
По умолчанию	proxy_ssl_verify off;
<i>Контекст</i>	http, server, location

Разрешает или запрещает проверку сертификата проксируемого HTTPS-сервера.

proxy_ssl_verify_depth

<i>Синтаксис</i>	<code>proxy_ssl_verify_depth число;</code>
По умолчанию	<code>proxy_ssl_verify_depth 1;</code>
<i>Контекст</i>	<code>http, server, location</code>

Устанавливает глубину проверки в цепочке сертификатов проксируемого HTTPS-сервера.

proxy_store

<i>Синтаксис</i>	<code>proxy_store on off строка;</code>
По умолчанию	<code>proxy_store off;</code>
<i>Контекст</i>	<code>http, server, location</code>

Разрешает сохранение на диск файлов.

<code>on</code>	сохраняет файлы в соответствии с путями, указанными в директивах <i>alias</i> или <i>root</i>
<code>off</code>	запрещает сохранение файлов

Имя файла можно задать явно с помощью строки с переменными:

```
proxy_store /data/www$original_uri;
```

Время изменения файлов выставляется согласно полученному полю **Last-Modified** в заголовке ответа. Ответ сначала записывается во временный файл, а потом этот файл переименовывается. Временный файл и постоянное место хранения ответа могут располагаться на разных файловых системах. Однако нужно учитывать, что в этом случае вместо дешевой операции переименовывания в пределах одной файловой системы файл копируется с одной файловой системы на другую. Поэтому лучше, если сохраняемые файлы будут находиться на той же файловой системе, что и каталог с временными файлами, задаваемый директивой *proxy_temp_path* для данного *location*.

Директиву можно использовать для создания локальных копий статических неизменяемых файлов, например, так:

```
location /images/ {
    root          /data/www;
    error_page    404 = /fetch$uri;
}

location /fetch/ {
    internal;

    proxy_pass    http://backend/;
    proxy_store    on;
    proxy_store_access user:rw group:rw all:r;
    proxy_temp_path /data/temp;

    alias         /data/www/;
}
```

или так:

```
location /images/ {
    root                /data/www;
    error_page          404 = @fetch;
}

location @fetch {
    internal;

    proxy_pass          http://backend;
    proxy_store          on;
    proxy_store_access  user:rw group:rw all:r;
    proxy_temp_path     /data/temp;

    root                /data/www;
}
```

proxy_store_access

<i>Синтаксис</i>	<code>proxy_store_access пользователи:права ...;</code>
По умолчанию	<code>proxy_store_access user:rw;</code>
<i>Контекст</i>	http, server, location

Задаёт права доступа для создаваемых файлов и каталогов, например,

```
proxy_store_access user:rw group:rw all:r;
```

Если заданы какие-либо права для *group* или *all*, то права для *user* указывать необязательно:

```
proxy_store_access group:rw all:r;
```

proxy_temp_file_write_size

<i>Синтаксис</i>	<code>proxy_temp_file_write_size размер;</code>
По умолчанию	<code>proxy_temp_file_write_size 8k 16k;</code>
<i>Контекст</i>	http, server, location

Ограничивает размер данных, сбрасываемых во временный файл за один раз, при включенной буферизации ответов проксируемого сервера во временные файлы. По умолчанию размер ограничен двумя буферами, заданными директивами *proxy_buffer_size* и *proxy_buffers*. Максимальный размер временного файла задается директивой *proxy_max_temp_file_size*.

proxy_temp_path

<i>Синтаксис</i>	<code>proxy_temp_path путь [уровень1 [уровень2 [уровень3]]]`;</code>
По умолчанию	<code>proxy_temp_path proxy_temp;</code> (путь зависит от параметра сборки <code>--http-proxy-temp-path</code>)
<i>Контекст</i>	<code>http, server, location</code>

Задаёт имя каталога для хранения временных файлов с данными, полученными от проксируемых серверов. В каталоге может использоваться иерархия подкаталогов до трёх уровней. Например, при такой конфигурации

```
proxy_temp_path /spool/angie/proxy_temp 1 2;
```

временный файл будет следующего вида:

```
/spool/angie/proxy_temp/7/45/00000123457
```

См. также параметр `use_temp_path` директивы `proxy_cache_path`.

Встроенные переменные

В модуле `http_proxy` есть встроенные переменные, которые можно использовать для формирования заголовков с помощью директивы `proxy_set_header`:

`$proxy_host`

имя и порт проксируемого сервера, как указано в директиве `proxy_pass`;

`$proxy_port`

порт проксируемого сервера, как указано в директиве `proxy_pass`, или стандартный порт протокола;

`$proxy_add_x_forwarded_for`

поле заголовка запроса клиента `X-Forwarded-For` и добавленная к нему через запятую переменная `$remote_addr`. Если же поля `X-Forwarded-For` в заголовке запроса клиента нет, то переменная `$proxy_add_x_forwarded_for` равна переменной `$remote_addr`.

Random Index

Обслуживает запросы, оканчивающиеся косой чертой (/), и выдает случайный файл в качестве индексного файла каталога. Модуль выполняется до модуля `http_index`.

При сборке из исходного кода модуль не собирается по умолчанию; его необходимо включить с помощью параметра сборки `--with-http-random-index-module`. В пакетах и образах из наших репозиториях модуль включен в сборку.

Пример конфигурации

```
location / {
    random_index on;
}
```

Директивы

random_index

<i>Синтаксис</i>	<code>random_index on off.;</code>
По умолчанию	<code>random_index off;</code>
<i>Контекст</i>	location

Разрешает или запрещает в содержащем эту директиву `location` обработку этим модулем.

RealIP

Позволяет менять адрес и необязательный порт клиента на переданные в указанном поле заголовка.

При сборке из исходного кода модуль не собирается по умолчанию; его необходимо включить с помощью параметра сборки `--with-http_realip_module`. В пакетах и образах из наших репозиториях модуль включен в сборку.

Пример конфигурации

```
set_real_ip_from 192.168.1.0/24;
set_real_ip_from 192.168.2.1;
set_real_ip_from 2001:0db8::/32;
real_ip_header X-Forwarded-For;
real_ip_recursive on;
```

Директивы

set_real_ip_from

<i>Синтаксис</i>	<code>set_real_ip_from адрес CIDR unix.;</code>
По умолчанию	—
<i>Контекст</i>	http, server, location

Задаёт доверенные адреса, которые передают верный адрес для замены. Если указано специальное значение `unix:`, доверенными будут считаться все UNIX-сокеты. Доверенные адреса могут быть также заданы при помощи имени хоста.

real_ip_header

<i>Синтаксис</i>	<code>real_ip_header none X-Real-IP X-Forwarded-For proxy_protocol;</code>
По умолчанию	<code>real_ip_header X-Real-IP;</code>
<i>Контекст</i>	http, server, location

Задаёт поле заголовка запроса, значение которого будет использоваться для замены адреса клиента.

Значение поля заголовка запроса, содержащее необязательный порт, также используется для замены порта клиента. Адрес и порт должны быть указаны согласно [RFC 3986](#).

Параметр `proxy_protocol` меняет адрес клиента на указанный в заголовке PROXY-протокола. Протокол PROXY должен быть предварительно включен при помощи установки параметра `proxy_protocol` в директиве *listen*.

real_ip_recursive

<i>Синтаксис</i>	<code>real_ip_recursive on off;</code>
По умолчанию	<code>real_ip_recursive off;</code>
<i>Контекст</i>	http, server, location

При выключенном рекурсивном поиске исходный адрес клиента, совпадающий с одним из доверенных адресов, заменяется на последний адрес, переданный в поле заголовка запроса, заданного директивой *real_ip_header*. При включенном рекурсивном поиске исходный адрес клиента, совпадающий с одним из доверенных адресов, заменяется на последний не доверенный адрес, переданный в заданном поле заголовка запроса.

Встроенные переменные

`$realip_remote_addr`

хранит исходный адрес клиента

`$realip_remote_port`

хранит исходный порт клиента

Referer

Позволяет блокировать доступ к сайту для запросов с неверными значениями поля **Referer** в заголовке. Следует иметь в виду, что подделать запрос с нужным значением поля **Referer** не составляет большого труда, поэтому цель использования данного модуля заключается не в стопроцентном блокировании подобных запросов, а в блокировании массового потока запросов, сделанных обычными браузерами. Нужно также учитывать, что обычные браузеры могут не передавать поле **Referer** даже для верных запросов.

Пример конфигурации

```
valid_referers none blocked server_names
               *.example.com example.* www.example.org/galleries/
               ~\.google\.;

if ($invalid_referer) {
    return 403;
}
```

Директивы

referer_hash_bucket_size

<i>Синтаксис</i>	referer_hash_bucket_size <i>размер</i> ;
По умолчанию	referer_hash_bucket_size 64;
<i>Контекст</i>	server, location

Задаёт размер корзины хэш-таблиц со значениями **Referer**. Подробнее настройка хэш-таблиц обсуждается *отдельно*.

referer_hash_max_size

<i>Синтаксис</i>	referer_hash_max_size <i>размер</i> ;
По умолчанию	referer_hash_max_size 2048;
<i>Контекст</i>	server, location

Задаёт максимальный размер хэш-таблиц со значениями **Referer**. Подробнее настройка хэш-таблиц обсуждается *отдельно*.

valid_referers

<i>Синтаксис</i>	valid_referers none blocked server_names <i>строка</i> ...;
По умолчанию	—
<i>Контекст</i>	server, location

Задаёт значения поля **Referer** заголовка запроса, при которых встроенная переменная *\$invalid_referer* будет иметь пустую строку в качестве значения. В противном случае значение переменной равно "1". Поиск совпадения производится без учёта регистра символов.

Параметры могут быть следующие:

none	поле Referer в заголовке запроса отсутствует;
blocked	поле Referer в заголовке запроса присутствует, но его значение удалено межсетевым экраном (firewall) или прокси-сервером; к таким значениям относятся строки, не начинающиеся на "http://" или "https://";
server_names	в поле Referer заголовка запроса указано одно из имен сервера;
произвольная строка	задает имя сервера и необязательное начало URI. В начале или конце имени сервера может быть "*". При проверке порт сервера в поле Referer игнорируется;
регулярное выражение	в начале должен быть символ "~". Необходимо учитывать, что на совпадение с выражением будет проверяться текст, начинающийся после "http://" или "https://".

Пример:

```
valid_referers none blocked server_names
               *.example.com example.* www.example.org/galleries/
               ~\.google\.;
```

Встроенные переменные

`$invalid_referer`

Пустая строка, если значение поля **Referer** заголовка запроса считается *правильным*, иначе "1".

Rewrite

Позволяет изменять URI запроса с помощью регулярных выражений PCRE, делать перенаправления и выбирать конфигурацию по условию.

Директивы *break*, *if*, *return*, *rewrite* и *set* обрабатываются в следующем порядке:

- последовательно выполняются директивы этого модуля, описанные на уровне **server**;
- в цикле:
 - ищется **location** по URI запроса;
 - последовательно выполняются директивы этого модуля, описанные в найденном **location**;
 - цикл повторяется, если URI запроса *изменялся*, но *не более* 10 раз.

Директивы

break

<i>Синтаксис</i>	break;
По умолчанию	—
<i>Контекст</i>	server, location, if

Завершает обработку текущего набора директив модуля **http_rewrite**.

Если директива указана внутри **location**, дальнейшая обработка запроса продолжается в этом **location**.

Пример:

```
if ($slow) {
    limit_rate 10k;
    break;
}
```

if

<i>Синтаксис</i>	<code>if (условие) { ... }</code>
По умолчанию	—
<i>Контекст</i>	server, location

Проверяется указанное условие. Если оно истинно, то выполняются указанные в фигурных скобках директивы этого модуля и запросу назначается конфигурация, указанная внутри директивы `if`. Конфигурации внутри директив `if` наследуются с предыдущего уровня конфигурации.

Предупреждение

Хотя внутри блока `if` можно применять директивы других модулей, делать это не рекомендуется, так как это может привести к непредвиденному поведению.

В качестве условия могут быть заданы:

- имя переменной; ложными значениями переменной являются пустая строка или `"0"`;
- сравнение переменной со строкой с помощью операторов `"="` и `"!="`;
- соответствие переменной регулярному выражению с учетом регистра символов — `"~"` и без него — `"~*"`. В регулярных выражениях можно использовать группы захвата, которые затем доступны в виде переменных `$1..$9`. Также можно использовать отрицательные операторы `"!~"` и `"!~*"`. Если в регулярном выражении встречаются символы `"}"` или `";"`, то все выражение следует заключить в одинарные или двойные кавычки.
- проверка существования файла с помощью операторов `"-f"` и `"!-f"`;
- проверка существования каталога с помощью операторов `"-d"` и `"!-d"`;
- проверка существования файла, каталога или символической ссылки с помощью операторов `"-e"` и `"!-e"`;
- проверка исполняемости файла с помощью операторов `"-x"` и `"!-x"`.

Примеры:

```
if ($http_user_agent ~ MSIE) {
    rewrite ^(.*)$ /msie/$1 break;
}

if ($http_cookie ~* "id=(\[^\;]+\)(?:;|$)") {
    set $id $1;
}

if ($request_method = POST) {
    return 405;
}
```

```
if ($slow) {
    limit_rate 10k;
}

if ($invalid_referer) {
    return 403;
}
```

Примечание

Значение встроенной переменной `$invalid_referer` задается директивой `valid_referers`.

return

Синтаксис	<code>return код [текст];</code> <code>return код URL;</code> <code>return URL;</code>
По умолчанию	—
Контекст	server, location, if

Завершает обработку и возвращает клиенту указанный код. Нестандартный код 444 закрывает соединение без передачи заголовка ответа.

Можно задать либо URL перенаправления (для кодов 301, 302, 303, 307 и 308) либо текст тела ответа (для остальных кодов). В тексте тела ответа и URL перенаправления можно использовать переменные. Как частный случай, URL перенаправления может быть задан как URI, локальный для данного сервера, при этом полный URL перенаправления формируется согласно схеме запроса (`$scheme`) и директивам `server_name_in_redirect` и `port_in_redirect`.

Кроме того, в качестве единственного параметра можно указать URL для временного перенаправления с кодом 302. Такой параметр должен начинаться со строк `http://`, `https://` или `"$scheme"`. В URL можно использовать переменные.

См. также директиву `error_page`.

rewrite

Синтаксис	<code>rewrite regex замена [флаг];</code>
По умолчанию	—
Контекст	server, location, if

Если указанное регулярное выражение `regex` соответствует URI запроса, URI изменяется в соответствии со строкой `замены`. Директивы `rewrite` выполняются последовательно, в порядке их следования в конфигурационном файле. С помощью `флага` можно прекратить дальнейшую обработку директив. Если строка `замены` начинается с `http://`, `https://` или `$scheme`, то обработка завершается и клиенту возвращается перенаправление.

Необязательный параметр `флаг` может быть задан одним из следующих значений:

last	завершает обработку текущего набора директив модуля <code>http_rewrite</code> , после чего ищется новый <code>location</code> , соответствующий измененному URI;
break	завершает обработку текущего набора директив модуля <code>http_rewrite</code> аналогично директиве <code>break</code> ;
redirect	возвращает временное перенаправление с кодом 302; используется, если строка замены не начинается с <code>http://</code> , <code>https://</code> или <code>"\$scheme"</code> ;
permanent	возвращает постоянное перенаправление с кодом 301.

Полный URL перенаправлений формируется согласно схеме запроса (`$scheme`) и директив `server_name_in_redirect` и `port_in_redirect`.

Пример:

```
server {
#   ...
    rewrite ^(/download/.*)/media/(.*)\..*$ $1/mp3/$2.mp3 last;
    rewrite ^(/download/.*)/audio/(.*)\..*$ $1/mp3/$2.ra last;
    return 403;
#   ...
}
```

Если же эти директивы поместить в `location "/download/"`, то нужно заменить флаг `last` на `break`, иначе Angie сделает 10 циклов и вернет ошибку 500:

```
location /download/ {
    rewrite ^(/download/.*)/media/(.*)\..*$ $1/mp3/$2.mp3 break;
    rewrite ^(/download/.*)/audio/(.*)\..*$ $1/mp3/$2.ra break;
    return 403;
}
```

Если в строке **замены** указаны новые аргументы запроса, то предыдущие аргументы запроса добавляются после них. Если такое поведение нежелательно, можно отказаться от этого добавления, указав в конце строки замены знак вопроса, например:

```
rewrite ^/users/(.*)$ /show?user=$1? last;
```

Если в регулярном выражении встречаются символы `"}"` или `";"`, то все выражение следует заключить в одинарные или двойные кавычки.

rewrite_log

<i>Синтаксис</i>	<code>rewrite_log on off;</code>
По умолчанию	<code>rewrite_log off;</code>
<i>Контекст</i>	<code>http</code> , <code>server</code> , <code>location</code> , <code>if</code>

Разрешает или запрещает записывать в `error_log` на уровне `notice` результаты обработки директив модуля `http_rewrite`.

set

<i>Синтаксис</i>	<code>set \$переменная значение;</code>
По умолчанию	—
<i>Контекст</i>	server, location, if

Устанавливает значение указанной переменной. В качестве значения можно использовать текст, переменные и их комбинации.

uninitialized_variable_warn

<i>Синтаксис</i>	<code>uninitialized_variable_warn on off;</code>
По умолчанию	<code>uninitialized_variable_warn on;</code>
<i>Контекст</i>	http, server, location, if

Определяет, нужно ли писать в лог предупреждения о неинициализированных переменных.

Внутреннее устройство

Директивы модуля `http_rewrite` компилируются на стадии конфигурации во внутренние инструкции, интерпретируемые во время обработки запроса. Интерпретатор представляет из себя простую стековую виртуальную машину.

Например, директивы

```
location /download/ {
    if ($forbidden) {
        return 403;
    }

    if ($slow) {
        limit_rate 10k;
    }

    rewrite ~/(download/.*)/media/(.*)\..*$ /$1/mp3/$2.mp3 break;
}
```

будут транслированы в такие инструкции:

```
переменная $forbidden
проверка на ноль
    возврат 403
    завершение всего кода
переменная $slow
проверка на ноль
проверка регулярного выражения
копирование "/"
копирование $1
копирование "/mp3/"
копирование $2
копирование ".mp3"
```

завершение регулярного выражения
завершение всего кода

Обратите внимание, что инструкций для директивы `limit_rate` нет, поскольку она не имеет отношения к модулю `http_rewrite`. Для блока `if` создается отдельная конфигурация. Если условие истинно, запрос получает эту конфигурацию, и в ней `limit_rate` равен 10k.

Директиву

```
rewrite ^/(download/.*)/media/(.*)\..*$ /$1/mp3/$2.mp3 break;
```

можно сделать на одну инструкцию меньше, если в регулярном выражении перенести первую косую черту внутрь скобок:

```
rewrite ^(/download/.*)/media/(.*)\..*$ $1/mp3/$2.mp3 break;
```

Тогда соответствующие инструкции будут выглядеть так:

```
проверка регулярного выражения
копирование $1
копирование "/mp3/"
копирование $2
копирование ".mp3"
завершение регулярного выражения
завершение всего кода
```

SCGI

Позволяет передавать запросы SCGI-серверу.

Пример конфигурации

```
location / {
    include    scgi_params;
    scgi_pass  localhost:9000;
}
```

Директивы

scgi_bind

<i>Синтаксис</i>	<code>scgi_bind адрес [transparent] off;</code>
По умолчанию	—
Контекст	http, server, location

Задаёт локальный IP-адрес с необязательным портом, который будет использоваться в исходящих соединениях с SCGI-сервером. В значении параметра допустимо использование переменных. Специальное значение `off` отменяет действие унаследованной с предыдущего уровня конфигурации директивы `scgi_bind`, позволяя системе самостоятельно выбирать локальный IP-адрес и порт.

Параметр `transparent` позволяет задать нелокальный IP-адрес, который будет использоваться в исходящих соединениях с SCGI-сервером, например, реальный IP-адрес клиента:


```
scgi_bind $remote_addr transparent;
```

Для работы параметра обычно требуется запустить рабочие процессы Angie с привилегиями *суперпользователя*. В Linux это не требуется, так как если указан параметр *transparent*, то рабочие процессы наследуют *capability CAP_NET_RAW* из главного процесса.

Примечание

Необходимо настроить таблицу маршрутизации ядра для перехвата сетевого трафика с SCGI-сервера.

scgi_buffer_size

<i>Синтаксис</i>	<code>scgi_buffer_size размер;</code>
По умолчанию	<code>scgi_buffer_size 4k 8k;</code>
<i>Контекст</i>	<code>http, server, location</code>

Задаёт размер буфера, в который будет читаться первая часть ответа, получаемого от SCGI-сервера. В этой части ответа обычно находится небольшой заголовок ответа. По умолчанию размер одного буфера равен размеру страницы памяти. В зависимости от платформы это или 4К, или 8К, однако его можно сделать меньше.

scgi_buffering

<i>Синтаксис</i>	<code>scgi_buffering размер;</code>
По умолчанию	<code>scgi_buffering on;</code>
<i>Контекст</i>	<code>http, server, location</code>

on	Angie принимает ответ SCGI-сервера как можно быстрее, сохраняя его в буферы, заданные директивами <i>scgi_buffer_size</i> и <i>scgi_buffers</i> . Если ответ не влезает целиком в память, то его часть может быть записана на диск во <i>временный файл</i> . Запись во временные файлы контролируется директивами <i>scgi_max_temp_file_size</i> и <i>scgi_temp_file_write_size</i> .
off	Ответ синхронно передается клиенту сразу же по мере его поступления. Angie не пытается считать весь ответ SCGI-сервера. Максимальный размер данных, который Angie может принять от сервера за один раз, задается директивой <i>scgi_buffer_size</i> .

Буферизация может быть также включена или выключена путем передачи значения "yes" или "no" в поле **X-Accel-Buffering** заголовка ответа. Эту возможность можно запретить с помощью директивы *scgi_ignore_headers*.

scgi_buffers

<i>Синтаксис</i>	<code>scgi_buffers</code> <i>число</i> <i>размер</i> ;
По умолчанию	<code>scgi_buffers 8 4k 8k</code> ;
<i>Контекст</i>	http, server, location

Задаёт число и размер буферов для одного соединения, в которые будет читаться ответ, получаемый от SCGI-сервера.

По умолчанию размер одного буфера равен размеру страницы. В зависимости от платформы это или 4К, или 8К.

scgi_busy_buffers_size

<i>Синтаксис</i>	<code>scgi_busy_buffers_size</code> <i>размер</i> ;
По умолчанию	<code>scgi_busy_buffers_size 8k 16k</code> ;
<i>Контекст</i>	http, server, location

При включённой *буферизации* ответов SCGI-сервера, ограничивает суммарный размер буферов, которые могут быть заняты для отправки ответа клиенту, пока ответ ещё не прочитан целиком. Оставшиеся буферы тем временем могут использоваться для чтения ответа и, при необходимости, буферизации части ответа во временный файл.

По умолчанию размер ограничен величиной двух буферов, заданных директивами `scgi_buffer_size` и `scgi_buffers`.

scgi_cache

<i>Синтаксис</i>	<code>scgi_cache</code> <i>зона</i> off;
По умолчанию	<code>scgi_cache off</code> ;
<i>Контекст</i>	http, server, location

Задаёт зону разделяемой памяти, используемой для кэширования. Одна и та же зона может использоваться в нескольких местах. В значении параметра можно использовать переменные.

off	запрещает кэширование, унаследованное с предыдущего уровня конфигурации.

scgi_cache_background_update

<i>Синтаксис</i>	scgi_cache_background_update on off;
По умолчанию	scgi_cache_background_update off;
<i>Контекст</i>	http, server, location

Позволяет запустить фоновый подзапрос для обновления просроченного элемента кэша, в то время как клиенту возвращается устаревший кэшированный ответ.

Предупреждение

Использование устаревшего кэшированного ответа в момент его обновления должно быть *разрешено*.

scgi_cache_bypass

<i>Синтаксис</i>	scgi_cache_bypass ...;
По умолчанию	—
<i>Контекст</i>	http, server, location

Задаёт условия, при которых ответ не будет браться из кэша. Если значение хотя бы одного из строковых параметров непустое и не равно "0", то ответ не берётся из кэша:

```
scgi_cache_bypass $cookie_nocache $arg_nocache$arg_comment;
scgi_cache_bypass $http_pragma $http_authorization;
```

Можно использовать совместно с директивой *scgi_no_cache*.

scgi_cache_key

<i>Синтаксис</i>	scgi_cache_key строка;
По умолчанию	—
<i>Контекст</i>	http, server, location

Задаёт ключ для кэширования, например,

```
scgi_cache_key localhost:9000$request_uri;
```

scgi_cache_lock

<i>Синтаксис</i>	<code>scgi_cache_lock on off;</code>
По умолчанию	<code>scgi_cache_lock off;</code>
<i>Контекст</i>	http, server, location

Если включено, одновременно только одному запросу будет позволено заполнить новый элемент кэша, идентифицируемый согласно директиве *scgi_cache_key*, передав запрос на SCGI-сервер. Остальные запросы этого же элемента будут либо ожидать появления ответа в кэше, либо освобождения блокировки этого элемента, в течение времени, заданного директивой *scgi_cache_lock_timeout*.

scgi_cache_lock_age

<i>Синтаксис</i>	<code>scgi_cache_lock_age время;</code>
По умолчанию	<code>scgi_cache_lock_age 5s;</code>
<i>Контекст</i>	http, server, location

Если последний запрос, переданный на SCGI-сервер для заполнения нового элемента кэша, не завершился за указанное время, на SCGI-сервер может быть передан еще один запрос.

scgi_cache_lock_timeout

<i>Синтаксис</i>	<code>scgi_cache_lock_timeout время;</code>
По умолчанию	<code>scgi_cache_lock_timeout 5s;</code>
<i>Контекст</i>	http, server, location

Задаёт таймаут для *scgi_cache_lock*. По истечении указанного времени запрос будет передан на SCGI-сервер, однако ответ не будет кэширован.

scgi_cache_max_range_offset

<i>Синтаксис</i>	<code>scgi_cache_max_range_offset число;</code>
По умолчанию	—
<i>Контекст</i>	http, server, location

Задаёт смещение в байтах для запросов с указанием диапазона запрашиваемых байт (byte-range requests). Если диапазон находится за указанным смещением, range-запрос будет передан на SCGI-сервер и ответ не будет кэширован.

scgi_cache_methods

<i>Синтаксис</i>	<code>scgi_cache_methods GET HEAD POST ...;</code>
По умолчанию	<code>scgi_cache_methods GET HEAD;</code>
<i>Контекст</i>	http, server, location

Если метод запроса клиента указан в этой директиве, то ответ будет кэширован. Методы "GET" и "HEAD" всегда добавляются в список, но тем не менее рекомендуется перечислять их явно. См. также директиву `scgi_no_cache`.

scgi_cache_min_uses

<i>Синтаксис</i>	<code>scgi_cache_min_uses число;</code>
По умолчанию	<code>scgi_cache_min_uses 1;</code>
<i>Контекст</i>	http, server, location

Задаёт число запросов, после которого ответ будет кэширован.

⚠ Предупреждение

Метаданные кэша хранятся в разделяемой памяти. Ручное удаление файлов кэша не сбрасывает счетчики и может привести к непредсказуемому поведению. Для полного сброса остановите сервер, удалите директорию кэша и запустите снова.

ℹ Примечание

Сторонние модули очистки кэша (например, `external-cache-purge`) удаляют только файлы, но не сбрасывают счетчик `scgi_cache_min_uses`. Директива предназначена для защиты кэша от загрязнения редкими запросами, и сброс счетчика при очистке может негативно повлиять на производительность.

scgi_cache_path

<i>Синтаксис</i>	<code>scgi_cache_path путь [levels=уровни] [use_temp_path=on off] keys_zone=имя:размер [inactive=время] [max_size=размер] [min_free=размер] [manager_files=число] [manager_sleep=время] [manager_threshold=время] [loader_files=число] [loader_sleep=время] [loader_threshold=время];</code>
По умолчанию	—
<i>Контекст</i>	http

Задаёт путь и другие параметры кэша. Данные кэша хранятся в файлах. Именем файла в кэше является результат функции MD5 от *ключа кэширования*.

Параметр `levels` задаёт уровни иерархии кэша: можно задать от 1 до 3 уровней, на каждом уровне допускаются значения 1 или 2.

Например, при использовании

```
scgi_cache_path /data/angie/cache levels=1:2 keys_zone=one:10m;
```

имена файлов в кэше будут такого вида:

```
/data/angie/cache/c/29/b7f54b2df7773722d382f4809d65029c
```

Кэшируемый ответ сначала записывается во временный файл, а потом этот файл переименовывается. Временные файлы и кэш могут располагаться на разных файловых системах. Однако нужно учитывать, что в этом случае вместо дешевой операции переименовывания в пределах одной файловой системы файл копируется с одной файловой системы на другую. Поэтому лучше, если кэш будет находиться на той же файловой системе, что и каталог с временными файлами.

Какой из каталогов будет использоваться для временных файлов, определяется параметром `use_temp_path`.

on	Если параметр не задан или установлен в значение "on", будет использоваться каталог, задаваемый директивой <code>scgi_temp_path</code> для данного <code>location</code>
off	временные файлы будут располагаться непосредственно в каталоге кэша

Кроме того, все активные ключи и информация о данных хранятся в зоне разделяемой памяти, имя и размер которой задаются параметром `keys_zone`. Зоны размером в 1 мегабайт достаточно для хранения около 8 тысяч ключей. Метаданные кэша хранятся в разделяемой памяти.

Если к данным кэша не обращаются в течение времени, заданного параметром `inactive`, то данные удаляются, независимо от их свежести.

По умолчанию `inactive` равен 10 минутам.

Специальный процесс **менеджера кэша** следит за максимальным размером кэша, а также за минимальным объемом свободного места на файловой системе с кэшем, и удаляет наименее востребованные данные при превышении максимального размера кэша или недостаточном объеме свободного места. Удаление данных происходит итерациями.

max_size	максимальное пороговое значение размера кэша
min_free	минимальное пороговое значение объема свободного места на файловой системе с кэшем
manager_files	максимальное количество удаляемых элементов кэша за одну итерацию По умолчанию: 100
manager_threshold	ограничивает время работы одной итерации По умолчанию: 200 миллисекунд
manager_sleep	время, в течение которого выдерживается пауза между итерациями По умолчанию: 50 миллисекунд

Через минуту после старта Angie активируется специальный процесс **загрузчика кэша**, который загружает в зону кэша информацию о ранее кэшированных данных, хранящихся на файловой системе. Загрузка также происходит итерациями.

loader_files	максимальное количество элементов кэша к загрузке в одну итерацию По умолчанию: 100
loader_threshold	ограничивает время работы одной итерации По умолчанию: 200 миллисекунд
loader_sleep	время, в течение которого выдерживается пауза между итерациями По умолчанию: 50 миллисекунд

scgi_cache_revalidate

<i>Синтаксис</i>	scgi_cache_revalidate on off;
По умолчанию	scgi_cache_revalidate off;
<i>Контекст</i>	http, server, location

Разрешает ревалидацию просроченных элементов кэша при помощи условных запросов с полями заголовка If-Modified-Since и If-None-Match .

scgi_cache_use_stale

<i>Синтаксис</i>	scgi_cache_use_stale error timeout invalid_header updating http_500 http_503 http_403 http_404 http_429 off ...;
По умолчанию	scgi_cache_use_stale off;
<i>Контекст</i>	http, server, location

Определяет, в каких случаях можно использовать устаревший кэшированный ответ. Параметры директивы совпадают с параметрами директивы *scgi_next_upstream*.

error	Позволяет использовать устаревший кэшированный ответ при невозможности выбора SCGI-сервера для обработки запроса.
updating	Дополнительный параметр, разрешает использовать устаревший кэшированный ответ, если на данный момент он уже обновляется. Это позволяет минимизировать число обращений к SCGI-серверам при обновлении кэшированных данных.

Использование устаревшего кэшированного ответа может также быть разрешено непосредственно в заголовке ответа на определенное количество секунд после того, как ответ устарел:

- Расширение **stale-while-revalidate** поля заголовка **Cache-Control** разрешает использовать устаревший кэшированный ответ, если на данный момент он уже обновляется.
- Расширение **stale-if-error** поля заголовка **Cache-Control** разрешает использовать устаревший кэшированный ответ в случае ошибки.

Примечание

Такой способ менее приоритетен, чем задание параметров директивы.

Чтобы минимизировать число обращений к SCGI-серверам при заполнении нового элемента кэша, можно воспользоваться директивой *scgi_cache_lock*.

scgi_cache_valid

<i>Синтаксис</i>	<code>scgi_cache_valid [код ...] время;</code>
По умолчанию	—
<i>Контекст</i>	http, server, location

Задаёт время кэширования для разных кодов ответа. Например, директивы

```
scgi_cache_valid 200 302 10m;
scgi_cache_valid 404 1m;
```

задают время кэширования 10 минут для ответов с кодами 200 и 302 и 1 минуту для ответов с кодом 404.

Если указано только время кэширования,

```
scgi_cache_valid 5m;
```

то кэшируются только ответы 200, 301 и 302.

Кроме того, можно кэшировать любые ответы с помощью параметра `any`:

```
scgi_cache_valid 200 302 10m;
scgi_cache_valid 301 1h;
scgi_cache_valid any 1m;
```

Примечание

Параметры кэширования могут также быть заданы непосредственно в заголовке ответа. Такой способ приоритетнее, чем задание времени кэширования с помощью директивы.

- Поле заголовка `X-Accel-Expires` задаёт время кэширования ответа в секундах. Значение `0` запрещает кэшировать ответ. Если значение начинается с префикса `@`, оно задаёт абсолютное время в секундах с начала эпохи, до которого ответ может быть кэширован.
- Если в заголовке нет поля `X-Accel-Expires`, параметры кэширования определяются по полям заголовка `Expires` или `Cache-Control`.
- Ответ, в заголовке которого есть поле `Set-Cookie`, не будет кэшироваться.
- Ответ, в заголовке которого есть поле `Vary` со специальным значением `"*"`, не будет кэшироваться. Ответ, в заголовке которого есть поле `Vary` с другим значением, будет кэширован с учетом соответствующих полей заголовка запроса.

Обработка одного или более из этих полей заголовка может быть отключена при помощи директивы `scgi_ignore_headers`.

scgi_connect_timeout

<i>Синтаксис</i>	scgi_connect_timeout <i>время</i> ;
По умолчанию	scgi_connect_timeout 60s;
<i>Контекст</i>	http, server, location

Задаёт таймаут для установления соединения с SCGI-сервером. Необходимо иметь в виду, что этот таймаут обычно не может превышать 75 секунд.

scgi_connection_drop

<i>Синтаксис</i>	scgi_connection_drop <i>время</i> on off;
По умолчанию	scgi_connection_drop off;
<i>Контекст</i>	http, server, location

Настраивает завершение всех соединений с проксируемым сервером, если он был удален из группы или помечен как постоянно недоступный в результате процесса *resolve* или команды *API DELETE*.

Соединение завершается, когда обрабатывается следующее событие чтения или записи для клиента или проксируемого сервера.

Установка *времени* включает *таймаут* до завершения соединения; при выборе значения *on* соединения завершаются немедленно.

scgi_force_ranges

<i>Синтаксис</i>	scgi_force_ranges off;
По умолчанию	scgi_force_ranges off;
<i>Контекст</i>	http, server, location

Включает поддержку диапазонов запрашиваемых байт (byte-range) для кэшированных и некешированных ответов SCGI-сервера вне зависимости от наличия поля *Accept-Range* в заголовках этих ответов.

scgi_hide_header

<i>Синтаксис</i>	scgi_hide_header <i>поле</i> ;
По умолчанию	—
<i>Контекст</i>	http, server, location

По умолчанию Angie не передает клиенту поля заголовка *Status* и *X-Accel-...* из ответа SCGI-сервера. Директива *scgi_hide_header* задает дополнительные поля, которые не будут передаваться. Если же передачу полей нужно разрешить, можно воспользоваться директивой *scgi_pass_header*.

scgi_ignore_client_abort

<i>Синтаксис</i>	scgi_ignore_client_abort on off;
По умолчанию	scgi_ignore_client_abort off;
<i>Контекст</i>	http, server, location

Определяет, закрывать ли соединение с SCGI-сервером в случае, если клиент закрыл соединение, не дождавшись ответа.

scgi_ignore_headers

<i>Синтаксис</i>	scgi_ignore_headers поле ...;
По умолчанию	—
<i>Контекст</i>	http, server, location

Запрещает обработку некоторых полей заголовка из ответа SCGI-сервера. В директиве можно указать поля X-Accel-Redirect, X-Accel-Expires, X-Accel-Limit-Rate, X-Accel-Buffering, X-Accel-Charset, Expires, Cache-Control, Set-Cookie и Vary.

Если не запрещено, обработка этих полей заголовка заключается в следующем:

- X-Accel-Expires, Expires, Cache-Control, Set-Cookie и Vary задают *параметры кэширования* ответа;
- X-Accel-Redirect производит *внутреннее перенаправление* на указанный URI;
- X-Accel-Limit-Rate задает *ограничение скорости* передачи ответа клиенту;
- X-Accel-Buffering включает или выключает *буферизацию* ответа;
- X-Accel-Charset задает желаемую *кодировку* ответа.

scgi_intercept_errors

<i>Синтаксис</i>	scgi_intercept_errors on off;
По умолчанию	scgi_intercept_errors off;
<i>Контекст</i>	http, server, location

Определяет, передавать ли клиенту ответы SCGI-сервера с кодом больше либо равным 300, или же перехватывать их и перенаправлять на обработку Angie с помощью директивы *error_page*.

scgi_limit_rate

<i>Синтаксис</i>	scgi_limit_rate <i>скорость</i> ;
По умолчанию	scgi_limit_rate 0;
<i>Контекст</i>	http, server, location

Ограничивает скорость чтения ответа от SCGI-сервера. *Скорость* задается в байтах в секунду; можно использовать переменные.

0	отключает ограничение скорости

Примечание

Ограничение устанавливается на запрос, поэтому, если Angie одновременно откроет два соединения к SCGI-серверу, суммарная скорость будет вдвое выше заданного ограничения. Ограничение работает только в случае, если включена *буферизация* ответов SCGI-сервера.

scgi_max_temp_file_size

<i>Синтаксис</i>	scgi_max_temp_file_size <i>размер</i> ;
По умолчанию	scgi_max_temp_file_size 1024m;
<i>Контекст</i>	http, server, location

Если включена *буферизация* ответов SCGI-сервера, и ответ не помещается целиком в буферы, заданные директивами *scgi_buffer_size* и *scgi_buffers*, часть ответа может быть записана во временный файл. Эта директива задает максимальный размер временного файла. Размер данных, сбрасываемых во временный файл за один раз, задается директивой *scgi_temp_file_write_size*.

0	отключает возможность буферизации ответов во временные файлы

Примечание

Данное ограничение не распространяется на ответы, которые будут *кэшированы* или сохранены на диске.

scgi_next_upstream

<i>Синтаксис</i>	scgi_next_upstream error timeout invalid_header http_500 http_503 http_403 http_404 http_429 non_idempotent off ...;
По умолчанию	scgi_next_upstream error timeout;
<i>Контекст</i>	http, server, location

Определяет, в каких случаях запрос будет передан следующему серверу:

error	произошла ошибка соединения с сервером, передачи ему запроса или чтения заголовка ответа сервера;
timeout	произошел таймаут во время соединения с сервером, передачи ему запроса или чтения заголовка ответа сервера;
invalid_header	сервер вернул пустой или неверный ответ;
http_500	сервер вернул ответ с кодом 500;
http_503	сервер вернул ответ с кодом 503;
http_403	сервер вернул ответ с кодом 403;
http_404	сервер вернул ответ с кодом 404;
http_429	сервер вернул ответ с кодом 429;
non_idempotent	обычно запросы с неидемпотентным методом (<i>POST</i> , <i>LOCK</i> , <i>PATCH</i>) не передаются на другой сервер, если запрос серверу группы уже был отправлен; включение параметра явно разрешает повторять подобные запросы;
off	запрещает передачу запроса следующему серверу.

Примечание

Необходимо понимать, что передача запроса следующему серверу возможна только при условии, что клиенту еще ничего не передавалось. То есть, если ошибка или таймаут возникли в середине передачи ответа клиенту, то действие директивы на такой запрос не распространяется.

Директива также определяет, что считается *неудачной попыткой* работы с сервером.

error	всегда считаются неудачными попытками, даже если они не указаны в директиве
timeout	
invalid_header	
http_500	считаются неудачными попытками, только если они указаны в директиве
http_503	
http_429	
http_403	никогда не считаются неудачными попытками
http_404	

Передача запроса следующему серверу может быть ограничена по *количеству попыток* и по *времени*.

scgi_next_upstream_timeout

<i>Синтаксис</i>	scgi_next_upstream_timeout <i>время</i> ;
По умолчанию	scgi_next_upstream_timeout 0;
<i>Контекст</i>	http, server, location

Ограничивает время, в течение которого возможна передача запроса *следующему* серверу.

0	отключает это ограничение

scgi_next_upstream_tries

<i>Синтаксис</i>	<code>scgi_next_upstream_tries</code> <i>число</i> ;
По умолчанию	<code>scgi_next_upstream_tries 0</code> ;
<i>Контекст</i>	http, server, location

Ограничивает число допустимых попыток для передачи запроса *следующему* серверу.

0	отключает это ограничение

scgi_no_cache

<i>Синтаксис</i>	<code>scgi_no_cache</code> <i>строка ...</i> ;
По умолчанию	—
<i>Контекст</i>	http, server, location

Задаёт условия, при которых ответ не будет сохраняться в кэш. Если значение хотя бы одного из строковых параметров непустое и не равно "0", то ответ не будет сохранен:

```
scgi_no_cache $cookie_nocache $arg_nocache$arg_comment;
scgi_no_cache $http_pragma $http_authorization;
```

Можно использовать совместно с директивой `scgi_cache_bypass`.

scgi_param

<i>Синтаксис</i>	<code>scgi_param</code> <i>параметр значение</i> [<i>if_not_empty</i>];
По умолчанию	—
<i>Контекст</i>	http, server, location

Задаёт параметр, который будет передаваться SCGI-серверу. В качестве значения можно использовать текст, переменные и их комбинации. Директивы наследуются с предыдущего уровня конфигурации при условии, что на данном уровне не описаны свои директивы *SCGI-param*.

Стандартные *переменные окружения CGI* должны передаваться как заголовки SCGI, см. файл *scgi_params* из дистрибутива:

```
location / {
    include scgi_params;
    # ...
}
```

Если директива указана с *if_not_empty*, то такой параметр с пустым значением передаваться на сервер не будет:

```
scgi_param HTTPS $https if_not_empty;
```

scgi_pass

<i>Синтаксис</i>	<code>scgi_pass uri;</code>
По умолчанию	—
<i>Контекст</i>	location, if в location

Задаёт адрес SCGI-сервера. Адрес может быть указан в виде доменного имени или IP-адреса, и порта:

```
scgi_pass localhost:9000;
```

или в виде пути UNIX-сокета:

```
scgi_pass unix:/tmp/scgi.socket;
```

Если доменному имени соответствует несколько адресов, то все они будут использоваться по очереди (round-robin). Кроме того, в качестве адреса можно указать *группу серверов*.

В значении параметра можно использовать переменные. В этом случае, если адрес указан в виде доменного имени, имя ищется среди описанных групп серверов и если не найдено, то определяется с помощью *resolver*'а.

Примечание

Если `scgi_pass` стоит в `location` с косой чертой в конце префикса (например, `location /name/`), и при этом в директиве `auto_redirect` указано `default`, запросы без косой черты в конце будут перенаправляться (`/name -> /name/`).

scgi_pass_header

<i>Синтаксис</i>	<code>scgi_pass_header поле ...;</code>
По умолчанию	—
<i>Контекст</i>	http, server, location

Разрешает передавать от SCGI-сервера клиенту *запрещенные для передачи* поля заголовка.

scgi_pass_request_body

<i>Синтаксис</i>	<code>scgi_pass_request_body on off;</code>
По умолчанию	<code>scgi_pass_request_body on;</code>
<i>Контекст</i>	http, server, location

Позволяет запретить передачу исходного тела запроса на SCGI-сервер. См. также директиву `scgi_pass_request_headers`.

scgi_pass_request_headers

<i>Синтаксис</i>	<code>scgi_pass_request_headers on off;</code>
По умолчанию	<code>scgi_pass_request_headers on;</code>
<i>Контекст</i>	http, server, location

Позволяет запретить передачу полей заголовка исходного запроса на SCGI-сервер. См. также директиву `scgi_pass_request_body`.

scgi_read_timeout

<i>Синтаксис</i>	<code>scgi_read_timeout время;</code>
По умолчанию	<code>scgi_read_timeout 60s;</code>
<i>Контекст</i>	http, server, location

Задаёт таймаут при чтении ответа SCGI-сервера. Таймаут устанавливается не на всю передачу ответа, а только между двумя операциями чтения. Если по истечении этого времени SCGI-сервер ничего не передаст, соединение закрывается.

scgi_request_buffering

<i>Синтаксис</i>	<code>scgi_request_buffering on off;</code>
По умолчанию	<code>scgi_request_buffering on;</code>
<i>Контекст</i>	http, server, location

Разрешает или запрещает использовать буферизацию тела запроса клиента.

on	тело запроса полностью <i>читается</i> от клиента перед отправкой запроса на SCGI-сервер.
off	тело запроса отправляется на SCGI-сервер сразу же по мере его поступления. В этом случае запрос не может быть передан <i>следующему серверу</i> , если Angie уже начал отправку тела запроса.

Если для отправки тела исходного запроса используется HTTP/1.1 и передача данных частями (chunked transfer encoding), то тело запроса буферизуется независимо от значения директивы.

scgi_send_timeout

<i>Синтаксис</i>	<code>scgi_send_timeout время;</code>
По умолчанию	<code>scgi_send_timeout 60s;</code>
<i>Контекст</i>	http, server, location

Задаёт таймаут при передаче запроса SCGI-серверу. Таймаут устанавливается не на всю передачу запроса, а только между двумя операциями записи. Если по истечении этого времени SCGI-сервер не примет новых данных, соединение закрывается.

scgi_socket_keepalive

<i>Синтаксис</i>	scgi_socket_keepalive on off;
По умолчанию	scgi_socket_keepalive off;
<i>Контекст</i>	http, server, location

Конфигурирует поведение "TCP keepalive" для исходящих соединений к SCGI-серверу.

" "	По умолчанию для сокета действуют настройки операционной системы.
on	для сокета включается параметр <i>SO_KEEPALIVE</i>

scgi_store

<i>Синтаксис</i>	scgi_store on off <i>строка</i> ;
По умолчанию	scgi_store off;
<i>Контекст</i>	http, server, location

Разрешает сохранение на диск файлов.

on	сохраняет файлы в соответствии с путями, указанными в директивах <i>alias</i> или <i>root</i>
off	запрещает сохранение файлов

Имя файла можно задать явно с помощью строки с переменными:

```
scgi_store /data/www$original_uri;
```

Время изменения файлов выставляется согласно полученному полю Last-Modified в заголовке ответа. Ответ сначала записывается во временный файл, а потом этот файл переименовывается. Временный файл и постоянное место хранения ответа могут располагаться на разных файловых системах. Однако нужно учитывать, что в этом случае вместо дешевой операции переименовывания в пределах одной файловой системы файл копируется с одной файловой системы на другую. Поэтому лучше, если сохраняемые файлы будут находиться на той же файловой системе, что и каталог с временными файлами, задаваемый директивой *scgi_temp_path* для данного *location*.

Директиву можно использовать для создания локальных копий статических неизменяемых файлов:

```
location /images/ {
    root          /data/www;
    error_page    404 = /fetch$uri;
}

location /fetch/ {
    internal;

    scgi_pass     backend:9000;
    # ...

    scgi_store    on;
    scgi_store_access user:rw group:rw all:r;
    scgi_temp_path /data/temp;
```



```
alias /data/www/;
}
```

scgi_store_access

<i>Синтаксис</i>	<code>scgi_store_access пользователи:права ...;</code>
По умолчанию	<code>scgi_store_access user:rw;</code>
<i>Контекст</i>	http, server, location

Задаёт права доступа для создаваемых файлов и каталогов, например,

```
scgi_store_access user:rw group:rw all:r;
```

Если заданы какие-либо права для *group* или *all*, то права для *user* указывать необязательно:

```
scgi_store_access group:rw all:r;
```

scgi_temp_file_write_size

<i>Синтаксис</i>	<code>scgi_temp_file_write_size размер;</code>
По умолчанию	<code>scgi_temp_file_write_size 8k 16k;</code>
<i>Контекст</i>	http, server, location

Ограничивает размер данных, сбрасываемых во временный файл за один раз, при включенной буферизации ответов SCGI-сервера во временные файлы. По умолчанию размер ограничен двумя буферами, заданными директивами *scgi_buffer_size* и *scgi_buffers*. Максимальный размер временного файла задается директивой *scgi_max_temp_file_size*.

scgi_temp_path

<i>Синтаксис</i>	<code>scgi_temp_path путь [уровень1 [уровень2 [уровень3]]]`;</code>
По умолчанию	<code>scgi_temp_path scgi_temp;</code> (путь зависит от параметра сборки <code>--http-scgi-temp-path</code>)
<i>Контекст</i>	http, server, location

Задаёт имя каталога для хранения временных файлов с данными, полученными от SCGI-серверов. В каталоге может использоваться иерархия подкаталогов до трех уровней. Например, при такой конфигурации

```
scgi_temp_path /spool/angie/scgi_temp 1 2;
```

временный файл будет следующего вида:

```
/spool/angie/scgi_temp/7/45/00000123457
```

См. также параметр *use_temp_path* директивы *scgi_cache_path*.

Secure Link

Позволяет проверять аутентичность запрашиваемых ссылок, защищать ресурсы от несанкционированного доступа, а также ограничивать срок действия ссылок.

Правильность запрашиваемой ссылки проверяется сравнением переданного в запросе значения контрольной суммы со значением, вычисляемым для запроса. Если ссылка имеет ограниченный срок действия и он истек, ссылка считается устаревшей. Результат этих проверок делается доступным в переменной `$secure_link`.

Модуль реализует два альтернативных режима работы. В первом режиме, который включается директивой `secure_link_secret`, можно проверить аутентичность запрашиваемых ссылок и защитить их от несанкционированного доступа. Второй режим включается директивами `secure_link` и `secure_link_md5`, и позволяет также ограничить срок действия ссылок.

При сборке из исходного кода модуль не собирается по умолчанию; его необходимо включить с помощью параметра сборки `--with-http_secure_link_module`. В пакетах и образах из наших репозиториях модуль включен в сборку.

Директивы

`secure_link`

<i>Синтаксис</i>	<code>secure_link выражение;</code>
По умолчанию	—
<i>Контекст</i>	http, server, location

Задаёт строку с переменными, из которой будет выделено значение контрольной суммы и время действия ссылки.

Используемые в выражении переменные обычно связаны с запросом; см. *пример* ниже.

Выделенное из строки значение контрольной суммы сравнивается со значением MD5-хэша, вычисляемым для выражения, заданного директивой `secure_link_md5`.

Если контрольные суммы не совпадают, значением переменной `$secure_link` становится пустая строка. Если контрольные суммы совпадают, проверяется время действия ссылки.

Если срок действия ссылки задан и истек, переменная `$secure_link` получает значение 0. В противном случае она получает значение 1. Значение MD5-хэша передается в запросе закодированным в `base64url`.

Если ссылка имеет ограниченный срок действия, время ее действия задается в секундах с начала эпохи (1 января 1970 года 00:00:00 GMT). Значение указывается в выражении после MD5-хэша и отделяется от него запятой. Время действия ссылки, переданное в запросе, делается доступным в переменной `$secure_link_expires` для использования в директиве `secure_link_md5`. Если время действия ссылки не задано, ссылка имеет неограниченный срок действия.

secure_link_md5

<i>Синтаксис</i>	<code>secure_link_md5</code> <i>выражение</i> ;
По умолчанию	—
<i>Контекст</i>	http, server, location

Задаёт выражение, для которого считается значение MD5-хэша, сравниваемое с переданным в запросе.

Выражение должно содержать защищаемую часть ссылки (ресурс) и секретную составляющую. Если ссылка имеет ограниченный срок действия, выражение также должно содержать `$secure_link_expires`.

Для предотвращения несанкционированного доступа выражение может содержать информацию о клиенте, например, его адрес и версию браузера.

Пример:

```
location /s/ {
    secure_link $arg_md5,$arg_expires;
    secure_link_md5 "$secure_link_expires$uri$remote_addr secret";

    if ($secure_link = "") {
        return 403;
    }

    if ($secure_link = "0") {
        return 410;
    }

    # ...
}
```

Ссылка `"/s/link?md5=_e4Nc3iduzkWRm01TBBNYw&expires=2147483647"` ограничивает доступ к `"/s/link"` для клиента с IP-адресом 127.0.0.1. Ссылка также имеет ограниченный срок действия до 19 января 2038 года (GMT).

Значение аргумента запроса md5 на UNIX можно получить так:

```
echo -n '2147483647/s/link127.0.0.1 secret' | \
    openssl md5 -binary | openssl base64 | tr +/ -_ | tr -d =
```

secure_link_secret

<i>Синтаксис</i>	<code>secure_link_secret</code> <i>слово</i> ;
По умолчанию	—
<i>Контекст</i>	location

Задаёт секретное слово для проверки аутентичности запрашиваемых ссылок.

Полный URI запрашиваемой ссылки выглядит так:

```
/префикс/хэш/ссылка
```

где хэш — MD5-хэш в шестнадцатеричном виде, вычисленный для конкатенации ссылки и секретного слова, а префикс — произвольная строка без косых черт.

Если запрашиваемая ссылка проходит проверку на аутентичность, значением переменной `$secure_link` становится ссылка, выделенная из URI запроса. В противном случае значением переменной `$secure_link` становится пустая строка.

Пример:

```
location /p/ {
    secure_link_secret secret;

    if ($secure_link = "") {
        return 403;
    }

    rewrite ^ /secure/$secure_link;
}

location /secure/ {
    internal;
}
```

По запросу `"/p/5e814704a28d9bc1914ff19fa0c4a00a/link"` будет выполнено внутреннее перенаправление на `"/secure/link"`.

Значение хэша для данного примера на UNIX можно получить так:

```
echo -n 'linksecret' | openssl md5 -hex
```

Встроенные переменные

`$secure_link`

Результат проверки ссылки. Конкретное значение зависит от выбранного режима работы.

`$secure_link_expires`

Время действия ссылки, переданное в запросе. Предназначено исключительно для использования в директиве `secure_link_md5`.

Slice

Фильтр, который разбивает запрос на подзапросы, каждый из которых возвращает определенный диапазон ответа. Фильтр обеспечивает более эффективное кэширование больших ответов.

При сборке из исходного кода модуль не собирается по умолчанию; его необходимо включить с помощью параметра сборки `--with-http_slice_module`. В пакетах и образах из наших репозиториев модуль включен в сборку.

Пример конфигурации

```
location / {
    slice          1m;
    proxy_cache    cache;
    proxy_cache_key $uri$is_args$args$slice_range;
    proxy_set_header Range $slice_range;
    proxy_cache_valid 200 206 1h;
    proxy_pass      http://localhost:8000;
}
```

В данном примере ответ разбивается на кэшируемые фрагменты размером в 1 мегабайт.

Директивы

slice

<i>Синтаксис</i>	<code>slice размер;</code>
По умолчанию	<code>slice 0;</code>
<i>Контекст</i>	http, server, location

Задаёт размер фрагмента. Нулевое значение запрещает разбиение ответов на фрагменты.

Предупреждение

Обратите внимание, что слишком низкое значение может привести к излишнему потреблению памяти и открытию большого количества файлов.

Для того, чтобы подзапрос вернул необходимый диапазон, переменная `$slice_range` должна быть передана на проксируемый сервер в качестве поля **Range** заголовка запроса. Если включено *кэширование*, то необходимо добавить `$slice_range` в *ключ кэширования* и *включить* кэширование ответов с кодом 206.

Встроенные переменные

`$slice_range`

текущий диапазон фрагмента в формате HTTP `byte range`, например `bytes=0-1048575`.

Split Clients

Модуль генерирует переменные для А/В-тестирования, канареечных релизов и других сценариев, которые направляют определенный процент клиентов на один сервер или конфигурацию, а остальных — куда-то еще.

Пример конфигурации

```
http {
    split_clients "${remote_addr}AAA" $variant {
        0.5% .one;
        2.0% .two;
        *    "";
    }

    server {
        location / {
            index index${variant}.html;
        }
    }
}
```

Директивы

split_clients

<i>Синтаксис</i>	<code>split_clients строка \$переменная { ... }</code>
По умолчанию	—
Контекст	http

Создает *\$переменную*, хэшируя *строку*; переменные в *строке* подставляются, результат хэшируется, затем по значению хэша выбирается строковое значение *\$переменной*.

Функция хэширования использует [MurmurHash2](#) (32 бит), и весь диапазон ее значений (с 0 по 4294967295) сопоставляется с корзинами в порядке появления; процентные величины определяют размер корзин. В конце может стоять метасимвол (*); хэши, не попавшие в другие корзины, сопоставляются с приданным ему значением.

Пример:

```
split_clients "${remote_addr}AAA" $variant {
    0.5% .one;
    2.0% .two;
    *    "";
}
```

Здесь после подстановки в строке `$remote_addrAAA` значения хэша распределяются следующим образом:

- значения от 0 до 21474835 (0,5%) дают `.one`;
- значения от 21474836 до 107374180 (2%) дают `.two`;
- значения от 107374181 до 4294967295 (все остальные) дают `""` (пустую строку).

SSI

Фильтр, обрабатывающий команды SSI (Server Side Includes) в проходящих через него ответах.

Пример конфигурации

```
location / {
    ssi on;
#    ...
}
```

Директивы

ssi

<i>Синтаксис</i>	<code>ssi on off;</code>
По умолчанию	<code>ssi off;</code>
<i>Контекст</i>	http, server, location, if в location

Разрешает или запрещает обработку команд SSI в ответах.

ssi_last_modified

<i>Синтаксис</i>	<code>ssi_last_modified on off;</code>
По умолчанию	<code>ssi_last_modified off;</code>
<i>Контекст</i>	http, server, location

Позволяет сохранить поле заголовка **Last-Modified** исходного ответа во время обработки SSI для лучшего кэширования ответов.

По умолчанию поле заголовка удаляется, так как содержимое ответа изменяется во время обработки и может содержать динамически созданные элементы или части, которые изменились независимо от исходного ответа.

ssi_min_file_chunk

<i>Синтаксис</i>	<code>ssi_min_file_chunk размер;</code>
По умолчанию	<code>ssi_min_file_chunk 1k;</code>
<i>Контекст</i>	http, server, location

Задаёт минимальный размер частей ответа, хранящихся на диске, начиная с которого имеет смысл посылать их с помощью *sendfile*.

ssi_silent_errors

Синтаксис	<code>ssi_silent_errors on off;</code>
По умолчанию	<code>ssi_silent_errors off;</code>
Контекст	http, server, location

Разрешает не выводить строку "*[an error occurred while processing the directive]*", если во время обработки SSI произошла ошибка.

ssi_types

Синтаксис	<code>ssi_types mime-mun ...;</code>
По умолчанию	<code>ssi_types text/html;</code>
Контекст	http, server, location

Разрешает обработку команд SSI в ответах с указанными MIME-типами в дополнение к `text/html`. Специальное значение "*" соответствует любому MIME-типу.

ssi_value_length

Синтаксис	<code>ssi_value_length длина;</code>
По умолчанию	<code>ssi_value_length 256;</code>
Контекст	http, server, location

Задаёт максимальную длину значений параметров в SSI-командах.

Команды SSI

Общий формат команд SSI такой:

```
<!--# команда параметр1=значение1 параметр2=значение2 ... -->
```

Поддерживаются следующие команды:

block

Описывает блок, который можно использовать как заглушку в команде *include*. Внутри блока могут быть другие команды SSI. Параметр команды:

`name`

имя блока.

Пример:

```
<!--# block name="one" -->
заглушка
<!--# endblock -->
```

`config`

Задаёт некоторые параметры, используемые при обработке SSI, а именно:

`errmsg`

строка, выводимая при ошибке во время обработки SSI. По умолчанию выводится такая строка:

```
`[an error occurred while processing the directive]`
```

`timefmt`

строка формата, передаваемая функции `strftime()` для вывода даты и времени. По умолчанию используется такой формат:

```
`"%A, %d-%b-%Y %H:%M:%S %Z" `
```

Для вывода времени в секундах подходит формат `"%s"`.

`echo`

Выводит значение переменной. Параметры команды:

`var`

имя переменной.

`encoding`

способ кодирования. Возможны три значения — `none`, `url` и `entity`. По умолчанию используется `entity`.

default

нестандартный параметр, задающий строку, которая выводится, если переменная не определена. По умолчанию выводится строка (**none**).

Команда

```
<!--# echo var="name" default="нет" -->
```

заменяет такую последовательность команд:

```
<!--# if expr="$name" --><!--# echo var="name" --><!--#  
    else -->нет<!--# endif -->
```

if

Выполняет условное включение. Поддерживаются следующие команды:

```
<!--# if expr="..." -->  
...  
<!--# elif expr="..." -->  
...  
<!--# else -->  
...  
<!--# endif -->
```

На данный момент поддерживается только один уровень вложенности. Параметр команды:

expr

выражение. В выражении может быть:

- проверка существования переменной:

```
<!--# if expr="$name" -->
```

- сравнение переменной с текстом:

```
<!--# if expr="$name = text" -->  
<!--# if expr="$name != text" -->
```

- сравнение переменной с регулярным выражением:

```
<!--# if expr="$name = /text/" -->  
<!--# if expr="$name != /text/" -->
```

Если в *text* встречаются переменные, то производится подстановка их значений. В регулярном выражении можно задать позиционные и именованные группы захвата, а затем использовать их через переменные, например:

```
<!--# if expr="$name = /(.)@(?P<domain>.)/" -->  
    <!--# echo var="1" -->  
    <!--# echo var="domain" -->  
<!--# endif -->
```

include

Включает в ответ результат другого запроса. Параметры команды:

file

задает включаемый файл, например:

```
<!--# include file="footer.html" -->
```

virtual

задает включаемый запрос, например:

```
<!--# include virtual="/remote/body.php?argument=value" -->
```

Несколько запросов, указанных на одной странице и обрабатываемых проксируемыми или FastCGI/uwsgi/SCGI/gRPC-серверами, работают параллельно. Если нужна последовательная обработка, следует воспользоваться параметром *wait*.

stub

нестандартный параметр, задающий имя блока, содержимое которого будет выведено, если тело ответа на включаемый запрос пустое или если при исполнении запроса произошла ошибка, например:

```
<!--# block name="one" -->&nbsp;  <!--# endblock -->
<!--# include virtual="/remote/body.php?argument=value" stub="one" -->
```

Содержимое замещающего блока обрабатывается в контексте включаемого запроса.

wait

нестандартный параметр, указывающий, нужно ли ждать полного исполнения данного запроса, прежде чем продолжать выполнение SSI, например:

```
<!--# include virtual="/remote/body.php?argument=value" wait="yes" -->
```

set

нестандартный параметр, указывающий, что удачный результат выполнения запроса нужно записать в заданную переменную, например:

```
<!--# include virtual="/remote/body.php?argument=value" set="one" -->
```

Максимальный размер ответа задается директивой *subrequest_output_buffer_size*:

```
location /remote/ {
    subrequest_output_buffer_size 64k;
    # ...
}
```

`set`

Присваивает значение переменной. Параметры команды:

`var`

имя переменной.

`value`

значение переменной. Если в присваиваемом значении есть переменные, то производится подстановка их значений.

Встроенные переменные

`$date_local`

текущее время в локальной временной зоне. Формат задается командой `config` с параметром `timefmt`.

`$date_gmt`

текущее время в GMT. Формат задается командой `config` с параметром `timefmt`.

SSL

Обеспечивает работу по протоколу HTTPS.

При сборке из исходного кода модуль не собирается по умолчанию; его необходимо включить с помощью параметра сборки `--with-http_ssl_module`. В пакетах и образах из наших репозиториев модуль включен в сборку.

Примечание

Для этого модуля нужна библиотека OpenSSL.

Пример конфигурации

Для уменьшения загрузки процессора рекомендуется

- установить число *рабочих процессов* равным числу процессоров,
- разрешить *keep-alive* соединения,
- включить *разделяемый кэш* сессий,
- выключить *встроенный кэш* сессий
- и, возможно, увеличить *время жизни* сессии (по умолчанию 5 минут):

```
worker_processes auto;

http {
```

```
# ...

server {
    listen          443 ssl;
    keepalive_timeout 70;

    ssl_protocols    TLSv1.2 TLSv1.3;
    ssl_ciphers       AES128-SHA:AES256-SHA:RC4-SHA:DES-CBC3-SHA:RC4-MD5;
    ssl_certificate   /usr/local/angie/conf/cert.pem;
    ssl_certificate_key /usr/local/angie/conf/cert.key;
    ssl_session_cache shared:SSL:10m;
    ssl_session_timeout 10m;

    # ...
}
```

Директивы

ssl_buffer_size

<i>Синтаксис</i>	ssl_buffer_size <i>размер</i> ;
По умолчанию	ssl_buffer_size 16k;
<i>Контекст</i>	http, server

Задаёт размер буфера, используемого при отправке данных.

По умолчанию размер буфера равен 16k, что соответствует минимальным накладным расходам при передаче больших ответов. С целью минимизации времени получения начала ответа (Time To First Byte) может быть полезно использовать меньшие значения, например:

```
ssl_buffer_size 4k;
```

ssl_certificate

<i>Синтаксис</i>	ssl_certificate <i>файл</i> ;
По умолчанию	—
<i>Контекст</i>	http, server

Указывает файл с сертификатом в формате PEM для данного виртуального сервера. Если вместе с основным сертификатом нужно указать промежуточные, то они должны находиться в этом же файле в следующем порядке: сначала основной сертификат, а затем промежуточные. В этом же файле может находиться секретный ключ в формате PEM.

Эта директива может быть указана несколько раз для загрузки сертификатов разных типов, например RSA и ECDSA:

```
server {
    listen          443 ssl;
    server_name      example.com;

    ssl_certificate   example.com.rsa.crt;
```

```
ssl_certificate_key example.com.rsa.key;

ssl_certificate      example.com.ecdsa.crt;
ssl_certificate_key  example.com.ecdsa.key;

# ...
}
```

Возможность задавать отдельные цепочки сертификатов для разных сертификатов есть только в OpenSSL 1.0.2 и выше. Для более старых версий следует указывать только одну цепочку сертификатов.

i Примечание

В имени файла можно использовать переменные при использовании OpenSSL 1.0.2 и выше:

```
ssl_certificate      $ssl_server_name.crt;
ssl_certificate_key  $ssl_server_name.key;
```

При использовании переменных сертификат загружается при каждой операции SSL-рукопожатия, что может отрицательно влиять на производительность.

Вместо *файла* можно указать значение **data:\$переменная**, при котором сертификат загружается из переменной без использования промежуточных файлов.

Ненадлежащее использование подобного синтаксиса может быть небезопасно, например данные секретного ключа могут попасть в *лог ошибок*.

i Примечание

Нужно иметь в виду, что из-за ограничения протокола HTTPS для максимальной совместимости виртуальные серверы должны слушать на разных IP-адресах.

Если задан режим *ssl_ntls*, директива может принимать два аргумента (части ключа для подписи и шифрования) вместо одного:

```
listen ... ssl;

ssl_ntls on;

# двойной сертификат NTLS
ssl_certificate      sign.crt enc.crt;
ssl_certificate_key  sign.key enc.key;

# можно использовать наряду с обычным сертификатом RSA
ssl_certificate      rsa.crt;
ssl_certificate_key  rsa.key;
```

ssl_certificate_cache

<i>Синтаксис</i>	<code>ssl_certificate_cache off;</code> <code>ssl_certificate_cache max=<i>N</i> [inactive=<i>time</i>] [valid=<i>time</i>];</code>
Значение по умолчанию	<code>ssl_certificate_cache off;</code>
<i>Контекст</i>	http, server

Определяет кэш для хранения *SSL-сертификатов* и *секретных ключей*, заданных через переменные.

Директива поддерживает следующие параметры:

- **max** — устанавливает максимальное количество элементов в кэше. При переполнении кэша удаляются наименее недавно использованные (LRU) элементы.
- **inactive** — определяет время, после которого элемент будет удален, если к нему не было обращений. Значение по умолчанию — 10 секунд.
- **valid** — определяет время, в течение которого элемент кэша считается действительным и может использоваться повторно. Значение по умолчанию — 60 секунд. По истечении этого времени сертификаты перезагружаются или проходят повторную проверку.
- **off** — отключает кэш.

Пример:

```
ssl_certificate      $ssl_server_name.crt;
ssl_certificate_key  $ssl_server_name.key;
ssl_certificate_cache max=1000 inactive=20s valid=1m;
```

ssl_certificate_key

<i>Синтаксис</i>	<code>ssl_certificate_key файл;</code>
По умолчанию	—
<i>Контекст</i>	http, server

Указывает файл с секретным ключом в формате PEM для данного виртуального сервера.

Примечание

В имени файла можно использовать переменные при использовании OpenSSL 1.0.2 и выше.

Вместо **файла** можно указать значение **engine:имя:id**, которое загружает ключ с указанным *id* из движка OpenSSL с заданным именем.

Вместо **файла** также можно указать значение **data:\$переменная**, при котором секретный ключ загружается из переменной без использования промежуточных файлов. При этом следует учитывать, что ненадлежащее использование подобного синтаксиса может быть небезопасно, например данные секретного ключа могут попасть в *лог ошибок*.

Если задан режим *ssl_nTLS*, директива может принимать два аргумента (части ключа для подписи и шифрования) вместо одного:

```
listen ... ssl;

ssl_ntls on;

# двойной сертификат NTLS
ssl_certificate      sign.crt enc.crt;
ssl_certificate_key  sign.key enc.key;

# можно использовать наряду с обычным сертификатом RSA
ssl_certificate      rsa.crt;
ssl_certificate_key  rsa.key;
```

ssl_ciphers

Синтаксис	ssl_ciphers <i>шифры</i> ;
По умолчанию	ssl_ciphers HIGH:!aNULL:!MD5;
Контекст	http, server

Описывает разрешенные шифры. Шифры задаются в формате, поддерживаемом библиотекой OpenSSL, например:

```
ssl_ciphers ALL:!aNULL:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP;
```

Список шифров зависит от установленной версии OpenSSL. Полный список можно посмотреть с помощью команды `openssl ciphers`.

⚠ Предупреждение

Директива `ssl_ciphers` не настраивает шифры для TLS 1.3 при использовании OpenSSL. Для настройки шифров TLS 1.3 в OpenSSL используйте директиву `ssl_conf_command`, добавленную для расширенной конфигурации SSL.

- В LibreSSL шифры TLS 1.3 *можно* настраивать с помощью `ssl_ciphers`.
- В BoringSSL шифры TLS 1.3 настроить невозможно.

ssl_client_certificate

Синтаксис	ssl_client_certificate <i>файл</i> ;
По умолчанию	—
Контекст	http, server

Указывает файл с доверенными сертификатами CA в формате PEM, которые используются для *проверки* клиентских сертификатов и ответов OCSP, если включен `ssl_stapling`.

Список сертификатов будет отправляться клиентам. Если это нежелательно, можно воспользоваться директивой `ssl_trusted_certificate`.

ssl_conf_command

<i>Синтаксис</i>	<code>ssl_conf_command имя значение;</code>
По умолчанию	—
<i>Контекст</i>	http, server

Задаёт произвольные конфигурационные команды OpenSSL.

Примечание

Директива поддерживается при использовании OpenSSL 1.0.2 и выше. Чтобы настроить шифры TLS 1.3 в OpenSSL, используйте команду `ciphersuites`.

На одном уровне может быть указано несколько директив `ssl_conf_command`:

```
ssl_conf_command Options PrioritizeChaCha;
ssl_conf_command Ciphersuites TLS_CHACHA20_POLY1305_SHA256;
```

Директивы наследуются с предыдущего уровня конфигурации при условии, что на данном уровне не описаны свои директивы `ssl_conf_command`.

Предупреждение

Изменение настроек OpenSSL напрямую может привести к неожиданному поведению.

ssl_crl

<i>Синтаксис</i>	<code>ssl_crl файл;</code>
По умолчанию	—
<i>Контекст</i>	http, server

Указывает файл с отозванными сертификатами (CRL) в формате PEM, используемыми для проверки клиентских сертификатов.

ssl_dhparam

<i>Синтаксис</i>	<code>ssl_dhparam файл;</code>
По умолчанию	—
<i>Контекст</i>	http, server

Указывает файл с параметрами для DHE-шифров.

Предупреждение

По умолчанию параметры не заданы, и соответственно DHE-шифры не будут использоваться.

ssl_early_data

<i>Синтаксис</i>	<code>ssl_early_data on off;</code>
По умолчанию	<code>ssl_early_data off;</code>
<i>Контекст</i>	http, server

Разрешает или запрещает TLS 1.3 early data.

Запросы, отправленные внутри early data, могут быть подвержены атакам повторного воспроизведения (replay). Для защиты от подобных атак на уровне приложения необходимо использовать переменную `$ssl_early_data`.

```
proxy_set_header Early-Data $ssl_early_data;
```

Примечание

Директива поддерживается при использовании OpenSSL 1.1.1 и выше или BoringSSL.

ssl_ecdh_curve

<i>Синтаксис</i>	<code>ssl_ecdh_curve кривая;</code>
По умолчанию	<code>ssl_ecdh_curve auto;</code>
<i>Контекст</i>	http, server

Задаёт кривую для ECDHE-шифров.

Примечание

При использовании OpenSSL 1.0.2 и выше можно указывать несколько кривых, например:

```
ssl_ecdh_curve prime256v1:secp384r1;
```

Специальное значение `auto` соответствует встроенному в библиотеку OpenSSL списку кривых для OpenSSL 1.0.2 и выше, или `prime256v1` для более старых версий.

Примечание

При использовании OpenSSL 1.0.2 и выше директива задает список кривых, поддерживаемых сервером. Поэтому для работы ECDSA-сертификатов важно, чтобы список включал кривые, используемые в сертификатах.

ssl_ntls

<i>Синтаксис</i>	ssl_ntls on off;
По умолчанию	ssl_ntls off;
<i>Контекст</i>	http, server

Включает серверную поддержку NTLS при использовании TLS библиотеки [TongSuo](#)

```
listen ... ssl;
ssl_ntls on;
```

Примечание

Angie необходимо собрать с использованием параметра конфигурации `--with-ntls`, с соответствующей SSL библиотекой с поддержкой NTLS

```
./configure --with-openssl=../Tongsuo-8.3.0 \
            --with-openssl-opt=enable-ntls \
            --with-ntls
```

ssl_ocsp

<i>Синтаксис</i>	ssl_ocsp on off leaf;
По умолчанию	ssl_ocsp off;
<i>Контекст</i>	http, server

Включает проверку OCSP для цепочки клиентских сертификатов. Параметр `leaf` включает проверку только клиентского сертификата.

Для работы проверки OCSP необходимо дополнительно установить значение директивы `ssl_verify_client` в `on` или `optional`.

Для преобразования имени хоста OCSP-респондера в адрес необходимо дополнительно задать директиву `resolver`.

Пример:

```
ssl_verify_client on;
ssl_ocsp          on;
resolver          127.0.0.53;
```

ssl_ocsp_cache

<i>Синтаксис</i>	<code>ssl_ocsp_cache off [shared:имя:размер];</code>
По умолчанию	<code>ssl_ocsp_cache off;</code>
<i>Контекст</i>	<code>http, server</code>

Задаёт имя и размер кэша, который хранит статус клиентских сертификатов для проверки OCSP-ответов. Кэш разделяется между всеми рабочими процессами. Кэш с одинаковым названием может использоваться в нескольких виртуальных серверах.

Параметр `off` запрещает использование кэша.

ssl_ocsp_responder

<i>Синтаксис</i>	<code>ssl_ocsp_responder uri;</code>
По умолчанию	<code>—</code>
<i>Контекст</i>	<code>http, server</code>

Переопределяет URI OCSP-респондера, указанный в расширении сертификата "Authority Information Access" для *проверки* клиентских сертификатов.

Поддерживаются только OCSP-респондеры на основе `http://`:

```
ssl_ocsp_responder http://ocsp.example.com/;
```

ssl_password_file

<i>Синтаксис</i>	<code>ssl_password_file файл;</code>
По умолчанию	<code>—</code>
<i>Контекст</i>	<code>http, server</code>

Задаёт файл с паролями от *секретных ключей*, где каждый пароль указан на отдельной строке. Пароли применяются по очереди в момент загрузки ключа.

Пример:

```
http {
    ssl_password_file /etc/keys/global.pass;
    ...

    server {
        server_name www1.example.com;
        ssl_certificate_key /etc/keys/first.key;
    }

    server {
        server_name www2.example.com;

        # вместо файла можно указать именованный канал
        ssl_password_file /etc/keys/fifo;
    }
}
```

```
    ssl_certificate_key /etc/keys/second.key;
}
}
```

ssl_prefer_server_ciphers

<i>Синтаксис</i>	ssl_prefer_server_ciphers on off;
По умолчанию	ssl_prefer_server_ciphers off;
<i>Контекст</i>	http, server

При использовании протоколов SSLv3 и TLS устанавливает приоритет серверных шифров над клиентскими.

ssl_protocols

<i>Синтаксис</i>	ssl_protocols [SSLv2] [SSLv3] [TLSv1] [TLSv1.1] [TLSv1.2] [TLSv1.3];
По умолчанию	ssl_protocols TLSv1.2 TLSv1.3;
<i>Контекст</i>	http, server

Изменено в версии 1.2.0: Параметр TLSv1.3 добавлен к используемым по умолчанию.

Разрешает указанные протоколы.

i Примечание

Параметры TLSv1.1 и TLSv1.2 работают только при использовании OpenSSL 1.0.1 и выше.

Параметр TLSv1.3 работает только при использовании OpenSSL 1.1.1 и выше.

ssl_reject_handshake

<i>Синтаксис</i>	ssl_reject_handshake on off;
По умолчанию	ssl_reject_handshake off;
<i>Контекст</i>	http, server

Если включено, то операции SSL-рукопожатия в блоке *server* будут отклонены.

Например, в этой конфигурации отклоняются все операции SSL-рукопожатия с именем сервера, отличным от *example.com*:

```
server {
    listen          443 ssl default_server;
    ssl_reject_handshake on;
}

server {
    listen          443 ssl;
```

```
server_name      example.com;
ssl_certificate   example.com.crt;
ssl_certificate_key example.com.key;
}
```

ssl_session_cache

<i>Синтаксис</i>	<code>ssl_session_cache off none [builtin[:размер]] [shared:название:размер];</code>
По умолчанию	<code>ssl_session_cache none;</code>
<i>Контекст</i>	http, server

Задаёт тип и размеры кэшей для хранения параметров сессий. Тип кэша может быть следующим:

off	жёсткое запрещение использования кэша сессий: Angie явно сообщает клиенту, что сессии не могут использоваться повторно.
none	мягкое запрещение использования кэша сессий: Angie сообщает клиенту, что сессии могут использоваться повторно, но на самом деле не хранит параметры сессии в кэше.
builtin	встроенный в OpenSSL кэш, используется в рамках только одного рабочего процесса. Размер кэша задается в сессиях. Если размер не задан, то он равен 20480 сессиям. Использование встроенного кэша может вести к фрагментации памяти.
shared	кэш, разделяемый между всеми рабочими процессами. Размер кэша задается в байтах, в 1 мегабайт может поместиться около 4000 сессий. У каждого разделяемого кэша должно быть произвольное название. Кэш с одинаковым названием может использоваться в нескольких виртуальных серверах. Также он используется для автоматического создания, хранения и периодического обновления ключей сессионных билетов TLS, если они не указаны явно с помощью директивы <code>ssl_session_ticket_key</code> .

Можно использовать одновременно оба типа кэша, например:

```
ssl_session_cache builtin:1000 shared:SSL:10m;
```

однако использование только разделяемого кэша без встроенного должно быть более эффективным.

ssl_session_ticket_key

<i>Синтаксис</i>	<code>ssl_session_ticket_key файл;</code>
По умолчанию	—
<i>Контекст</i>	http, server

Задаёт файл с секретным ключом, применяемым при шифровании и расшифровке сессионных билетов TLS. Директива необходима, если один и тот же ключ нужно использовать на нескольких серверах. По умолчанию используется случайно сгенерированный ключ.

Если указано несколько ключей, то только первый ключ используется для шифрования сессионных билетов TLS. Это позволяет настроить ротацию ключей, например:

```
ssl_session_ticket_key current.key;
ssl_session_ticket_key previous.key;
```

Файл должен содержать 80 или 48 байт случайных данных и может быть создан следующей командой:

```
openssl rand 80 > ticket.key
```

В зависимости от размера файла для шифрования будет использоваться либо AES256 (для 80-байтных ключей), либо AES128 (для 48-байтных ключей).

ssl_session_tickets

<i>Синтаксис</i>	ssl_session_tickets on off;
По умолчанию	ssl_session_tickets on;
<i>Контекст</i>	http, server

Разрешает или запрещает возобновление сессий при помощи [сессионных билетов TLS](#).

ssl_session_timeout

<i>Синтаксис</i>	ssl_session_timeout время;
По умолчанию	ssl_session_timeout 5m;
<i>Контекст</i>	http, server

Задаёт время, в течение которого клиент может повторно использовать параметры сессии.

ssl_stapling

<i>Синтаксис</i>	ssl_stapling on off;
По умолчанию	ssl_stapling off;
<i>Контекст</i>	http, server

Разрешает или запрещает [прикрепление OCSP-ответов](#) сервером. Пример:

```
ssl_stapling on;
resolver 127.0.0.53;
```

Для работы OCSP-прикрепления должен быть известен сертификат издателя сертификата сервера. Если в заданном директивой [ssl_certificate](#) файле не содержится промежуточных сертификатов, то сертификат издателя сертификата сервера следует поместить в файл, заданный директивой [ssl_trusted_certificate](#).



Предупреждение

Для преобразования имени хоста OCSP-респондера в адрес необходимо дополнительно задать директиву [resolver](#).

ssl_stapling_file

<i>Синтаксис</i>	<code>ssl_stapling_file файл;</code>
По умолчанию	—
<i>Контекст</i>	http, server

Если значение задано, то вместо опроса OCSP-респондера, указанного в сертификате сервера, ответ берется из указанного файла.

Ответ должен быть в формате DER и может быть сгенерирован командой `openssl ocsp`.

ssl_stapling_responder

<i>Синтаксис</i>	<code>ssl_stapling_responder uri;</code>
По умолчанию	—
<i>Контекст</i>	http, server

Переопределяет URI OCSP-респондера, указанный в расширении сертификата "Authority Information Access".

Поддерживаются только OCSP-респондеры на основе `http://`:

```
ssl_stapling_responder http://ocsp.example.com/;
```

ssl_stapling_verify

<i>Синтаксис</i>	<code>ssl_stapling_verify on off;</code>
По умолчанию	<code>ssl_stapling_verify off;</code>
<i>Контекст</i>	http, server

Разрешает или запрещает проверку ответов OCSP сервером.

Для работоспособности проверки сертификат издателя сертификата сервера, корневой сертификат и все промежуточные сертификаты должны быть указаны как доверенные с помощью директивы `ssl_trusted_certificate`.

ssl_trusted_certificate

<i>Синтаксис</i>	<code>ssl_trusted_certificate файл;</code>
По умолчанию	—
<i>Контекст</i>	http, server

Задаёт файл с доверенными сертификатами CA в формате PEM, которые используются для *про-верки* клиентских сертификатов и ответов OCSP, если включен `ssl_stapling`.

В отличие от `ssl_client_certificate`, список этих сертификатов не будет отправляться клиентам.

ssl_verify_client

<i>Синтаксис</i>	<code>ssl_verify_client on off optional optional_no_ca;</code>
По умолчанию	<code>ssl_verify_client off;</code>
<i>Контекст</i>	http, server

Разрешает проверку клиентских сертификатов. Результат проверки доступен через переменную `$ssl_client_verify`.

<code>optional</code>	запрашивает клиентский сертификат, и если сертификат был предоставлен, проверяет его
<code>optional_no_ca</code>	запрашивает сертификат клиента, но не требует, чтобы он был подписан доверенным сертификатом СА. Это предназначено для случаев, когда фактическая проверка сертификата осуществляется внешним по отношению к Angie сервисом.

ssl_verify_depth

<i>Синтаксис</i>	<code>ssl_verify_depth число;</code>
По умолчанию	<code>ssl_verify_depth 1;</code>
<i>Контекст</i>	http, server

Устанавливает глубину проверки в цепочке клиентских сертификатов.

Обработка ошибок

Модуль `http_ssl` поддерживает несколько нестандартных кодов ошибок, которые можно использовать для перенаправления с помощью директивы `error_page`:

495	при проверке клиентского сертификата произошла ошибка;
496	клиент не предоставил требуемый сертификат;
497	обычный запрос был послан на порт HTTPS.

Перенаправление делается после того, как запрос полностью разобран и доступны такие переменные, как `$request_uri`, `$uri`, `$args` и другие переменные.

Встроенные переменные

Модуль `http_ssl` поддерживает встроенные переменные:

`$ssl_alpn_protocol`

возвращает протокол, выбранный при помощи ALPN во время SSL-рукопожатия, либо пустую строку.

`$ssl_cipher`

возвращает название используемого шифра для установленного SSL-соединения.

`$ssl_ciphers`

возвращает список шифров, поддерживаемых клиентом. Известные шифры указаны по имени, неизвестные указаны в шестнадцатеричном виде, например:

AES128-SHA:AES256-SHA:0x00ff

Примечание

Переменная полностью поддерживается при использовании OpenSSL версии 1.0.2 и выше. При использовании более старых версий переменная доступна только для новых сессий и может содержать только известные шифры.

`$ssl_client_escaped_cert`

возвращает клиентский сертификат в формате PEM (закодирован в формате urlencode) для установленного SSL-соединения.

`$ssl_client_fingerprint`

возвращает SHA1-отпечаток клиентского сертификата для установленного SSL-соединения.

`$ssl_client_i_dn`

возвращает строку "issuer DN" клиентского сертификата для установленного SSL-соединения согласно RFC 2253.

`$ssl_client_i_dn_legacy`

возвращает строку "issuer DN" клиентского сертификата для установленного SSL-соединения.

`$ssl_client_raw_cert`

возвращает клиентский сертификат для установленного SSL-соединения в формате PEM.

`$ssl_client_s_dn`

возвращает строку "subject DN" клиентского сертификата для установленного SSL-соединения согласно RFC 2253.

`$ssl_client_s_dn_legacy`

возвращает строку "subject DN" клиентского сертификата для установленного SSL-соединения.

`$ssl_client_serial`

возвращает серийный номер клиентского сертификата для установленного SSL-соединения.

`$ssl_client_v_end`

возвращает дату окончания срока действия клиентского сертификата.

`$ssl_client_v_remain`

возвращает число дней, оставшихся до истечения срока действия клиентского сертификата.

`$ssl_client_v_start`

возвращает дату начала срока действия клиентского сертификата.

`$ssl_client_verify`

возвращает результат проверки клиентского сертификата: SUCCESS, FAILED:reason и, если сертификат не был предоставлен, NONE.

`$ssl_curve`

возвращает согласованную кривую, использованную для обмена ключами во время SSL-рукопожатия. Известные кривые указаны по имени, неизвестные указаны в шестнадцатеричном виде, например:

prime256v1

Примечание

Переменная поддерживается при использовании OpenSSL версии 3.0 и выше. При использовании более старых версий значением переменной будет пустая строка.

`$ssl_curves`

возвращает список кривых, поддерживаемых клиентом. Известные кривые указаны по имени, неизвестные указаны в шестнадцатеричном виде, например:

`0x001d:prime256v1:secp521r1:secp384r1`

Примечание

Переменная поддерживается при использовании OpenSSL версии 1.0.2 и выше. При использовании более старых версий значением переменной будет пустая строка.

Переменная доступна только для новых сессий.

`$ssl_early_data`

возвращает 1, если используется TLS 1.3 *early data* и операция SSL-рукопожатия не завершена, иначе "".

`$ssl_protocol`

возвращает протокол установленного SSL-соединения.

`$ssl_server_cert_type`

принимает значения RSA, DSA, ECDSA, ED448, ED25519, SM2, RSA-PSS или **unknown** в зависимости от типа сертификата и ключа сервера.

`$ssl_server_name`

возвращает имя сервера, запрошенное через [SNI](#).

`$ssl_session_id`

возвращает идентификатор сессии установленного SSL-соединения.

`$ssl_session_reused`

возвращает **r**, если сессия была использована повторно, иначе ".".

Stub Status

Предоставляет доступ к базовой информации о состоянии сервера.

При сборке из исходного кода модуль не собирается по умолчанию; его необходимо включить с помощью параметра сборки `--with-http_stub_status_module`. В пакетах и образах из наших репозиториях модуль включен в сборку.

Пример конфигурации

```
location = /basic_status {
    stub_status;
}
```

В данной конфигурации создается простая веб-страница с основной информацией о состоянии, которая может выглядеть следующим образом:

```
Active connections: 291
server accepts handled requests
16630948 16630948 31070465
Reading: 6 Writing: 179 Waiting: 106
```

Директивы

stub_status

<i>Синтаксис</i>	stub_status;
По умолчанию	—
<i>Контекст</i>	server, location

Информация о состоянии будет доступна из данного location.

Данные

Доступна следующая информация:

Active connections

Текущее число активных клиентских соединений, включая Waiting-соединения.

accepts

Суммарное число принятых клиентских соединений.

handled

Суммарное число обработанных соединений. Обычно значение этого параметра совпадает с `accepts`, если не достигнуто какое-нибудь системное ограничение (например, лимит `worker_connections`).

requests

Суммарное число клиентских запросов.

Reading

Текущее число соединений, в которых Angie в настоящий момент читает заголовок запроса.

Writing

Текущее число соединений, в которых Angie в настоящий момент отвечает клиенту.

Waiting

Текущее число бездействующих клиентских соединений в ожидании запроса.

Встроенные переменные

`$connections_active`

то же, что и значение *Active connections*;

`$connections_reading`

то же, что и значение *Reading*;

`$connections_writing`

то же, что и значение *Writing*;

`$connections_waiting`

то же, что и значение *Waiting*.

Sub

Фильтр, изменяющий в ответе одну заданную строку на другую.

При сборке из исходного кода модуль не собирается по умолчанию; его необходимо включить с помощью параметра сборки `--with-http_sub_module`. В пакетах и образах из наших репозиториев модуль включен в сборку.

Пример конфигурации

```
location / {
    sub_filter '<a href="http://127.0.0.1:8080/' '<a href="https://$host/';
    sub_filter 'Синтаксис</i> | <code>sub_filter строка замена;</code> |
| По умолчанию     | —                                      |
| <i>Контекст</i>  | http, server, location                 |

Задаёт строку, которую нужно заменить, и строку замены. Заменяемая строка проверяется без учёта регистра. В заменяемой строке и в строке замены можно использовать переменные. На одном уровне конфигурации может быть указано несколько директив `sub_filter`. Директивы наследуются с предыдущего уровня конфигурации при условии, что на данном уровне не описаны свои директивы `sub_filter`.

### sub\_filter\_last\_modified

|                  |                                                 |
|------------------|-------------------------------------------------|
| <i>Синтаксис</i> | <code>sub_filter_last_modified on   off;</code> |
| По умолчанию     | <code>sub_filter_last_modified off;</code>      |
| <i>Контекст</i>  | http, server, location                          |

Позволяет сохранить поле заголовка **Last-Modified** исходного ответа во время замены для лучшего кэширования ответов.

По умолчанию поле заголовка удаляется, так как содержимое ответа изменяется во время обработки.

### sub\_filter\_once

|                  |                                        |
|------------------|----------------------------------------|
| <i>Синтаксис</i> | <code>sub_filter_once on   off;</code> |
| По умолчанию     | <code>sub_filter_once on;</code>       |
| <i>Контекст</i>  | http, server, location                 |

Определяет, сколько раз нужно искать каждую из заменяемых строк: один раз или многократно.

## sub\_filter\_types

|                  |                                             |
|------------------|---------------------------------------------|
| <i>Синтаксис</i> | <code>sub_filter_types mime-mun ...;</code> |
| По умолчанию     | <code>sub_filter_types text/html;</code>    |
| <i>Контекст</i>  | <code>http, server, location</code>         |

Разрешает замену строк в ответах с указанными MIME-типами в дополнение к `text/html`. Специальное значение "\*" соответствует любому MIME-типу.

## Upstream

Предоставляет контекст для описания группы серверов, которые могут использоваться в директивах `proxy_pass`, `fastcgi_pass`, `uwsgi_pass`, `scgi_pass`, `memcached_pass` и `grpc_pass`.

## Пример конфигурации

```
upstream backend {
 zone backend 1m;
 server backend1.example.com weight=5;
 server backend2.example.com:8080;
 server backend3.example.com service=_example._tcp resolve;
 server unix:/tmp/backend3;

 server backup1.example.com:8080 backup;
 server backup2.example.com:8080 backup;
}

resolver 127.0.0.53 status_zone=resolver;

server {
 location / {
 proxy_pass http://backend;
 }
}
```

## Директивы

### backup\_switch

|                  |                                                      |
|------------------|------------------------------------------------------|
| <i>Синтаксис</i> | <code>backup_switch permanent[=<i>время</i>];</code> |
| По умолчанию     | —                                                    |
| <i>Контекст</i>  | <code>upstream</code>                                |

Директива включает возможность начать выбор серверов не с основной группы, а с *активной*, то есть той, где в предыдущий раз был успешно найден сервер. Если найти сервер в активной группе для очередного запроса не удастся, и поиск переходит к резервной группе, то уже эта группа становится активной, и последующие запросы сначала направляются на серверы этой группы.



Если параметр **permanent** определен без значения *времени*, группа остается активной после выбора, и автоматическая перепроверка групп с меньшим уровнем не происходит. Если *время* задано, то активный статус группы истекает через указанный интервал, и балансировщик снова проверяет группы с меньшим уровнем, возвращаясь к ним, если серверы работают нормально.

Пример:

```
upstream my_backend {
 server primary1.example.com;
 server primary2.example.com;

 server backup1.example.com backup;
 server backup2.example.com backup;

 backup_switch permanent=2m;
}
```

Если балансировщик переключается с основных серверов на резервную группу, все последующие запросы обрабатываются этой резервной группой в течение 2 минут. По истечении 2 минут балансировщик повторно проверяет основные серверы и снова делает их активными, если они работают нормально.

## bind\_conn

|              |                            |
|--------------|----------------------------|
| Синтаксис    | <b>bind_conn</b> значение; |
| По умолчанию | —                          |
| Контекст     | upstream                   |

Позволяет привязать серверное соединение к клиентскому в момент, когда *значение*, заданное строкой из переменных, становится отличным от "" и "0".

### ⚠ Предупреждение

Директива **bind\_conn** должна использоваться после всех директив, задающих тот или иной метод балансировки нагрузки, иначе она не будет работать. Если она используется наряду с директивой *sticky*, то **bind\_conn** должна стоять после *sticky*.

### ⚠ Предупреждение

При использовании директивы настройки модуля *Proxy* должны допускать использование постоянных соединений, например:

```
proxy_http_version 1.1;
proxy_set_header Connection "";
```

Типичный пример использования директивы — проксирование соединений с NTLM-аутентификацией, где требуется обеспечить привязку клиента к серверу в начале согласования:

```
map $http_authorization $ntlm {
 ~*~N(?:TLM|egotiate) 1;
}

upstream ntlm_backend {
```

```
server 127.0.0.1:8080;
bind_conn $ntlm;
}

server {
 # ...
 location / {
 proxy_pass http://ntlm_backend;
 proxy_http_version 1.1;
 proxy_set_header Connection "";
 # ...
 }
}
```

## feedback

|                  |                                                                                                      |
|------------------|------------------------------------------------------------------------------------------------------|
| <i>Синтаксис</i> | <code>feedback переменная [inverse] [factor=число] [account=условная_переменная] [last_byte];</code> |
| По умолчанию     | —                                                                                                    |
| <i>Контекст</i>  | upstream                                                                                             |

Задаёт в `upstream` механизм балансировки нагрузки по обратной связи. Он динамически корректирует решения при балансировке, умножая вес каждого проксируемого сервера на среднее значение обратной связи, которое меняется с течением времени в зависимости от значения *переменной* и подчиняется необязательному условию.

Могут быть заданы следующие параметры:

|                                                                                                                                                                                                                                                                                                                                            |                                                                                                                                                                                                                                                                                                                                                                                    |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>переменная</b>                                                                                                                                                                                                                                                                                                                          | Переменная, из которой берётся значение обратной связи. Она должна представлять собой метрику производительности или состояния; предполагается, что сервер передаёт её в заголовках или иным образом. Значение оценивается при каждом ответе от сервера и учитывается в скользящем среднем согласно настройкам <b>inverse</b> и <b>factor</b> .                                    |
| <b>inverse</b>                                                                                                                                                                                                                                                                                                                             | Если параметр задан, значение обратной связи интерпретируется наоборот: более низкие значения указывают на лучшую производительность.                                                                                                                                                                                                                                              |
| <b>factor</b>                                                                                                                                                                                                                                                                                                                              | Коэффициент, по которому значение обратной связи учитывается при расчёте среднего. Допустимы целые числа от 0 до 99. По умолчанию — 90. Среднее рассчитывается по формуле <a href="#">экспоненциального сглаживания</a> . Чем больше коэффициент, тем меньше новые значения влияют на среднее; если указать 90, то будет взято 90 % от предыдущего значения и лишь 10 % от нового. |
| <b>account</b>                                                                                                                                                                                                                                                                                                                             | Указывает условную переменную, которая контролирует, какие ответы учитываются при расчёте. Среднее значение обновляется с учётом значения обратной связи из ответа, только если условная переменная этого ответа не равна "" или "0".                                                                                                                                              |
| <div>  <b>Примечание</b> </div> <p>По умолчанию ответы в ходе <i>активных проверок</i> не включаются в расчёт; комбинация переменной <code>\$upstream_probe</code> с <b>account</b> позволяет включить эти ответы или даже исключить все остальное.</p> |                                                                                                                                                                                                                                                                                                                                                                                    |
| <b>last_byte</b>                                                                                                                                                                                                                                                                                                                           | Позволяет обрабатывать данные от проксируемого сервера после получения полного ответа, а не только заголовка.                                                                                                                                                                                                                                                                      |

Пример:

```
upstream backend {
 zone backend 1m;

 feedback $feedback_value factor=80 account=$condition_value;

 server backend1.example.com;
 server backend2.example.com;
}

map $upstream_http_custom_score $feedback_value {
 "high" 100;
 "medium" 75;
 "low" 50;
 default 10;
}

map $upstream_probe $condition_value {
 "high_priority" "1";
 "low_priority" "0";
 default "1";
}
```

Эта конфигурация категоризирует ответы серверов по уровням обратной связи на основе определенных оценок из полей заголовков ответа, а также добавляет условие на `$upstream_probe`, чтобы учитывать только ответы от активной проверки `high_priority` или ответы на обычные клиентские запросы.

## hash

|                  |                                             |
|------------------|---------------------------------------------|
| <i>Синтаксис</i> | <code>hash <i>ключ</i> [consistent];</code> |
| По умолчанию     | —                                           |
| <i>Контекст</i>  | upstream                                    |

Задаёт метод балансировки нагрузки для группы, при котором соответствие клиента серверу определяется при помощи хэшированного значения ключа. В качестве ключа может использоваться текст, переменные и их комбинации. Следует отметить, что любое добавление или удаление серверов в группе может привести к перераспределению большинства ключей на другие серверы. Метод совместим с библиотекой Perl `Cache::Memcached`.

```
hash $remote_addr;
```

При использовании доменных имен, разрешающихся в несколько IP-адресов (например, с параметром `resolve`), сервер не сортирует полученные адреса, поэтому их порядок может различаться на разных серверах, что влияет на распределение клиентов. Чтобы обеспечить одинаковое распределение, используйте параметр `consistent`.

Если задан параметр `consistent`, то вместо вышеописанного метода будет использоваться метод консистентного хэширования `ketama`. Метод гарантирует, что при добавлении сервера в группу или его удалении на другие серверы будет перераспределено минимальное число ключей. Применение метода для кэширующих серверов обеспечивает больший процент попаданий в кэш. Метод совместим с библиотекой Perl `Cache::Memcached::Fast` при значении параметра `ketama_points`, равном 160.

## ip\_hash

|                  |                       |
|------------------|-----------------------|
| <i>Синтаксис</i> | <code>ip_hash;</code> |
| По умолчанию     | —                     |
| <i>Контекст</i>  | upstream              |

Задаёт для группы метод балансировки нагрузки, при котором запросы распределяются по серверам на основе IP-адресов клиентов. В качестве ключа для хэширования используются первые три октета IPv4-адреса клиента или IPv6-адрес клиента целиком. Метод гарантирует, что запросы одного и того же клиента будут всегда передаваться на один и тот же сервер. Если же этот сервер будет считаться недоступным, то запросы этого клиента будут передаваться на другой сервер. С большой долей вероятности это также будет один и тот же сервер.

Если один из серверов нужно убрать на некоторое время, то для сохранения текущего хэширования IP-адресов клиентов этот сервер нужно пометить параметром `down`:

```
upstream backend {
 ip_hash;

 server backend1.example.com;
 server backend2.example.com;
 server backend3.example.com down;
 server backend4.example.com;
}
```

## keepalive

|                  |                                    |
|------------------|------------------------------------|
| <i>Синтаксис</i> | <code>keepalive соединения;</code> |
| По умолчанию     | —                                  |
| <i>Контекст</i>  | upstream                           |

Задействует кэш соединений для группы серверов.

Параметр `соединения` устанавливает максимальное число неактивных постоянных соединений с серверами группы, которые будут сохраняться в кэше каждого рабочего процесса. При превышении этого числа наиболее давно не используемые соединения закрываются.

### Примечание

Следует особо отметить, что директива *keepalive* не ограничивает общее число соединений с серверами группы, которые рабочие процессы Angie могут открыть. Параметр соединения следует устанавливать достаточно консервативно, чтобы серверы группы по-прежнему могли обрабатывать новые входящие соединения.

### Предупреждение

Директива `keepalive` должна использоваться после всех директив, задающих тот или иной метод балансировки нагрузки, иначе она не будет работать.

Пример конфигурации группы серверов memcached с постоянными соединениями:

```
upstream memcached_backend {
 server 127.0.0.1:11211;
 server 10.0.0.2:11211;

 keepalive 32;
}

server {
 #...

 location /memcached/ {
 set $memcached_key $uri;
 memcached_pass memcached_backend;
 }
}
```

Для HTTP директиву *proxy\_http\_version* следует установить в "1.1", а поле заголовка **Connection** — очистить:

```
upstream http_backend {
 server 127.0.0.1:8080;

 keepalive 16;
}

server {
 #...

 location /http/ {
 proxy_pass http://http_backend;
 proxy_http_version 1.1;
 proxy_set_header Connection "";
 }
}
```

#### Примечание

Хоть это и не рекомендуется, но также возможно использование постоянных соединений с HTTP/1.0, путем передачи поля заголовка "Connection: Keep-Alive" серверу группы.

Для работы постоянных соединений с FastCGI-серверами потребуется включить директиву *fastcgi\_keep\_conn*:

```
upstream fastcgi_backend {
 server 127.0.0.1:9000;

 keepalive 8;
}

server {
 #...

 location /fastcgi/ {
 fastcgi_pass fastcgi_backend;
 }
}
```

```
 fastcgi_keep_conn on;
...
}
}
```

#### **Примечание**

Протоколы SCGI и uwsgi не определяют семантику постоянных соединений.

### **keepalive\_requests**

|                  |                                                |
|------------------|------------------------------------------------|
| <i>Синтаксис</i> | <code>keepalive_requests</code> <i>число</i> ; |
| По умолчанию     | <code>keepalive_requests 1000</code> ;         |
| <i>Контекст</i>  | upstream                                       |

Задаёт максимальное число запросов, которые можно сделать по одному постоянному соединению. После того как сделано максимальное число запросов, соединение закрывается.

Периодическое закрытие соединений необходимо для освобождения памяти, выделенной под конкретные соединения. Поэтому использование слишком большого максимального числа запросов может приводить к чрезмерному потреблению памяти и не рекомендуется.

### **keepalive\_time**

|                  |                                            |
|------------------|--------------------------------------------|
| <i>Синтаксис</i> | <code>keepalive_time</code> <i>время</i> ; |
| По умолчанию     | <code>keepalive_time 1h</code> ;           |
| <i>Контекст</i>  | upstream                                   |

Ограничивает максимальное время, в течение которого могут обрабатываться запросы в рамках постоянного соединения. По достижении заданного времени соединение закрывается после обработки очередного запроса.

### **keepalive\_timeout**

|                  |                                               |
|------------------|-----------------------------------------------|
| <i>Синтаксис</i> | <code>keepalive_timeout</code> <i>время</i> ; |
| По умолчанию     | <code>keepalive_timeout 60s</code> ;          |
| <i>Контекст</i>  | upstream                                      |

Задаёт таймаут, в течение которого неактивное постоянное соединение с сервером группы не будет закрыто.

## least\_bandwidth

|                  |                                                                            |
|------------------|----------------------------------------------------------------------------|
| <i>Синтаксис</i> | <code>least_bandwidth both   upstream   downstream [factor=number];</code> |
| По умолчанию     | <code>both</code>                                                          |
| <i>Контекст</i>  | <code>upstream</code>                                                      |

Модуль балансировки нагрузки `least_bandwidth` выбирает проксируемый сервер, который использует наименьший объем пропускной способности, стремясь равномерно распределить сетевой трафик, направляя новые сессии на сервер с наименьшим потреблением пропускной способности. Соответственно, чем меньше трафика, тем сервер считается менее загруженным и тем больше запросов будет на него отправляться.

|                         |                                                                                                                                                                                                |
|-------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>upstream</code>   | Директива учитывает только пропускную способность, используемую <b>от клиента к серверу</b> .                                                                                                  |
| <code>downstream</code> | Директива учитывает только пропускную способность, используемую <b>от сервера к клиенту</b> .                                                                                                  |
| <code>both</code>       | Директива учитывает сумму пропускной способности, используемой вверх (от клиента к серверу) и вниз по потоку (от сервера к клиенту). Режим по умолчанию.                                       |
| <code>factor</code>     | Необязательный коэффициент сглаживания для расчета скользящего среднего, в диапазоне от 0 до 100 (по умолчанию 90). Более высокий коэффициент означает более медленную адаптацию к изменениям. |

Модуль периодически вычисляет среднее использование пропускной способности для каждого проксируемого сервера, используя формулу скользящего среднего для сглаживания колебаний со временем. Когда начинается новая сессия, он выбирает проксируемый сервер с наименьшим средним использованием пропускной способности, скорректированным с учетом веса сервера, если он указан. Если несколько серверов имеют одинаковое минимальное среднее использование пропускной способности, применяется взвешенный метод round-robin.

## least\_conn

|                  |                          |
|------------------|--------------------------|
| <i>Синтаксис</i> | <code>least_conn;</code> |
| По умолчанию     | —                        |
| <i>Контекст</i>  | <code>upstream</code>    |

Задаёт для группы метод балансировки нагрузки, при котором запрос передается серверу с наименьшим числом активных соединений, с учетом весов серверов. Если подходит сразу несколько серверов, они выбираются циклически (в режиме round-robin) с учетом их весов.

## least\_time

|              |                                                                             |
|--------------|-----------------------------------------------------------------------------|
| Синтаксис    | least_time header   last_byte [factor=число] [account=условная_переменная]; |
| По умолчанию | —                                                                           |
| Контекст     | upstream                                                                    |

Задаёт для группы метод балансировки нагрузки, при котором вероятность передачи запроса активному серверу обратно пропорциональна среднему времени его ответа; чем оно меньше, тем больше запросов будет получать сервер.

|           |                                                                |
|-----------|----------------------------------------------------------------|
| header    | Директива учитывает среднее время получения заголовков ответа. |
| last_byte | Директива использует среднее время получения полного ответа.   |

|         |                                                                                                                                                                                      |
|---------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| factor  | Выполняет ту же функцию, что и <i>response_time_factor</i> , и переопределяет его, если параметр задан.                                                                              |
| account | Указывает условную переменную, которая контролирует, какие ответы учитываются при расчёте. Среднее значение обновляется, только если условная переменная ответа не равна "" или "0". |

### Примечание

По умолчанию ответы в ходе *активных проверок* не включаются в расчёт; комбинация переменной *\$upstream\_probe* с *account* позволяет включить эти ответы или даже исключить все остальное.

Текущие значения представлены как *header\_time* (только заголовки) и *response\_time* (ответы целиком) в объекте *health* сервера среди *метрик апстрима* в API.

## queue

|              |                              |
|--------------|------------------------------|
| Синтаксис    | queue число [timeout=время]; |
| По умолчанию | —                            |
| Контекст     | upstream                     |

Если для запроса не удастся назначить проксируемый сервер с первой попытки (например, при краткосрочном перебое в работе или всплеске нагрузки с достижением предела *max\_conns*), запрос не отклоняется; вместо этого Angie пытается поставить его в очередь на обработку.

Численный параметр директивы задаёт максимальное количество запросов в очереди *рабочего процесса*. Если очередь целиком заполнена, клиенту отдаётся ошибка 502 (Bad Gateway).

### Примечание

К запросам в очереди также применяется логика директивы *proxy\_next\_upstream*. В частности, если для запроса был выбран сервер, но передать его туда не удалось, то он может вернуться в очередь.

Если сервер для передачи запроса в очереди не был выбран за *время timeout* (по умолчанию — 60 секунд), клиенту отдаётся ошибка 502 (Bad Gateway). Ещё из очереди удаляются запросы



от клиентов, преждевременно закрывших соединение; в *API* есть счетчики состояний запросов, проходящих через очередь.

#### ⚠ Предупреждение

Директива `queue` должна использоваться после всех директив, задающих тот или иной метод балансировки нагрузки, иначе она не будет работать.

### random

|                  |                            |
|------------------|----------------------------|
| <i>Синтаксис</i> | <code>random [two];</code> |
| По умолчанию     | —                          |
| <i>Контекст</i>  | upstream                   |

Задаёт для группы метод балансировки нагрузки, при котором запрос передается случайно выбранному серверу, с учетом весов серверов.

Если указан необязательный параметр `two`, Angie случайным образом выбирает два сервера, из которых выбирает сервер, используя метод *least\_conn*, при котором запрос передается на сервер с наименьшим количеством активных соединений.

### response\_time\_factor

|                  |                                          |
|------------------|------------------------------------------|
| <i>Синтаксис</i> | <code>response_time_factor число;</code> |
| По умолчанию     | <code>response_time_factor 90;</code>    |
| <i>Контекст</i>  | upstream                                 |

Задаёт для метода балансировки нагрузки *least\_time* коэффициент сглаживания **предыдущего** значения при вычислении среднего времени ответа по формуле **экспоненциально взвешенного скользящего среднего**.

Чем больше указанное *число*, тем меньше новые значения влияют на среднее; если указать 90, то будет взято 90 % от предыдущего значения и лишь 10 % от нового. Допустимые значения — от 0 до 99 включительно.

Текущие результаты вычислений представлены как `header_time` (только заголовки) и `response_time` (ответы целиком) в объекте `health` сервера среди *метрик апстрима* в API.

#### ℹ Примечание

При подсчете учитываются только успешные ответы; что считать неуспешным ответом, определяют директивы *proxy\_next\_upstream*, *fastcgi\_next\_upstream*, *uwsgi\_next\_upstream*, *scgi\_next\_upstream*, *memcached\_next\_upstream* и *grpc\_next\_upstream*. Кроме того, значение `header_time` пересчитывается, только если получены и обработаны все заголовки, а `response_time` — только если получен весь ответ.

## server

|                  |                                                  |
|------------------|--------------------------------------------------|
| <i>Синтаксис</i> | <b>server</b> <i>адрес</i> [ <i>параметры</i> ]; |
| По умолчанию     | —                                                |
| <i>Контекст</i>  | upstream                                         |

Задаёт адрес и другие параметры сервера. Адрес может быть указан в виде доменного имени или IP-адреса, и необязательного порта, или в виде пути UNIX-сокета, который указывается после префикса `unix:`. Если порт не указан, используется порт 80. Доменное имя, которому соответствует несколько IP-адресов, задаёт сразу несколько серверов.

Могут быть заданы следующие параметры:

|                        |                                                                                                                                                                                                                                                                                       |
|------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>weight=число</b>    | Задаёт вес сервера. По умолчанию — 1.                                                                                                                                                                                                                                                 |
| <b>max_conns=число</b> | Ограничивает максимальное число одновременных активных соединений к проксируемому серверу. Значение по умолчанию равно 0 и означает, что ограничения нет. Если группа не находится в <i>зоне разделяемой памяти</i> , то ограничение работает отдельно для каждого рабочего процесса. |

### Примечание

При включённых *неактивных постоянных соединениях*, нескольких *рабочих процессах* и *зоне разделяемой памяти* суммарное число активных и неактивных соединений с проксируемым сервером может превышать значение `max_conns`.

**max\_fails=число** — задаёт число неудачных попыток связи с сервером, которые должны произойти в течение заданного `fail_timeout` времени для того, чтобы сервер считался недоступным; после этого он будет повторно проверен через то же самое время.

Что считается неудачной попыткой, определяется директивами `proxy_next_upstream`, `fastcgi_next_upstream`, `uwsgi_next_upstream`, `scgi_next_upstream`, `memcached_next_upstream` и `grpc_next_upstream`.

При превышении `max_fails` сервер также признаётся неработающим с точки зрения `upstream_probe`; клиентские запросы не будут направляться к нему, пока проверки не признают его работающим.

### Примечание

Если директива **server** в группе разрешается в несколько серверов, ее настройка `max_fails` применяется к каждому серверу отдельно.

Если после разрешения всех директив **server** в апстриме остается только один сервер, настройка `max_fails` не действует и будет проигнорирована.

|                    |                             |
|--------------------|-----------------------------|
| <b>max_fails=1</b> | Число попыток по умолчанию; |
| <b>max_fails=0</b> | Отключает учет попыток.     |

**fail\_timeout=время** — задаёт период времени, в течение которого должно произойти определенное число неудачных попыток связи с сервером (*max\_fails*), чтобы сервер считался недоступным. Затем сервер остается недоступным в течение того же самого времени, прежде чем будет проверен повторно.

Значение по умолчанию — 10 секунд.

#### Примечание

Если директива **server** в группе разрешается в несколько серверов, ее настройка **fail\_timeout** применяется к каждому серверу отдельно.

Если после разрешения всех директив **server** в апстриме остается только один сервер, настройка **fail\_timeout** не действует и будет проигнорирована.

|               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>backup</b> | Помечает сервер как запасной. На него будут передаваться запросы в случае, если не работают основные серверы.<br>Можно создать несколько групп резервирования серверов и задать уровень для каждой группы с помощью <b>backup=&lt;уровень_группы&gt;</b> . Например, <b>backup=1</b> (или просто <b>backup</b> ) — группа резервных серверов первого уровня, (в API отображается как <b>true</b> ), <b>backup=2</b> — группа резервных серверов второго уровня, (в API отображается как соответствующее число).<br>Если задана директива <b>backup_switch</b> , также применяется ее логика активного резервирования. |
| <b>down</b>   | Помечает сервер как постоянно недоступный.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <b>drain</b>  | Помечает сервер как разгружаемый ( <b>draining</b> ); это значит, что он получает только запросы сессий, привязанных ранее через <b>sticky</b> . В остальном поведение такое же, как в режиме <b>down</b> .                                                                                                                                                                                                                                                                                                                                                                                                           |

#### Предупреждение

Параметр **backup** нельзя использовать совместно с методами балансировки нагрузки **hash**, **ip\_hash** и **random**.

Параметры **down** и **drain** взаимно исключающие.

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>resolve</b>     | Позволяет отслеживать изменения списка IP-адресов, соответствующего доменному имени, и обновлять его без перезагрузки конфигурации. При этом группа должна находиться в <i>зоне разделяемой памяти</i> ; также должен быть определен <i>преобразователь имен в адреса</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| <b>service=имя</b> | Включает преобразование SRV-записей DNS и задает имя сервиса. Для работы параметра необходимо задать параметр <b>resolve</b> у сервера, не указывая порт сервера при имени хоста.<br>Если в имени службы нет точек, формируется имя по стандарту RFC: к имени службы добавляется префикс <b>_</b> , затем через точку добавляется <b>_tcp</b> . Так, имя службы <b>http</b> даст в результате <b>_http._tcp</b> .<br>Angie разрешает SRV-записи, объединяя нормализованное имя службы и имя хоста и получая список серверов для полученной комбинации через DNS, вместе с их приоритетами и весами. <ul style="list-style-type: none"> <li>SRV-записи с наивысшим приоритетом (те, которые имеют минимальное значение приоритета) разрешаются как основные серверы, а прочие записи становятся запасными серверами. Если <b>backup</b> установлено с <b>server</b>, SRV-записи с наивысшим приоритетом разрешаются как запасные серверы, а прочие записи игнорируются.</li> <li>Вес аналогичен параметру <b>weight</b> директивы <b>server</b>. Если вес задан как в самой директиве, так и в SRV-записи, используется вес, установленный в директиве.</li> </ul> |

В этом примере выполняется поиск записи `_http._tcp.backend.example.com`:

```
server backend.example.com service=http resolve;
```

|               |                                                                                                                                          |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------|
| <b>sid=id</b> | Задаёт ID сервера в группе. Если параметр не задан, то ID задается как шестнадцатеричный MD5-хэш IP-адреса и порта или пути UNIX-сокета. |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------|

|                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|-------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>slow_start=время</b> | <p>Задаёт <i>время</i> восстановления веса сервера, возвращающегося к работе при балансировке нагрузки методом <i>round-robin</i> или <i>least_conn</i>.</p> <p>Если параметр задан и сервер после сбоя снова считается работающим с точки зрения <i>max_fails</i> и <i>upstream_probe</i>, то такой сервер равномерно набирает указанный для него вес в течение заданного времени.</p> <p>Если параметр не задан, то в аналогичной ситуации сервер сразу начинает работу с указанным для него весом.</p> |
|-------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

#### Примечание

Если в апстриме задан только один `server`, `slow_start` не работает и будет игнорироваться.

## state

|                  |                          |
|------------------|--------------------------|
| <i>Синтаксис</i> | <code>state файл;</code> |
| По умолчанию     | —                        |
| <i>Контекст</i>  | <code>upstream</code>    |

Указывает *файл*, где постоянно хранится список серверов апстрима. Для хранения таких файлов специально создается каталог `/var/lib/angie/state/` (`/var/db/angie/state/` во FreeBSD) с соответствующими правами доступа, и в конфигурации остается добавить лишь имя файла:

```
upstream backend {
 zone backend 1m;
 state /var/lib/angie/state/<ИМЯ ФАЙЛА>;
}
```

Список серверов здесь имеет формат, аналогичный `server`. Содержимое файла изменяется при любом изменении серверов в разделе `/config/http/upstreams/` через API конфигурации. Файл считывается при запуске Angie ADC или перезагрузке конфигурации.

#### Предупреждение

Чтобы использовать директиву `state` в блоке `upstream`, в нем не должно быть директив `server`, но нужна зона разделяемой памяти (`zone`).

## sticky

|                  |                                                                                                                                                                                                                                                                                                                       |
|------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Синтаксис</i> | <pre>sticky cookie name [attr=значение]...; sticky route \$variable...; sticky learn zone=зона create=\$create_var1... lookup=\$lookup_var1... [header] [norefresh] [timeout=время]; sticky learn [zone=зона] lookup=\$lookup_var1... remote_action=uri remote_result=\$remote_var [norefresh] [timeout=время];</pre> |
| По умолчанию     | —                                                                                                                                                                                                                                                                                                                     |
| <i>Контекст</i>  | upstream                                                                                                                                                                                                                                                                                                              |

Настраивает привязку клиентских сессий к проксируемым серверам в режиме, заданном первым параметром; для разгрузки серверов, у которых задана директива **sticky**, можно использовать опцию **drain** в блоке *server*.

### Предупреждение

Директива **sticky** должна использоваться после всех директив, задающих тот или иной метод балансировки нагрузки, иначе она не будет работать. Если она используется наряду с директивой *bind\_conn*, то *bind\_conn* должна стоять после **sticky**.

## Режим cookie

Этот режим использует cookie для хранения сессий. Он подходит для ситуаций, когда cookie уже используются для управления сессиями.

Здесь запрос от клиента, пока не привязанного к какому-то серверу, отправляется на сервер, выбираемый согласно настроенному методу балансировки. При этом Angie устанавливает cookie с уникальным значением, идентифицирующим сервер.

Имя cookie (**name**) задается директивой **sticky**, а значение (**value**) соответствует параметру *sid* директивы *server*. Учтите, что параметр дополнительно хэшируется, если задана директива *sticky\_secret*.

Последующие запросы от клиента, содержащие такой cookie, передаются на сервер, заданный значением cookie, то есть сервер с указанным *sid*. Если выбрать сервер не удастся или выбранный сервер не может обработать запрос, то будет выбран другой сервер согласно настроенному методу балансировки.

Директива позволяет назначать атрибуты такого cookie; единственный атрибут, устанавливаемый по умолчанию, — **path=**/. Значения атрибутов задаются строками с переменными. Чтобы удалить атрибут, задайте для него пустое значение: **attr=**. Так, **sticky cookie path=** задает cookie без атрибута **path**.

Здесь Angie создает cookie **srv\_id** со сроком действия в 1 час и доменом, заданным переменной:

```
upstream backend {
 server backend1.example.com:8080;
 server backend2.example.com:8080;

 sticky cookie srv_id domain=$my_domain max-age=3600;
}
```

## Режим route

Этот режим использует предопределенные идентификаторы маршрутов, которые могут быть встроены в URL, cookie или другие свойства запроса. Он менее гибок, так как зависит от предопределенных значений, но лучше подходит, если такие идентификаторы уже используются.

Здесь проксируемый сервер при получении запроса может назначить клиенту маршрут и вернуть его идентификатор способом, известным им обоим. В качестве идентификатора маршрута должно использоваться значение параметра *sid* директивы *server*. Учтите, что параметр дополнительно хэшируется, если задана директива *sticky\_secret*.

Последующие запросы от клиентов, желающих использовать этот маршрут, должны содержать выданный сервером идентификатор, причем так, чтобы он попал в переменные Angie, например в *cookie* или *аргументы запроса*.

В параметрах директивы указываются переменные для маршрутизации. Чтобы выбрать сервер, куда передается поступивший запрос, используется первая непустая переменная; она затем сравнивается с параметром *sid* директивы *server*. Если выбрать сервер не удастся или выбранный сервер не может обработать запрос, то будет выбран другой сервер согласно настроенному методу балансировки.

Здесь Angie ищет идентификатор маршрута в *cookie route*, затем в аргументе запроса *route*:

```
upstream backend {
 server backend1.example.com:8080 "sid=server 1";
 server backend2.example.com:8080 "sid=server 2";

 sticky route $cookie_route $arg_route;
}
```

### Режим *learn*

В этом режиме для привязки клиента к конкретному проксируемому серверу используется динамически генерируемый ключ; этот режим более гибок, так как назначает серверы на ходу, хранит сесансы в зоне разделяемой памяти и поддерживает различные способы передачи идентификаторов сессий.

Здесь сессия создается на основе ответа проксируемого сервера. С параметрами *create* и *lookup* перечисляются переменные, указывающие, как создаются новые и ищутся существующие сессии. Оба параметра можно использовать по несколько раз.

Идентификатором сессии служит значение первой непустой переменной, указанной с *create*; например, это может быть *cookie с проксируемого сервера*.

Сессии хранятся в зоне разделяемой памяти; ее имя и размер задаются параметром *zone*. Если к сессии не было обращений в течение *времени timeout*, она удаляется. Значение по умолчанию — 1 час.

По умолчанию Angie продлевает срок действия сессии, обновляя метку времени последнего обращения при каждом использовании. Параметр *norefresh* отключает это поведение: сессия истечет строго по таймауту, даже если продолжает использоваться. Такой режим удобен, когда требуется принудительно завершать сессию по истечении времени, например, при интеграции с внешними менеджерами сессий.

Последующие запросы от клиентов, желающих использовать сессию, должны содержать ее идентификатор. Параметр *lookup* ищет идентификатор сессии в пользовательском запросе по заданному для него списку переменных, останавливаясь на первой непустой. Если ничего не найдено — запрос считается новым. Значение найденного идентификатора сопоставляется с сессиями в разделяемой памяти. Если выбрать сервер не удастся или выбранный сервер не может обработать запрос, то будет выбран другой сервер согласно настроенному методу балансировки.

Параметр *header* позволяет создать сессию сразу после получения заголовков ответа от проксируемого сервера. Без него сессия создается только после завершения обработки запроса.

В примере Angie создает сессию, устанавливая в ответе *cookie* с именем *examplecookie*:

```
upstream backend {
 server backend1.example.com:8080;
 server backend2.example.com:8080;
```

```

sticky learn
 create=$upstream_cookie_examplecookie
 lookup=$cookie_examplecookie
 zone=client_sessions:1m;
}

```

Режим `learn` с `remote_action`

Параметры `remote_action` и `remote_result` позволяют динамически назначать идентификаторы сессий и управлять ими с использованием удаленного хранилища сессий. При этом зона разделяемой памяти выступает в роли локального кэша, а удаленное хранилище является авторитетным источником. Поэтому параметр `create` несовместим с `remote_action`, так как идентификаторы сессий должны создаваться удаленно.

Если к сессии не было обращений в течение *времени* `timeout`, она удаляется. Значение по умолчанию — 1 час. Настройка `remote_action` не влияет на таймаут.

По умолчанию Angie продлевает срок действия сессии, обновляя метку времени последнего обращения при каждом ее использовании. Параметр `norefresh` меняет это поведение: сессия истекает строго по таймауту, даже если используется.

Параметр `zone` в конфигурации `sticky` с `remote_action` необязателен. Если он не задан, Angie полностью полагается на удаленное хранилище: не кэширует сессии локально (хотя позволяет кэшировать ответы хранилища через `proxy_cache`) и обращается к удаленному хранилищу всякий раз, когда требуется получить или создать сессию.

Общий принцип работы режима таков: если идентификатор сессии не найден локально или истек таймаут, Angie отправляет синхронный подзапрос в некое удаленное хранилище, заданное параметром `remote_action`.

При поступлении HTTP-запроса Angie выполняет следующие действия:

- Сначала извлекается идентификатор сессии из первой непустой переменной в списке `lookup`. Если все переменные пустые, используется обычный алгоритм балансировки нагрузки без привязки.
- Если задан параметр `zone`, Angie ищет существующую сессию в локальной разделяемой памяти. При обнаружении используется связанный с ней сервер и обработка завершается.
- Если сессия не найдена локально или параметр `zone` не задан, сервер выбирается как обычно, согласно настроенному методу балансировки. Затем Angie отправляет в удаленное хранилище, заданное параметром `remote_action`, синхронный HTTP-подзапрос, который должен содержать в понятном хранилищу виде:
  - идентификатор *сессии* из параметра `lookup` (в конфигурации это переменная `$sticky_sessid`);
  - идентификатор предварительно выбранного *сервера*: значение параметра `sid=` из директивы `server`, если оно задано, либо MD5-хэш имени сервера (в конфигурации это переменная `$sticky_sid`).

Проще всего можно передать эти идентификаторы через HTTP-заголовки с помощью `proxy_set_header`.

- Удаленное хранилище обрабатывает запрос и возвращает HTTP-ответ:

Ответ с кодом 200, 201 или 204 подтверждает выбранный сервер. Если задан параметр `zone`, сессия сохраняется в зоне разделяемой памяти.

Ответ с кодом 409 указывает на конфликт (лишь при наличии `zone`): данный идентификатор сессии уже существует, но связан с другим сервером. Удаленное хранилище должно одновременно с этим вернуть правильный идентификатор сервера в HTTP-заголовке; извлечь его можно через `remote_result`.



При получении от хранилища любого другого HTTP-кода (включая ошибки сети и таймауты) либо несуществующего идентификатора сервера Angie использует изначально выбранный сервер.

Идентификатор сервера извлекается из ответа удаленного хранилища через параметр `remote_result`: в нем можно указывать переменные с префиксом `upstream_http_`, которые создаются Angie автоматически для доступа к заголовкам HTTP-ответов от удаленного хранилища. Например, заголовок `X-Sid: server1` в таком ответе становится доступным в переменной `$upstream_http_x_sid` со значением `server1`.

В следующем примере Angie создает сессию, использует переменную `$cookie_bar` для начального идентификатора сессии, а альтернативные идентификаторы сессий, возвращенные удаленным хранилищем, сохраняет в `$upstream_http_x_sticky_sid`:

```
http {

 upstream u1 {

 server srv1;
 server srv2;

 sticky learn zone=sz:1m
 lookup=$cookie_bar
 remote_action=/remote_session
 remote_result=$upstream_http_x_sticky_sid;

 zone z 1m;
 }

 server {

 listen localhost;

 location / {

 proxy_pass http://u1/;
 }

 location /remote_session {

 internal;
 proxy_set_header X-Sticky-Sessid $sticky_sessid;
 proxy_set_header X-Sticky-Sid $sticky_sid;
 proxy_set_header X-Sticky-Last $msec;
 proxy_pass http://remote;
 }
 }
}
```

Ниже показан упрощенный пример конфигурации. Удаленное хранилище возвращает идентификатор сессии в заголовке `X-Sid` и таким образом подтверждает или переопределяет выбор Angie:

```
http {

 proxy_cache_path c1 keys_zone=s1:1m;

 upstream tc_0 {
 server 10.0.0.1 sid=web-server-01;
 server 10.0.0.2 sid=web-server-02;
 }
}
```



```

 sticky learn
 lookup=$arg_id
 remote_action=@create_session
 remote_result=$upstream_http_x_sid;
 }

 server {
 listen 127.0.0.1:8080;

 location / {
 proxy_pass http://tc_0/;
 }

 # Запрос к удаленному хранилищу сессий
 location @create_session {
 internal;

 proxy_set_header X-Sticky-Sessid $sticky_sessid;
 proxy_set_header X-Sticky-Sid $sticky_sid;
 proxy_set_header X-Sticky-Last $msec;

 proxy_pass http://session_backend;

 proxy_connect_timeout 1s;
 proxy_read_timeout 1s;

 proxy_cache s1;
 proxy_cache_valid 200 1d;
 proxy_cache_key "$scheme$proxy_host$request_uri$sticky_sessid";
 }
 }
}

```

Здесь при следующем ответе от удаленного хранилища:

```

HTTP/1.1 200 OK
...
X-Sid: web-server-01
X-Session-Backend: backend-pool-1

```

Становятся доступными две переменные:

- \$upstream\_http\_x\_sid, со значением web-server-01;
- \$upstream\_http\_x\_session\_backend, со значением backend-pool-1.

Так как переменная \$upstream\_http\_x\_sid указана в параметре remote\_result, то ее значение будет использовано для выбора сервера с sid=web-server-01.

Директива sticky учитывает состояние серверов в *upstream*:

- Серверы, помеченные как down или временно недоступные из-за сбоев, исключаются из выбора.
- Серверы, которые достигли максимального количества соединений (при использовании max\_conns), временно пропускаются.
- Серверы с опцией drain могут быть выбраны для создания новых сессий в режиме sticky при совпадении идентификаторов.

- Если ранее недоступный сервер восстанавливается, `sticky` автоматически возобновляет его использование.

Поведение `sticky` можно дополнительно настроить директивами `sticky_secret` и `sticky_strict`. Если в ходе работы `sticky` выбрать сервер не удастся или он недоступен, запрос будет обработан согласно выбранному методу балансировки нагрузки, если только не включена директива `sticky_strict`. В режиме `sticky_strict on`; запрос отклоняется с ошибкой.

Зоны разделяемой памяти, указываемые в параметре `zone` директивы `sticky`, не могут использоваться совместно различными `upstream`; каждая группа должна использовать свою собственную зону.

## sticky\_secret

|              |                                    |
|--------------|------------------------------------|
| Синтаксис    | <code>sticky_secret строка;</code> |
| По умолчанию | —                                  |
| Контекст     | <code>upstream</code>              |

Добавляет *строку* как соль в функцию MD5-хэширования для директивы `sticky` в режимах `cookie` и `route`. *Строка* может содержать переменные, например `$remote_addr`:

```
upstream backend {
 server backend1.example.com:8080;
 server backend2.example.com:8080;

 sticky cookie cookie_name;
 sticky_secret my_secret.$remote_addr;
}
```

Соль добавляется после хэшируемого значения; чтобы независимо проверить механизм хэширования:

```
$ echo -n "<VALUE><SALT>" | md5sum
```

## sticky\_strict

|              |                                      |
|--------------|--------------------------------------|
| Синтаксис    | <code>sticky_strict on   off;</code> |
| По умолчанию | <code>sticky_strict off;</code>      |
| Контекст     | <code>upstream</code>                |

При включении `Angie` будет возвращать клиенту ошибку HTTP 502, если желаемый сервер недоступен, вместо использования любого другого доступного сервера, как это происходит, когда в группе нет доступных серверов.

## upstream

|                  |                                   |
|------------------|-----------------------------------|
| <i>Синтаксис</i> | <code>upstream имя { ... }</code> |
| По умолчанию     | —                                 |
| <i>Контекст</i>  | http                              |

Описывает группу серверов. Серверы могут слушать на разных портах. Кроме того, можно одновременно использовать серверы, слушающие на TCP- и UNIX-сокетах.

Пример:

```
upstream backend {
 server backend1.example.com weight=5;
 server 127.0.0.1:8080 max_fails=3 fail_timeout=30s;
 server unix:/tmp/backend3;

 server backup1.example.com backup;
}
```

По умолчанию запросы распределяются по серверам циклически (в режиме round-robin) с учетом весов серверов. В вышеприведенном примере каждые 7 запросов будут распределены так: 5 запросов на backend1.example.com и по одному запросу на второй и третий серверы.

Если при попытке работы с сервером происходит ошибка, то запрос передается следующему серверу, и так далее до тех пор, пока не будут опробованы все работающие серверы. Если не удастся получить успешный ответ ни от одного из серверов, то клиенту будет возвращен результат работы с последним сервером.

## zone

|                  |                                 |
|------------------|---------------------------------|
| <i>Синтаксис</i> | <code>zone имя [размер];</code> |
| По умолчанию     | —                               |
| <i>Контекст</i>  | upstream                        |

Задаёт имя и размер зоны разделяемой памяти, в которой хранятся конфигурация группы и ее рабочее состояние, разделяемые между рабочими процессами. В одной и той же зоне могут быть сразу несколько групп. В этом случае достаточно указать размер только один раз.

## Встроенные переменные

Модуль `http_upstream` поддерживает следующие встроенные переменные:

`$sticky_sessid`

Используется с `remote_action` в *sticky*; хранит начальный идентификатор сессии, взятый из `lookup`.

`$sticky_sid`

Используется с `remote_action` в *sticky*; хранит идентификатор сервера, предварительно связанный с сессией.

`$upstream_addr`

хранит IP-адрес и порт или путь к UNIX-сокету сервера группы. Если при обработке запроса были сделаны обращения к нескольким серверам, то их адреса разделяются запятой, например:

192.168.1.1:80, 192.168.1.2:80, unix:/tmp/sock

Если произошло внутреннее перенаправление от одной группы серверов на другую с помощью X-Accel-Redirect или *error\_page*, то адреса, соответствующие разным группам серверов, разделяются двоеточием, например:

192.168.1.1:80, 192.168.1.2:80, unix:/tmp/sock : 192.168.10.1:80, 192.168.10.2:80

Если сервер не может быть выбран, то переменная хранит *имя группы серверов*.

`$upstream_bytes_received`

число байт, полученных от сервера группы. Значения нескольких соединений разделяются запятыми и двоеточиями подобно адресам в переменной *\$upstream\_addr*.

`$upstream_bytes_sent`

число байт, переданных на сервер группы. Значения нескольких соединений разделяются запятыми и двоеточиями подобно адресам в переменной *\$upstream\_addr*.

`$upstream_cache_status`

хранит статус доступа к кэшу ответов. Статус может быть одним из MISS, BYPASS, EXPIRED, STALE, UPDATING, REVALIDATED или HIT:

- MISS: Ответ не найден в кэше, и запрос передан на сервер.
- BYPASS: Кэш обойден, и запрос напрямую передан на сервер.
- EXPIRED: Ответ в кэше устарел, и на сервер передан новый запрос для обновления контента.
- STALE: Ответ в кэше устарел, но по-прежнему передается клиентам, пока через какое-то время не произойдет обновление контента с сервера.
- UPDATING: Ответ в кэше устарел, но по-прежнему передается клиентам, пока уже идущее обновление контента с сервера не завершится.
- REVALIDATED: Ответ в кэше устарел, но был успешно перепроверен и не нуждается в обновлении с сервера.
- HIT: Ответ был взят из кэша.

Если запрос пошел в обход кэша без обращения к нему, переменная не устанавливается.

#### `$upstream_connect_time`

хранит время, затраченное на установление соединения с сервером группы; время хранится в секундах с точностью до миллисекунд. В случае SSL включает в себя время, потраченное на рукопожатие. Времена нескольких соединений разделяются запятыми и двоеточиями подобно адресам в переменной `$upstream_addr`.

#### `$upstream_cookie_<имя>`

cookie с указанным именем, переданный сервером группы в поле **Set-Cookie** заголовка ответа. Необходимо иметь в виду, что cookie запоминаются только из ответа последнего сервера.

#### `$upstream_header_time`

хранит время, затраченное на получение заголовка ответа от сервера группы; время хранится в секундах с точностью до миллисекунд. Времена нескольких ответов разделяются запятыми и двоеточиями подобно адресам в переменной `$upstream_addr`.

#### `$upstream_http_<имя>`

хранят поля заголовка ответа сервера. Например, поле заголовка ответа **Server** доступно в переменной `$upstream_http_server`. Правила преобразования имен полей заголовка ответа в имена переменных такие же, как для переменных с префиксом `$http_`. Необходимо иметь в виду, что поля заголовка запоминаются только из ответа последнего сервера.

#### `$upstream_queue_time`

хранит время, проведенное запросом в *очереди* до очередного выбора сервера и выраженное в секундах с точностью до миллисекунд. Времена нескольких попыток разделяются запятыми и двоеточиями подобно адресам в переменной `$upstream_addr`.

#### `$upstream_response_length`

хранит длину ответа, полученного от сервера группы; длина хранится в байтах. Длины нескольких ответов разделяются запятыми и двоеточиями подобно адресам в переменной `$upstream_addr`.

#### `$upstream_response_time`

хранит время, затраченное на получение ответа от сервера группы; время хранится в секундах с точностью до миллисекунд. Времена нескольких ответов разделяются запятыми и двоеточиями подобно адресам в переменной `$upstream_addr`.

#### `$upstream_status`

хранит статус ответа, полученного от сервера группы. Статусы нескольких ответов разделяются запятыми и двоеточиями подобно адресам в переменной `$upstream_addr`. Если сервер не может быть выбран, то переменная хранит статус 502 (Bad Gateway).

#### `$upstream_sticky_status`

Статус привязанных запросов.

|      |                                                                              |
|------|------------------------------------------------------------------------------|
| ""   | Запрос отправлен в группу серверов, где привязка не включена.                |
| NEW  | Запрос не содержит информации о привязке к серверу.                          |
| HIT  | Запрос с привязкой отправлен на желаемый сервер.                             |
| MISS | Запрос с привязкой отправлен на сервер, выбранный по алгоритму балансировки. |

Статусы из нескольких соединений разделяются запятыми и двоеточиями подобно адресам в переменной `$upstream_addr`.

#### `$upstream_trailer_<имя>`

хранит поля из конца ответа, полученного от сервера группы.

### Upstream Probe

Реализует активные проверки работоспособности (health probes) для *Upstream*.

### Пример конфигурации

```
server {
 listen ...;

 location /backend {
 ...
 proxy_pass http://backend;

 upstream_probe backend_probe
 uri=/probe
 port=10004
 interval=5s
 test=$good
 essential
 fails=3
 passes=3
 max_body=10m
 mode=idle;
 }
}
```

## Директивы

### upstream\_probe

|                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Синтаксис</i> | <code>upstream_probe</code> <i>имя</i> [ <code>uri=адрес</code> ] [ <code>port=число</code> ] [ <code>interval=время</code> ] [ <code>method=метод</code> ] [ <code>test=условие</code> ] [ <code>ping</code> [ <code>ping_timeout=время</code> ]] [ <code>essential</code> [ <code>persistent</code> ]] [ <code>fails=число</code> ] [ <code>passes=число</code> ] [ <code>max_body=размер</code> ] [ <code>mode=always</code>   <code>idle</code>   <code>onfail</code> ]; |
| По умолчанию     | —                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <i>Контекст</i>  | location                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |

Задаёт активную проверку работоспособности серверов тех *upstream*, которые указаны в директивах *proxy\_pass*, *uwsgi\_pass* и т. д. в том же контексте *location*, где находится директива *upstream\_probe*. При этом Angie ADC регулярно выполняет запросы согласно указанным параметрам к каждому серверу в составе апстрима.

Сервер проходит проверку, если запрос к нему успешно выполняется с учетом всех параметров самой директивы *upstream\_probe* и всех параметров, влияющих на использование апстримов тем контекстом *location*, где она задана. Это касается в том числе директив *proxy\_next\_upstream*, *uwsgi\_next\_upstream* и пр., а также *proxy\_set\_header* и т. д.

Чтобы использовать проверки, в апстриме необходима зона разделяемой памяти (*zone*). Для одного апстрима можно определить несколько проверок.

Могут быть заданы следующие параметры:

|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| имя                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | Обязательное имя проверки.                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| uri                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | URI запроса, который добавляется к аргументу <i>proxy_pass</i> , <i>uwsgi_pass</i> и т. д. По умолчанию — <code>/</code> .                                                                                                                                                                                                                                                                                                                                         |
| port                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | Альтернативный порт для запроса.                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| interval                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | Интервал между проверками. По умолчанию — <code>5s</code> .                                                                                                                                                                                                                                                                                                                                                                                                        |
| method                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | HTTP-метод запроса проверки. По умолчанию — <code>GET</code> .                                                                                                                                                                                                                                                                                                                                                                                                     |
| test                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | Проверяемое при запросе условие; задается строкой с переменными. Если результат подстановки переменных — <code>"</code> или <code>0</code> , проверка не пройдена.                                                                                                                                                                                                                                                                                                 |
| ping                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | Использовать проверку методом ICMP Echo вместо HTTP-запроса. При выборе этого режима Angie ADC отправляет эхо-запрос (ICMP Echo Request) на IP-адрес сервера и ожидает ответ (Echo Reply). Проверка осуществляется по IP-адресу, а номер порта из <i>upstream</i> игнорируется.                                                                                                                                                                                    |
| <div> <i>i</i> <b>Примечание</b> </div> <div> <p>Параметры <code>uri</code>, <code>port</code>, <code>method</code> и <code>test</code> несовместимы с <code>ping</code> и игнорируются.</p> <p>На Linux рабочие процессы должны быть запущены от имени группы, разрешенной в <code>/proc/sys/net/ipv4/ping_group_range</code>. На FreeBSD необходимы права суперпользователя или наличие соответствующих <i>capability</i> для использования raw-сокетов.</p> <p>При использовании ICMP-проверок переменные, связанные с HTTP-запросом, всегда пусты, так как запрос фактически не выполняется; результат определяется только фактом получения ICMP Echo Reply.</p> </div> |                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| ping_timeout                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | <i>Время</i> ожидания ICMP-ответа; по умолчанию — <code>1s</code> . Используется только вместе с <code>ping</code> .                                                                                                                                                                                                                                                                                                                                               |
| essential                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | Если параметр задан, то изначально состояние сервера подлежит уточнению и клиентские запросы не передаются ему, пока проверка не будет пройдена.                                                                                                                                                                                                                                                                                                                   |
| persistent                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | Установка этого параметра требует сначала включить <code>essential</code> ; серверы с <code>persistent</code> , работавшие до <i>перезагрузки конфигурации</i> , начинают получать запросы без необходимости сначала пройти эту проверку.                                                                                                                                                                                                                          |
| fails                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | Число последовательных неуспешных запросов, при котором проверка считает сервер неработающим. По умолчанию — <code>1</code> .                                                                                                                                                                                                                                                                                                                                      |
| passes                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | Число последовательных успешных запросов, при котором проверка считает сервер работающим. По умолчанию — <code>1</code> .                                                                                                                                                                                                                                                                                                                                          |
| max_body                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | Максимальный объем памяти для тела ответа. По умолчанию — <code>256k</code> .                                                                                                                                                                                                                                                                                                                                                                                      |
| mode                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | Режим проверки в зависимости от работоспособности серверов: <ul style="list-style-type: none"> <li><code>always</code> — серверы проверяются независимо от состояния;</li> <li><code>idle</code> — проверяются неработающие серверы, а также серверы, где с последнего клиентского запроса прошло время <code>interval</code>.</li> <li><code>onfail</code> — проверяются серверы только в неработающем состоянии.</li> </ul> По умолчанию — <code>always</code> . |

Пример:

```
upstream backend {
 zone backend 1m;

 server backend1.example.com;
 server backend2.example.com;
}

map $upstream_status $good {
 200 "1";
}
```



```
server {
 listen ...;

 location /backend {
 ...
 proxy_pass http://backend;

 upstream_probe backend_probe
 uri=/probe
 port=10004
 interval=5s
 test=$good
 essential
 persistent
 fails=3
 passes=3
 max_body=10m
 mode=idle;
 }
}
```

Детали работы:

- Изначально сервер не получает клиентские запросы, пока не пройдет *все* заданные для него проверки с параметром **essential** (пропуская помеченные как **persistent**, если конфигурация перезагружена и до этого сервер считался работающим). Если таких проверок нет, сервер считается работающим.
- Сервер считается неработающим и не получает клиентские запросы, если *какая-либо* заданная для него проверка достигает своего порога **fails** или сам сервер достигает порога *max\_fails*.
- Чтобы неработающий сервер снова мог считаться работающим, *все* заданные для него проверки должны достичь своего порога **passes**; после этого учитывается порог *max\_fails*.

## Встроенные переменные

Модуль `http_upstream_probe` поддерживает следующие встроенные переменные:

`$upstream_probe`

Имя активной сейчас проверки *upstream\_probe*.

`$upstream_probe_body`

Тело ответа от сервера, полученного при проверке *upstream\_probe*. Его размер ограничен параметром `max_body`.

## UserID

Выдает cookie для идентификации клиентов. Для записи в лог полученных и выданных cookie можно использовать встроенные переменные `$uid_got` и `$uid_set`. Модуль совместим с модулем `mod_uid` для Apache.

### Пример конфигурации

```
userid on;
userid_name uid;
userid_domain example.com;
userid_path /;
userid_expires 365d;
userid_p3p 'policyref="/w3c/p3p.xml", CP="CUR ADM OUR NOR STA NID" ';
```

## Директивы

### userid

|                  |                                          |
|------------------|------------------------------------------|
| <i>Синтаксис</i> | <code>userid on   v1   log   off;</code> |
| По умолчанию     | <code>userid off;</code>                 |
| <i>Контекст</i>  | http, server, location                   |

Разрешает или запрещает выдачу cookie и запись приходящих cookie в лог:

|            |                                                                       |
|------------|-----------------------------------------------------------------------|
| <b>on</b>  | разрешает выдачу cookie версии 2 и запись приходящих cookie в лог;    |
| <b>v1</b>  | разрешает выдачу cookie версии 1 и запись приходящих cookie в лог;    |
| <b>log</b> | запрещает выдачу cookie, но разрешает запись приходящих cookie в лог; |
| <b>off</b> | запрещает выдачу cookie и запись приходящих cookie в лог.             |

### userid\_domain

|                  |                                        |
|------------------|----------------------------------------|
| <i>Синтаксис</i> | <code>userid_domain имя   none;</code> |
| По умолчанию     | <code>userid_domain none;</code>       |
| <i>Контекст</i>  | http, server, location                 |

Задаёт домен, для которого устанавливается cookie. Параметр `none` запрещает выдавать домен для cookie.

## userid\_expires

|                  |                                                |
|------------------|------------------------------------------------|
| <i>Синтаксис</i> | <code>userid_expires время   max   off;</code> |
| По умолчанию     | <code>userid_expires off;</code>               |
| <i>Контекст</i>  | http, server, location                         |

Задаёт время, в течение которого браузер должен хранить cookie. Параметр `max` устанавливает срок хранения cookie до 31 декабря 2037 года 23:55:55 GMT. Указание параметра `off` позволяет ограничить время действия cookie сессией браузера.

## userid\_flags

|                  |                                           |
|------------------|-------------------------------------------|
| <i>Синтаксис</i> | <code>userid_flags off   флаг ...;</code> |
| По умолчанию     | <code>userid_flags off;</code>            |
| <i>Контекст</i>  | http, server, location                    |

Если параметр не `off`, задаёт один или несколько дополнительных флагов для cookie: *secure*, *httponly*, *samesite=strict*, *samesite=lax*, *samesite=none*.

## userid\_mark

|                  |                                                   |
|------------------|---------------------------------------------------|
| <i>Синтаксис</i> | <code>userid_mark буква   цифра   =   off;</code> |
| По умолчанию     | <code>userid_mark off;</code>                     |
| <i>Контекст</i>  | http, server, location                            |

Если параметр не `off`, включает механизм маркировки cookie и задаёт символ, используемый в качестве метки. Этот механизм позволяет добавить или изменить *userid\_p3p* и/или время хранения cookie, но при этом оставить неизменным идентификатор клиента. Меткой может быть любая буква английского алфавита (с учетом регистра), цифра или знак "=".

Если метка задана, то она сравнивается с первым дополняющим символом в base64 представлении идентификатора клиента, передаваемом в cookie. Если они не совпадают, то cookie перепосылается с заданной меткой, временем хранения и заголовком РЗР.

## userid\_name

|                  |                               |
|------------------|-------------------------------|
| <i>Синтаксис</i> | <code>userid_name имя;</code> |
| По умолчанию     | <code>userid_name uid;</code> |
| <i>Контекст</i>  | http, server, location        |

Задаёт имя cookie.

## userid\_p3p

|                  |                                        |
|------------------|----------------------------------------|
| <i>Синтаксис</i> | <code>userid_p3p строка   none;</code> |
| По умолчанию     | <code>userid_p3p none;</code>          |
| <i>Контекст</i>  | <code>http, server, location</code>    |

Задаёт значение для поля заголовка P3P, которое будет выдаваться вместе с cookie. Если задано специальное значение **none**, то в ответе не будет заголовка P3P.

## userid\_path

|                  |                                     |
|------------------|-------------------------------------|
| <i>Синтаксис</i> | <code>userid_path путь;</code>      |
| По умолчанию     | <code>userid_path /;</code>         |
| <i>Контекст</i>  | <code>http, server, location</code> |

Задаёт путь, для которого устанавливается cookie.

## userid\_service

|                  |                                                       |
|------------------|-------------------------------------------------------|
| <i>Синтаксис</i> | <code>userid_service номер;</code>                    |
| По умолчанию     | <code>userid_service &lt;IP-адрес сервера&gt;;</code> |
| <i>Контекст</i>  | <code>http, server, location</code>                   |

Если идентификаторы выдаются несколькими серверами (сервисами), то каждому сервису следует назначить свой собственный **номер**, для обеспечения уникальности выдаваемых идентификаторов клиентов. По умолчанию для cookie первой версии используется ноль. Для cookie второй версии по умолчанию используется число, составленное из последних четырех октетов IP-адреса сервера.

## Встроенные переменные

`$uid_got`

Имя cookie и полученный идентификатор клиента.

`$uid_reset`

Если значением является непустая строка не равная 0, то клиентские идентификаторы перевыдаются. Специальное значение `log` дополнительно приводит к выдаче сообщений о перевыданных идентификаторах в *error\_log*.

`$uid_set`

Имя cookie и выданный идентификатор клиента.

## uWSGI

Позволяет передавать запросы uWSGI-серверу.

### Пример конфигурации

```
location / {
 include uwsgi_params;
 uwsgi_pass localhost:9000;
}
```

## Директивы

### uwsgi\_bind

|                  |                                                    |
|------------------|----------------------------------------------------|
| <i>Синтаксис</i> | <code>uwsgi_bind адрес [transparent]   off;</code> |
| По умолчанию     | —                                                  |
| <i>Контекст</i>  | http, server, location                             |

Задаёт локальный IP-адрес с необязательным портом, который будет использоваться в исходящих соединениях с uwsgi-сервером. В значении параметра допустимо использование переменных. Специальное значение `off` отменяет действие унаследованной с предыдущего уровня конфигурации директивы `uwsgi_bind`, позволяя системе самостоятельно выбирать локальный IP-адрес и порт.

Параметр `transparent` позволяет задать нелокальный IP-адрес, который будет использоваться в исходящих соединениях с uwsgi-сервером, например, реальный IP-адрес клиента:

```
uwsgi_bind $remote_addr transparent;
```

Для работы параметра обычно требуется запустить рабочие процессы Angie с привилегиями *суперпользователя*. В Linux это не требуется, так как если указан параметр `transparent`, то рабочие процессы наследуют *capability CAP\_NET\_RAW* из главного процесса.

#### Примечание

Необходимо настроить таблицу маршрутизации ядра для перехвата сетевого трафика с uwsgi-сервера.

## uwsgi\_buffer\_size

|                  |                                        |
|------------------|----------------------------------------|
| <i>Синтаксис</i> | <code>uwsgi_buffer_size размер;</code> |
| По умолчанию     | <code>uwsgi_buffer_size 4k 8k;</code>  |
| <i>Контекст</i>  | http, server, location                 |

Задаёт размер буфера, в который будет читаться первая часть ответа, получаемого от uwsgi-сервера. В этой части ответа обычно находится небольшой заголовок ответа. По умолчанию размер одного буфера равен размеру страницы памяти. В зависимости от платформы это или 4К, или 8К, однако его можно сделать меньше.

## uwsgi\_buffering

|                  |                                      |
|------------------|--------------------------------------|
| <i>Синтаксис</i> | <code>uwsgi_buffering размер;</code> |
| По умолчанию     | <code>uwsgi_buffering on;</code>     |
| <i>Контекст</i>  | http, server, location               |

|     |                                                                                                                                                                                                                                                                                                                                                                                                                        |
|-----|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| on  | Angie принимает ответ uwsgi-сервера как можно быстрее, сохраняя его в буферы, заданные директивами <code>uwsgi_buffer_size</code> и <code>uwsgi_buffers</code> . Если ответ не помещается целиком в память, то его часть может быть записана на диск во <i>временный файл</i> . Запись во временные файлы контролируется директивами <code>uwsgi_max_temp_file_size</code> и <code>uwsgi_temp_file_write_size</code> . |
| off | Ответ синхронно передается клиенту сразу же по мере его поступления. Angie не пытается считать весь ответ uwsgi-сервера. Максимальный размер данных, который Angie может принять от сервера за один раз, задается директивой <code>uwsgi_buffer_size</code> .                                                                                                                                                          |

Буферизация может быть также включена или выключена путем передачи значения "yes" или "no" в поле X-Accel-Buffering заголовка ответа. Эту возможность можно запретить с помощью директивы `uwsgi_ignore_headers`.

## uwsgi\_buffers

|                  |                                          |
|------------------|------------------------------------------|
| <i>Синтаксис</i> | <code>uwsgi_buffers число размер;</code> |
| По умолчанию     | <code>uwsgi_buffers 8 4k   8k;</code>    |
| <i>Контекст</i>  | http, server, location                   |

Задаёт число и размер буферов для одного соединения, в которые будет читаться ответ, получаемый от uwsgi-сервера.

По умолчанию размер одного буфера равен размеру страницы. В зависимости от платформы это или 4К, или 8К.

## uwsgi\_busy\_buffers\_size

|                  |                                                |
|------------------|------------------------------------------------|
| <i>Синтаксис</i> | <code>uwsgi_busy_buffers_size размер;</code>   |
| По умолчанию     | <code>uwsgi_busy_buffers_size 8k   16k;</code> |
| <i>Контекст</i>  | <code>http, server, location</code>            |

При включенной *буферизации* ответов uwsgi-сервера директива ограничивает суммарный размер буферов, которые могут быть заняты для отправки ответа клиенту, пока ответ еще не прочитан целиком. Оставшиеся буферы тем временем могут использоваться для чтения ответа и, при необходимости, буферизации части ответа во временный файл.

По умолчанию размер ограничен величиной двух буферов, заданных директивами `uwsgi_buffer_size` и `uwsgi_buffers`.

## uwsgi\_cache

|                  |                                      |
|------------------|--------------------------------------|
| <i>Синтаксис</i> | <code>uwsgi_cache зона   off;</code> |
| По умолчанию     | <code>uwsgi_cache off;</code>        |
| <i>Контекст</i>  | <code>http, server, location</code>  |

Задаёт зону разделяемой памяти, используемой для кэширования. Одна и та же зона может использоваться в нескольких местах. В значении параметра можно использовать переменные.

|                  |                                                                          |
|------------------|--------------------------------------------------------------------------|
| <code>off</code> | запрещает кэширование, унаследованное с предыдущего уровня конфигурации. |
|------------------|--------------------------------------------------------------------------|

## uwsgi\_cache\_background\_update

|                  |                                                      |
|------------------|------------------------------------------------------|
| <i>Синтаксис</i> | <code>uwsgi_cache_background_update on   off;</code> |
| По умолчанию     | <code>uwsgi_cache_background_update off;</code>      |
| <i>Контекст</i>  | <code>http, server, location</code>                  |

Позволяет запустить фоновый подзапрос для обновления просроченного элемента кэша, в то время как клиенту возвращается устаревший кэшированный ответ.

### Предупреждение

Использование устаревшего кэшированного ответа в момент его обновления должно быть *разрешено*.

## uwsgi\_cache\_bypass

|                  |                                      |
|------------------|--------------------------------------|
| <i>Синтаксис</i> | <code>uwsgi_cache_bypass ...;</code> |
| По умолчанию     | —                                    |
| <i>Контекст</i>  | http, server, location               |

Задаёт условия, при которых ответ не будет браться из кэша. Если значение хотя бы одного из строковых параметров непустое и не равно "0", то ответ не берётся из кэша:

```
uwsgi_cache_bypass $cookie_nocache $arg_nocache$arg_comment;
uwsgi_cache_bypass $http_pragma $http_authorization;
```

Можно использовать совместно с директивой `uwsgi_no_cache`.

## uwsgi\_cache\_key

|                  |                                      |
|------------------|--------------------------------------|
| <i>Синтаксис</i> | <code>uwsgi_cache_key строка;</code> |
| По умолчанию     | —                                    |
| <i>Контекст</i>  | http, server, location               |

Задаёт ключ для кэширования, например,

```
uwsgi_cache_key localhost:9000$request_uri;
```

## uwsgi\_cache\_lock

|                  |                                         |
|------------------|-----------------------------------------|
| <i>Синтаксис</i> | <code>uwsgi_cache_lock on   off;</code> |
| По умолчанию     | <code>uwsgi_cache_lock off;</code>      |
| <i>Контекст</i>  | http, server, location                  |

Если включено, одновременно только одному запросу будет позволено заполнить новый элемент кэша, идентифицируемый согласно директиве `uwsgi_cache_key`, передав запрос на uwsgi-сервер. Остальные запросы этого же элемента будут либо ожидать появления ответа в кэше, либо освобождения блокировки этого элемента, в течение времени, заданного директивой `uwsgi_cache_lock_timeout`.

## uwsgi\_cache\_lock\_age

|                  |                                          |
|------------------|------------------------------------------|
| <i>Синтаксис</i> | <code>uwsgi_cache_lock_age время;</code> |
| По умолчанию     | <code>uwsgi_cache_lock_age 5s;</code>    |
| <i>Контекст</i>  | http, server, location                   |

Если последний запрос, переданный на uwsgi-сервер для заполнения нового элемента кэша, не завершился за указанное время, на uwsgi-сервер может быть передан ещё один запрос.



## uwsgi\_cache\_lock\_timeout

|                  |                                              |
|------------------|----------------------------------------------|
| <i>Синтаксис</i> | <code>uwsgi_cache_lock_timeout время;</code> |
| По умолчанию     | <code>uwsgi_cache_lock_timeout 5s;</code>    |
| <i>Контекст</i>  | <code>http, server, location</code>          |

Задаёт таймаут для `uwsgi_cache_lock`. По истечении указанного времени запрос будет передан на uwsgi-сервер, однако ответ не будет кэширован.

## uwsgi\_cache\_max\_range\_offset

|                  |                                                  |
|------------------|--------------------------------------------------|
| <i>Синтаксис</i> | <code>uwsgi_cache_max_range_offset число;</code> |
| По умолчанию     | —                                                |
| <i>Контекст</i>  | <code>http, server, location</code>              |

Задаёт смещение в байтах для запросов с указанием диапазона запрашиваемых байт (byte-range requests). Если диапазон находится за указанным смещением, range-запрос будет передан на uwsgi-сервер и ответ не будет кэширован.

## uwsgi\_cache\_methods

|                  |                                                         |
|------------------|---------------------------------------------------------|
| <i>Синтаксис</i> | <code>uwsgi_cache_methods GET   HEAD   POST ...;</code> |
| По умолчанию     | <code>uwsgi_cache_methods GET HEAD;</code>              |
| <i>Контекст</i>  | <code>http, server, location</code>                     |

Если метод запроса клиента указан в этой директиве, то ответ будет кэширован. Методы "GET" и "HEAD" всегда добавляются в список, но тем не менее рекомендуется перечислять их явно. См. также директиву `uwsgi_no_cache`.

## uwsgi\_cache\_min\_uses

|                  |                                          |
|------------------|------------------------------------------|
| <i>Синтаксис</i> | <code>uwsgi_cache_min_uses число;</code> |
| По умолчанию     | <code>uwsgi_cache_min_uses 1;</code>     |
| <i>Контекст</i>  | <code>http, server, location</code>      |

Задаёт число запросов, после которого ответ будет кэширован.

### Предупреждение

Метаданные кэша хранятся в разделяемой памяти. Ручное удаление файлов кэша не сбрасывает счетчики и может привести к непредсказуемому поведению. Для полного сброса остановите сервер, удалите директорию кэша и запустите снова.

#### **Примечание**

Сторонние модули очистки кэша (например, external-cache-purge) удаляют только файлы, но не сбрасывают счетчик `uwsgi_cache_min_uses`. Директива предназначена для защиты кэша от загрязнения редкими запросами, и сброс счетчика при очистке может негативно повлиять на производительность.

### **uwsgi\_cache\_path**

|                  |                                                                                                                                                                                                                                                                                                                                                                                                   |
|------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Синтаксис</i> | <code>uwsgi_cache_path <i>путь</i> [levels=<i>уровни</i>] [use_temp_path=<i>on</i>   <i>off</i>] keys_zone=<i>имя:размер</i> [inactive=<i>время</i>] [max_size=<i>размер</i>] [min_free=<i>размер</i>] [manager_files=<i>число</i>] [manager_sleep=<i>время</i>] [manager_threshold=<i>время</i>] [loader_files=<i>число</i>] [loader_sleep=<i>время</i>] [loader_threshold=<i>время</i>];</code> |
| По умолчанию     | —                                                                                                                                                                                                                                                                                                                                                                                                 |
| <i>Контекст</i>  | http                                                                                                                                                                                                                                                                                                                                                                                              |

Задаёт путь и другие параметры кэша. Данные кэша хранятся в файлах. Именем файла в кэше является результат функции MD5 от *ключа кэширования*.

Параметр `levels` задаёт уровни иерархии кэша: можно задать от 1 до 3 уровней, на каждом уровне допускаются значения 1 или 2.

Например, при использовании

```
uwsgi_cache_path /data/angie/cache levels=1:2 keys_zone=one:10m;
```

имена файлов в кэше будут такого вида:

```
/data/angie/cache/c/29/b7f54b2df7773722d382f4809d65029c
```

Кэшируемый ответ сначала записывается во временный файл, а потом этот файл переименовывается. Временные файлы и кэш могут располагаться на разных файловых системах. Однако нужно учитывать, что в этом случае вместо дешевой операции переименовывания в пределах одной файловой системы файл копируется с одной файловой системы на другую. Поэтому лучше, если кэш будет находиться на той же файловой системе, что и каталог с временными файлами.

Какой из каталогов будет использоваться для временных файлов, определяется параметром `use_temp_path`.

|            |                                                                                                                                                                     |
|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>on</b>  | Если параметр не задан или установлен в значение "on", будет использоваться каталог, задаваемый директивой <code>uwsgi_temp_path</code> для данного <i>location</i> |
| <b>off</b> | временные файлы будут располагаться непосредственно в каталоге кэша                                                                                                 |

Кроме того, все активные ключи и информация о данных хранятся в зоне разделяемой памяти, имя и размер которой задаются параметром `keys_zone`. Зоны размером в 1 мегабайт достаточно для хранения около 8 тысяч ключей. Метаданные кэша хранятся в разделяемой памяти.

Если к данным кэша не обращаются в течение времени, заданного параметром `inactive`, то данные удаляются, независимо от их свежести.

По умолчанию `inactive` равен 10 минутам.

Специальный процесс **менеджера кэша** следит за максимальным размером кэша, а также за минимальным объемом свободного места на файловой системе с кэшем, и удаляет наименее востребованные данные при превышении максимального размера кэша или недостаточном объеме свободного места. Удаление данных происходит итерациями.

|                          |                                                                                                |
|--------------------------|------------------------------------------------------------------------------------------------|
| <b>max_size</b>          | максимальное пороговое значение размера кэша                                                   |
| <b>min_free</b>          | минимальное пороговое значение объема свободного места на файловой системе с кэшем             |
| <b>manager_files</b>     | максимальное количество удаляемых элементов кэша за одну итерацию<br>По умолчанию: 100         |
| <b>manager_threshold</b> | ограничивает время работы одной итерации<br>По умолчанию: 200 миллисекунд                      |
| <b>manager_sleep</b>     | время, в течение которого выдерживается пауза между итерациями<br>По умолчанию: 50 миллисекунд |

Через минуту после старта Angie активируется специальный процесс **загрузчика кэша**, который загружает в зону кэша информацию о ранее кэшированных данных, хранящихся на файловой системе. Загрузка также происходит итерациями.

|                         |                                                                                                |
|-------------------------|------------------------------------------------------------------------------------------------|
| <b>loader_files</b>     | максимальное количество элементов кэша к загрузке в одну итерацию<br>По умолчанию: 100         |
| <b>loader_threshold</b> | ограничивает время работы одной итерации<br>По умолчанию: 200 миллисекунд                      |
| <b>loader_sleep</b>     | время, в течение которого выдерживается пауза между итерациями<br>По умолчанию: 50 миллисекунд |

### uwsgi\_cache\_revalidate

|                  |                                               |
|------------------|-----------------------------------------------|
| <i>Синтаксис</i> | <code>uwsgi_cache_revalidate on   off;</code> |
| По умолчанию     | <code>uwsgi_cache_revalidate off;</code>      |
| <i>Контекст</i>  | <code>http, server, location</code>           |

Разрешает ревалидацию просроченных элементов кэша при помощи условных запросов с полями заголовка `If-Modified-Since` и `If-None-Match`.

### uwsgi\_cache\_use\_stale

|                  |                                                                                                                                                  |
|------------------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Синтаксис</i> | <code>uwsgi_cache_use_stale error   timeout   invalid_header   updating   http_500   http_503   http_403   http_404   http_429   off ...;</code> |
| По умолчанию     | <code>uwsgi_cache_use_stale off;</code>                                                                                                          |
| <i>Контекст</i>  | <code>http, server, location</code>                                                                                                              |

Определяет, в каких случаях можно использовать устаревший кэшированный ответ. Параметры директивы совпадают с параметрами директивы `uwsgi_next_upstream`.

|                 |                                                                                                                                                                                                                            |
|-----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>error</b>    | Позволяет использовать устаревший кэшированный ответ при невозможности выбора uwsgi-сервера для обработки запроса.                                                                                                         |
| <b>updating</b> | Дополнительный параметр, разрешает использовать устаревший кэшированный ответ, если на данный момент он уже обновляется. Это позволяет минимизировать число обращений к uwsgi-серверам при обновлении кэшированных данных. |

Использование устаревшего кэшированного ответа может также быть разрешено непосредственно в заголовке ответа на определенное количество секунд после того, как ответ устарел:

- Расширение `stale-while-revalidate` поля заголовка `Cache-Control` разрешает использовать устаревший кэшированный ответ, если на данный момент он уже обновляется.
- Расширение `stale-if-error` поля заголовка `Cache-Control` разрешает использовать устаревший кэшированный ответ в случае ошибки.

#### **i** Примечание

Такой способ менее приоритетен, чем задание параметров директивы.

Чтобы минимизировать число обращений к uwsgi-серверам при заполнении нового элемента кэша, можно воспользоваться директивой `uwsgi_cache_lock`.

### `uwsgi_cache_valid`

|                  |                                                 |
|------------------|-------------------------------------------------|
| <i>Синтаксис</i> | <code>uwsgi_cache_valid [код ...] время;</code> |
| По умолчанию     | —                                               |
| <i>Контекст</i>  | <code>http, server, location</code>             |

Задаёт время кэширования для разных кодов ответа. Например, директивы

```
uwsgi_cache_valid 200 302 10m;
uwsgi_cache_valid 404 1m;
```

задают время кэширования 10 минут для ответов с кодами 200 и 302 и 1 минуту для ответов с кодом 404.

Если указано только время кэширования,

```
uwsgi_cache_valid 5m;
```

то кэшируются только ответы 200, 301 и 302.

Кроме того, можно кэшировать любые ответы с помощью параметра `any`:

```
uwsgi_cache_valid 200 302 10m;
uwsgi_cache_valid 301 1h;
uwsgi_cache_valid any 1m;
```

#### **i** Примечание

Параметры кэширования могут также быть заданы непосредственно в заголовке ответа. Такой способ приоритетнее, чем задание времени кэширования с помощью директивы.

- Поле заголовка `X-Accel-Expires` задаёт время кэширования ответа в секундах. Значение `0` запрещает кэшировать ответ. Если значение начинается с префикса `@`, оно задаёт абсолютное время в секундах с начала эпохи, до которого ответ может быть кэширован.
- Если в заголовке нет поля `X-Accel-Expires`, параметры кэширования определяются по полям заголовка `Expires` или `Cache-Control`.
- Ответ, в заголовке которого есть поле `Set-Cookie`, не будет кэшироваться.

- Ответ, в заголовке которого есть поле **Vary** со специальным значением "\*", не будет кэшироваться. Ответ, в заголовке которого есть поле **Vary** с другим значением, будет кэширован с учетом соответствующих полей заголовка запроса.

Обработка одного или более из этих полей заголовка может быть отключена при помощи директивы `uwsgi_ignore_headers`.

### **uwsgi\_connect\_timeout**

|                  |                                           |
|------------------|-------------------------------------------|
| <i>Синтаксис</i> | <code>uwsgi_connect_timeout время;</code> |
| По умолчанию     | <code>uwsgi_connect_timeout 60s;</code>   |
| <i>Контекст</i>  | <code>http, server, location</code>       |

Задаёт таймаут для установления соединения с uwsgi-сервером. Необходимо иметь в виду, что этот таймаут обычно не может превышать 75 секунд.

### **uwsgi\_connection\_drop**

|                  |                                                      |
|------------------|------------------------------------------------------|
| <i>Синтаксис</i> | <code>uwsgi_connection_drop время   on   off;</code> |
| По умолчанию     | <code>uwsgi_connection_drop off;</code>              |
| <i>Контекст</i>  | <code>http, server, location</code>                  |

Настраивает завершение всех соединений с проксируемым сервером, если он был удален из группы или помечен как постоянно недоступный в результате процесса *reresholve* или команды *API DELETE*.

Соединение завершается, когда обрабатывается следующее событие чтения или записи для клиента или проксируемого сервера.

Установка *времени* включает *таймаут* до завершения соединения; при выборе значения *on* соединения завершаются немедленно.

### **uwsgi\_force\_ranges**

|                  |                                           |
|------------------|-------------------------------------------|
| <i>Синтаксис</i> | <code>uwsgi_force_ranges on   off;</code> |
| По умолчанию     | <code>uwsgi_force_ranges off;</code>      |
| <i>Контекст</i>  | <code>http, server, location</code>       |

Включает поддержку диапазонов запрашиваемых байт (byte-range) для кэшированных и некешированных ответов uwsgi-сервера вне зависимости от наличия поля **Accept-Ranges** в заголовках этих ответов.

## uwsgi\_hide\_header

|                  |                         |
|------------------|-------------------------|
| <i>Синтаксис</i> | uwsgi_hide_header поле; |
| По умолчанию     | —                       |
| <i>Контекст</i>  | http, server, location  |

По умолчанию Angie не передает клиенту поля заголовка **Date**, **Server**, **X-Pad** и **X-Accel-...** из ответа uwsgi-сервера. Директива *uwsgi\_hide\_header* задает дополнительные поля, которые не будут передаваться. Если же передачу полей нужно разрешить, можно воспользоваться директивой *uwsgi\_pass\_header*.

## uwsgi\_ignore\_client\_abort

|                  |                                     |
|------------------|-------------------------------------|
| <i>Синтаксис</i> | uwsgi_ignore_client_abort on   off; |
| По умолчанию     | uwsgi_ignore_client_abort off;      |
| <i>Контекст</i>  | http, server, location              |

Определяет, закрывать ли соединение с uwsgi-сервером в случае, если клиент закрыл соединение, не дождавшись ответа.

## uwsgi\_ignore\_headers

|                  |                                |
|------------------|--------------------------------|
| <i>Синтаксис</i> | uwsgi_ignore_headers поле ...; |
| По умолчанию     | —                              |
| <i>Контекст</i>  | http, server, location         |

Запрещает обработку некоторых полей заголовка из ответа uwsgi-сервера. В директиве можно указать поля **X-Accel-Redirect**, **X-Accel-Expires**, **X-Accel-Limit-Rate**, **X-Accel-Buffering**, **X-Accel-Charset**, **Expires**, **Cache-Control**, **Set-Cookie** и **Vary**.

Если не запрещено, обработка этих полей заголовка заключается в следующем:

- **X-Accel-Expires**, **Expires**, **Cache-Control**, **Set-Cookie** и **Vary** задают *параметры кэширования* ответа;
- **X-Accel-Redirect** производит *внутреннее перенаправление* на указанный URI;
- **X-Accel-Limit-Rate** задает *ограничение скорости* передачи ответа клиенту;
- **X-Accel-Buffering** включает или выключает *буферизацию* ответа;
- **X-Accel-Charset** задает желаемую *кодировку* ответа.

## uwsgi\_intercept\_errors

|                  |                                  |
|------------------|----------------------------------|
| <i>Синтаксис</i> | uwsgi_intercept_errors on   off; |
| По умолчанию     | uwsgi_intercept_errors off;      |
| <i>Контекст</i>  | http, server, location           |

Определяет, передавать ли клиенту ответы uwsgi-сервера с кодом больше либо равным 300, или же перехватывать их и перенаправлять на обработку Angie с помощью директивы *error\_page*.

## uwsgi\_limit\_rate

|                  |                                    |
|------------------|------------------------------------|
| <i>Синтаксис</i> | uwsgi_limit_rate <i>скорость</i> ; |
| По умолчанию     | uwsgi_limit_rate 0;                |
| <i>Контекст</i>  | http, server, location             |

Ограничивает скорость чтения ответа от uwsgi-сервера. *Скорость* задается в байтах в секунду; можно использовать переменные.

|   |                                |
|---|--------------------------------|
| 0 | отключает ограничение скорости |
|---|--------------------------------|

### **i** Примечание

Ограничение устанавливается на запрос, поэтому, если Angie одновременно откроет два соединения к uwsgi-серверу, суммарная скорость будет вдвое выше заданного ограничения. Ограничение работает только в случае, если включена *буферизация* ответов uwsgi-сервера.

## uwsgi\_max\_temp\_file\_size

|                  |                                          |
|------------------|------------------------------------------|
| <i>Синтаксис</i> | uwsgi_max_temp_file_size <i>размер</i> ; |
| По умолчанию     | uwsgi_max_temp_file_size 1024m;          |
| <i>Контекст</i>  | http, server, location                   |

Если включена *буферизация* ответов uwsgi-сервера, и ответ не помещается целиком в буферы, заданные директивами *uwsgi\_buffer\_size* и *uwsgi\_buffers*, часть ответа может быть записана во временный файл. Эта директива задает максимальный размер временного файла. Размер данных, сбрасываемых во временный файл за один раз, задается директивой *uwsgi\_temp\_file\_write\_size*.

|   |                                                              |
|---|--------------------------------------------------------------|
| 0 | отключает возможность буферизации ответов во временные файлы |
|---|--------------------------------------------------------------|

### **i** Примечание

Данное ограничение не распространяется на ответы, которые будут *кэшированы* или сохранены на диске.

## uwsgi\_modifier1

|                  |                                     |
|------------------|-------------------------------------|
| <i>Синтаксис</i> | <code>uwsgi_modifier1 число;</code> |
| По умолчанию     | <code>uwsgi_modifier1 0;</code>     |
| <i>Контекст</i>  | <code>http, server, location</code> |

Задаёт значение поля `modifier1` в заголовке пакета `uwsgi`.

## uwsgi\_modifier2

|                  |                                     |
|------------------|-------------------------------------|
| <i>Синтаксис</i> | <code>uwsgi_modifier2 число;</code> |
| По умолчанию     | <code>uwsgi_modifier2 0;</code>     |
| <i>Контекст</i>  | <code>http, server, location</code> |

Задаёт значение поля `modifier2` в заголовке пакета `uwsgi`.

## uwsgi\_next\_upstream

|                  |                                                                                                                                                      |
|------------------|------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Синтаксис</i> | <code>uwsgi_next_upstream error   timeout   invalid_header   http_500   http_503   http_403   http_404   http_429   non_idempotent   off ...;</code> |
| По умолчанию     | <code>uwsgi_next_upstream error timeout;</code>                                                                                                      |
| <i>Контекст</i>  | <code>http, server, location</code>                                                                                                                  |

Определяет, в каких случаях запрос будет передан следующему в группе *upstream* серверу:

|                       |                                                                                                                                                                                                                                           |
|-----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>error</b>          | произошла ошибка соединения с сервером, передачи ему запроса или чтения заголовка ответа сервера;                                                                                                                                         |
| <b>timeout</b>        | произошёл таймаут во время соединения с сервером, передачи ему запроса или чтения заголовка ответа сервера;                                                                                                                               |
| <b>invalid_header</b> | сервер вернул пустой или неверный ответ;                                                                                                                                                                                                  |
| <b>http_500</b>       | сервер вернул ответ с кодом 500;                                                                                                                                                                                                          |
| <b>http_503</b>       | сервер вернул ответ с кодом 503;                                                                                                                                                                                                          |
| <b>http_403</b>       | сервер вернул ответ с кодом 403;                                                                                                                                                                                                          |
| <b>http_404</b>       | сервер вернул ответ с кодом 404;                                                                                                                                                                                                          |
| <b>http_429</b>       | сервер вернул ответ с кодом 429;                                                                                                                                                                                                          |
| <b>non_idempotent</b> | обычно запросы с <b>неидемпотентным</b> методом ( <i>POST</i> , <i>LOCK</i> , <i>PATCH</i> ) не передаются на другой сервер, если запрос серверу группы уже был отправлен; включение параметра явно разрешает повторять подобные запросы; |
| <b>off</b>            | запрещает передачу запроса следующему серверу.                                                                                                                                                                                            |

### Примечание

Необходимо понимать, что передача запроса следующему серверу возможна только при условии, что клиенту ещё ничего не передавалось. То есть, если ошибка или таймаут возникли в середине передачи ответа клиенту, то действие директивы на такой запрос не распространяется.



Директива также определяет, что считается *неудачной попыткой* работы с сервером.

|                |                                                                             |
|----------------|-----------------------------------------------------------------------------|
| error          | всегда считаются неудачными попытками, даже если они не указаны в директиве |
| timeout        |                                                                             |
| invalid_header |                                                                             |
| http_500       | считаются неудачными попытками, только если они указаны в директиве         |
| http_503       |                                                                             |
| http_429       |                                                                             |
| http_403       | никогда не считаются неудачными попытками                                   |
| http_404       |                                                                             |

Передача запроса следующему серверу может быть ограничена по *количеству попыток* и по *времени*.

### uwsgi\_next\_upstream\_timeout

|                  |                                            |
|------------------|--------------------------------------------|
| <i>Синтаксис</i> | uwsgi_next_upstream_timeout <i>время</i> ; |
| По умолчанию     | uwsgi_next_upstream_timeout 0;             |
| <i>Контекст</i>  | http, server, location                     |

Ограничивает время, в течение которого возможна передача запроса *следующему* серверу.

|   |                           |
|---|---------------------------|
| 0 | отключает это ограничение |
|---|---------------------------|

### uwsgi\_next\_upstream\_tries

|                  |                                          |
|------------------|------------------------------------------|
| <i>Синтаксис</i> | uwsgi_next_upstream_tries <i>число</i> ; |
| По умолчанию     | uwsgi_next_upstream_tries 0;             |
| <i>Контекст</i>  | http, server, location                   |

Ограничивает число допустимых попыток для передачи запроса *следующему* серверу.

|   |                           |
|---|---------------------------|
| 0 | отключает это ограничение |
|---|---------------------------|

### uwsgi\_no\_cache

|                  |                                   |
|------------------|-----------------------------------|
| <i>Синтаксис</i> | uwsgi_no_cache <i>строка</i> ...; |
| По умолчанию     | —                                 |
| <i>Контекст</i>  | http, server, location            |

Задаёт условия, при которых ответ не будет сохраняться в кэш. Если значение хотя бы одного из строковых параметров непустое и не равно "0", то ответ не будет сохранен:

```
uwsgi_no_cache $cookie_nocache $arg_nocache$arg_comment;
uwsgi_no_cache $http_pragma $http_authorization;
```

Можно использовать совместно с директивой `uwsgi_cache_bypass`.

## uwsgi\_param

|                       |                                                            |
|-----------------------|------------------------------------------------------------|
| <i>Синтаксис</i>      | <code>uwsgi_param параметр значение [if_not_empty];</code> |
| Значение по умолчанию | —                                                          |
| <i>Контекст</i>       | http, server, location                                     |

Устанавливает параметр, который будет передан серверу uwsgi. Значение может содержать текст, переменные или их комбинацию. Эти директивы наследуются с предыдущего уровня конфигурации только в том случае, если на текущем уровне директивы `uwsgi_param` отсутствуют.

Стандартные [переменные окружения CGI](#) следует передавать в виде заголовков uwsgi, как показано в файле `uwsgi_params`, поставляемом вместе с дистрибутивом:

```
location / {
 include uwsgi_params;
 # ...
}
```

Если директива указана с параметром `if_not_empty`, такой параметр будет передан серверу только в случае, если его значение не является пустым:

```
uwsgi_param HTTPS $https if_not_empty;
```

## uwsgi\_pass

|                  |                                              |
|------------------|----------------------------------------------|
| <i>Синтаксис</i> | <code>uwsgi_pass [протокол://] адрес;</code> |
| По умолчанию     | —                                            |
| <i>Контекст</i>  | location, if в location                      |

Задаёт протокол и адрес uwsgi-сервера. В качестве протокола можно указать `uwsgi` или `suwsgi` (secured uwsgi, uwsgi через SSL). Адрес может быть указан в виде доменного имени или IP-адреса, и порта:

```
uwsgi_pass localhost:9000;
uwsgi_pass uwsgi://localhost:9000;
uwsgi_pass suwsgi://[2001:db8::1]:9090;
```

или в виде пути UNIX-сокета, который указывается после слова `unix` и заключается в двоеточия:

```
uwsgi_pass unix:/tmp/uwsgi.socket;
```

Если доменному имени соответствует несколько адресов, то все они будут использоваться по очереди (round-robin). Кроме того, в качестве адреса можно указать [группу серверов](#).

В значении параметра можно использовать переменные. В этом случае, если адрес указан в виде доменного имени, имя ищется среди описанных групп серверов и если не найдено, то определяется с помощью `resolver`'а.

#### **Примечание**

Если `uwsgi_pass` стоит в `location` с косой чертой в конце префикса (например, `location /name/`), и при этом в директиве `auto_redirect` указано `default`, запросы без косой черты в конце будут перенаправляться (`/name -> /name/`).

### **uwsgi\_pass\_header**

|                  |                                          |
|------------------|------------------------------------------|
| <i>Синтаксис</i> | <code>uwsgi_pass_header поле ...;</code> |
| По умолчанию     | —                                        |
| <i>Контекст</i>  | <code>http, server, location</code>      |

Разрешает передавать от uwsgi-сервера клиенту *запрещенные для передачи* поля заголовка.

### **uwsgi\_pass\_request\_body**

|                  |                                                |
|------------------|------------------------------------------------|
| <i>Синтаксис</i> | <code>uwsgi_pass_request_body on   off;</code> |
| По умолчанию     | <code>uwsgi_pass_request_body on;</code>       |
| <i>Контекст</i>  | <code>http, server, location</code>            |

Позволяет запретить передачу исходного тела запроса на uwsgi-сервер. См. также директиву `uwsgi_pass_request_headers`.

### **uwsgi\_pass\_request\_headers**

|                  |                                                   |
|------------------|---------------------------------------------------|
| <i>Синтаксис</i> | <code>uwsgi_pass_request_headers on   off;</code> |
| По умолчанию     | <code>uwsgi_pass_request_headers on;</code>       |
| <i>Контекст</i>  | <code>http, server, location</code>               |

Позволяет запретить передачу полей заголовка исходного запроса на uwsgi-сервер. См. также директиву `uwsgi_pass_request_body`.

### **uwsgi\_read\_timeout**

|                  |                                        |
|------------------|----------------------------------------|
| <i>Синтаксис</i> | <code>uwsgi_read_timeout время;</code> |
| По умолчанию     | <code>uwsgi_read_timeout 60s;</code>   |
| <i>Контекст</i>  | <code>http, server, location</code>    |

Задаёт таймаут при чтении ответа uwsgi-сервера. Таймаут устанавливается не на всю передачу ответа, а только между двумя операциями чтения. Если по истечении этого времени uwsgi-сервер ничего не передаст, соединение закрывается.

## uwsgi\_request\_buffering

|                  |                                   |
|------------------|-----------------------------------|
| <i>Синтаксис</i> | uwsgi_request_buffering on   off; |
| По умолчанию     | uwsgi_request_buffering on;       |
| <i>Контекст</i>  | http, server, location            |

Разрешает или запрещает использовать буферизацию тела запроса клиента.

|     |                                                                                                                                                                                                |
|-----|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| on  | тело запроса полностью <i>читается</i> от клиента перед отправкой запроса на uwsgi-сервер.                                                                                                     |
| off | тело запроса отправляется на uwsgi-сервер сразу же по мере его поступления. В этом случае запрос не может быть передан <i>следующему серверу</i> , если Angie уже начал отправку тела запроса. |

Если для отправки тела исходного запроса используется HTTP/1.1 и передача данных частями (chunked transfer encoding), то тело запроса буферизуется независимо от значения директивы.

## uwsgi\_send\_timeout

|                  |                                   |
|------------------|-----------------------------------|
| <i>Синтаксис</i> | uwsgi_send_timeout <i>время</i> ; |
| По умолчанию     | uwsgi_send_timeout 60s;           |
| <i>Контекст</i>  | http, server, location            |

Задаёт таймаут при передаче запроса uwsgi-серверу. Таймаут устанавливается не на всю передачу запроса, а только между двумя операциями записи. Если по истечении этого времени uwsgi-сервер не примет новых данных, соединение закрывается.

## uwsgi\_socket\_keepalive

|                  |                                  |
|------------------|----------------------------------|
| <i>Синтаксис</i> | uwsgi_socket_keepalive on   off; |
| По умолчанию     | uwsgi_socket_keepalive off;      |
| <i>Контекст</i>  | http, server, location           |

Конфигурирует поведение "TCP keepalive" для исходящих соединений к uwsgi-серверу.

|     |                                                                   |
|-----|-------------------------------------------------------------------|
| " " | По умолчанию для сокета действуют настройки операционной системы. |
| on  | для сокета включается параметр <i>SO_KEEPALIVE</i>                |

## uwsgi\_ssl\_certificate

|                  |                                     |
|------------------|-------------------------------------|
| <i>Синтаксис</i> | uwsgi_ssl_certificate <i>файл</i> ; |
| По умолчанию     | —                                   |
| <i>Контекст</i>  | http, server, location              |

Задаёт файл с сертификатом в формате PEM для аутентификации на suwsgi-сервере. В имени файла можно использовать переменные.

## uwsgi\_ssl\_certificate\_cache

|                       |                                                                                                                               |
|-----------------------|-------------------------------------------------------------------------------------------------------------------------------|
| <i>Синтаксис</i>      | uwsgi_ssl_certificate_cache off;<br>uwsgi_ssl_certificate_cache max= <i>N</i> [inactive= <i>time</i> ] [valid= <i>time</i> ]; |
| Значение по умолчанию | uwsgi_ssl_certificate_cache off;                                                                                              |
| <i>Контекст</i>       | http, server, location                                                                                                        |

Определяет кэш для хранения *SSL-сертификатов* и *секретных ключей*, заданных через переменные.

Директива поддерживает следующие параметры:

- **max** — устанавливает максимальное количество элементов в кэше. При переполнении кэша удаляются наименее недавно использованные (LRU) элементы.
- **inactive** — определяет время, после которого элемент будет удален, если к нему не было обращений. Значение по умолчанию — 10 секунд.
- **valid** — определяет время, в течение которого элемент кэша считается действительным и может использоваться повторно. Значение по умолчанию — 60 секунд. По истечении этого времени сертификаты перезагружаются или проходят повторную проверку.
- **off** — отключает кэш.

Пример:

```
uwsgi_ssl_certificate $uwsgi_ssl_server_name.crt;
uwsgi_ssl_certificate_key $uwsgi_ssl_server_name.key;
uwsgi_ssl_certificate_cache max=1000 inactive=20s valid=1m;
```

## uwsgi\_ssl\_certificate\_key

|                  |                                         |
|------------------|-----------------------------------------|
| <i>Синтаксис</i> | uwsgi_ssl_certificate_key <i>файл</i> ; |
| По умолчанию     | —                                       |
| <i>Контекст</i>  | http, server, location                  |

Задаёт файл с секретным ключом в формате PEM для аутентификации на suwsgi-сервере.

Вместо файла можно указать значение "engine:*имя*:*id*", которое загружает ключ с указанным *id* из OpenSSL engine с заданным именем.

В имени файла можно использовать переменные.

## uwsgi\_ssl\_ciphers

|                  |                                         |
|------------------|-----------------------------------------|
| <i>Синтаксис</i> | <code>uwsgi_ssl_ciphers шифры;</code>   |
| По умолчанию     | <code>uwsgi_ssl_ciphers DEFAULT;</code> |
| <i>Контекст</i>  | http, server, location                  |

Описывает разрешенные шифры для запросов к uwsgi-серверу. Шифры задаются в формате, поддерживаемом библиотекой OpenSSL.

Список шифров зависит от установленной версии OpenSSL. Полный список можно посмотреть с помощью команды `openssl ciphers`.

### ⚠ Предупреждение

Директива `uwsgi_ssl_ciphers` не настраивает шифры для TLS 1.3 при использовании OpenSSL. Для настройки шифров TLS 1.3 в OpenSSL используйте директиву `uwsgi_ssl_conf_command`, добавленную для расширенной конфигурации SSL.

- В LibreSSL шифры TLS 1.3 можно настраивать с помощью `uwsgi_ssl_ciphers`.
- В BoringSSL шифры TLS 1.3 настроить невозможно.

## uwsgi\_ssl\_conf\_command

|                  |                                                   |
|------------------|---------------------------------------------------|
| <i>Синтаксис</i> | <code>uwsgi_ssl_conf_command имя значение;</code> |
| По умолчанию     | —                                                 |
| <i>Контекст</i>  | http, server, location                            |

Задаёт произвольные конфигурационные команды OpenSSL при установлении соединения с uwsgi HTTPS-сервером.

### ℹ Примечание

Директива поддерживается при использовании OpenSSL 1.0.2 и выше. Чтобы настроить шифры TLS 1.3 в OpenSSL, используйте команду `ciphersuites`.

На одном уровне может быть указано несколько директив `uwsgi_ssl_conf_command`. Директивы наследуются с предыдущего уровня конфигурации при условии, что на данном уровне не описаны свои директивы `uwsgi_ssl_conf_command`.

### ⚠ Предупреждение

Следует учитывать, что изменение настроек OpenSSL напрямую может привести к неожиданному поведению.

## uwsgi\_ssl\_crl

|                  |                             |
|------------------|-----------------------------|
| <i>Синтаксис</i> | uwsgi_ssl_crl <i>файл</i> ; |
| По умолчанию     | —                           |
| <i>Контекст</i>  | http, server, location      |

Указывает файл с отозванными сертификатами (CRL) в формате PEM, используемыми при *проверке* сертификата suwsgi-сервера.

## uwsgi\_ssl\_name

|                  |                                           |
|------------------|-------------------------------------------|
| <i>Синтаксис</i> | uwsgi_ssl_name <i>имя</i> ;               |
| По умолчанию     | uwsgi_ssl_name `имя хоста из uwsgi_pass`; |
| <i>Контекст</i>  | http, server, location                    |

Позволяет переопределить имя сервера, используемое при *проверке* сертификата uwsgi HTTPS-сервера, а также для *передачи его через SNI* при установлении соединения с suwsgi-сервером.

По умолчанию используется имя хоста из URL'а, заданного директивой *uwsgi\_pass*.

## uwsgi\_ssl\_password\_file

|                  |                                       |
|------------------|---------------------------------------|
| <i>Синтаксис</i> | uwsgi_ssl_password_file <i>файл</i> ; |
| По умолчанию     | —                                     |
| <i>Контекст</i>  | http, server, location                |

Задаёт файл с паролями от *секретных ключей*, где каждый пароль указан на отдельной строке. Пароли применяются по очереди в момент загрузки ключа.

## uwsgi\_ssl\_protocols

|                  |                                                                            |
|------------------|----------------------------------------------------------------------------|
| <i>Синтаксис</i> | uwsgi_ssl_protocols [SSLv2] [SSLv3] [TLSv1] [TLSv1.1] [TLSv1.2] [TLSv1.3]; |
| По умолчанию     | uwsgi_ssl_protocols TLSv1.2 TLSv1.3;                                       |
| <i>Контекст</i>  | http, server, location                                                     |

Изменено в версии 1.2.0: Параметр TLSv1.3 добавлен к используемым по умолчанию.

Разрешает указанные протоколы для запросов к suwsgi-серверу.

## uwsgi\_ssl\_server\_name

|                  |                                 |
|------------------|---------------------------------|
| <i>Синтаксис</i> | uwsgi_ssl_server_name on   off; |
| По умолчанию     | uwsgi_ssl_server_name off;      |
| <i>Контекст</i>  | http, server, location          |

Разрешает или запрещает передачу имени сервера, заданного директивой *uwsgi\_ssl\_name*, через расширение [Server Name Indication](#) протокола TLS (SNI, RFC 6066) при установлении соединения с защищенным uwsgi-сервером.

## uwsgi\_ssl\_session\_reuse

|                  |                                   |
|------------------|-----------------------------------|
| <i>Синтаксис</i> | uwsgi_ssl_session_reuse on   off; |
| По умолчанию     | uwsgi_ssl_session_reuse on;       |
| <i>Контекст</i>  | http, server, location            |

Определяет, использовать ли повторно SSL-сессии при работе с uwsgi-сервером. Если в логах появляются ошибки "*SSL3\_GET\_FINISHED:digest check failed*", то можно попробовать выключить повторное использование сессий.

## uwsgi\_ssl\_trusted\_certificate

|                  |                                             |
|------------------|---------------------------------------------|
| <i>Синтаксис</i> | uwsgi_ssl_trusted_certificate <i>файл</i> ; |
| По умолчанию     | —                                           |
| <i>Контекст</i>  | http, server, location                      |

Задаёт файл с доверенными сертификатами СА в формате PEM, используемыми при *проверке* сертификата uwsgi HTTPS-сервера.

## uwsgi\_ssl\_verify

|                  |                            |
|------------------|----------------------------|
| <i>Синтаксис</i> | uwsgi_ssl_verify on   off; |
| По умолчанию     | uwsgi_ssl_verify off;      |
| <i>Контекст</i>  | http, server, location     |

Разрешает или запрещает проверку сертификата uwsgi-сервера.



## uwsgi\_ssl\_verify\_depth

|                  |                                            |
|------------------|--------------------------------------------|
| <i>Синтаксис</i> | <code>uwsgi_ssl_verify_depth число;</code> |
| По умолчанию     | <code>uwsgi_ssl_verify_depth 1;</code>     |
| <i>Контекст</i>  | <code>http, server, location</code>        |

Устанавливает глубину проверки в цепочке сертификатов suwsgi-сервера.

## uwsgi\_store

|                  |                                             |
|------------------|---------------------------------------------|
| <i>Синтаксис</i> | <code>uwsgi_store on   off   строка;</code> |
| По умолчанию     | <code>uwsgi_store off;</code>               |
| <i>Контекст</i>  | <code>http, server, location</code>         |

Разрешает сохранение на диск файлов.

|                  |                                                                                               |
|------------------|-----------------------------------------------------------------------------------------------|
| <code>on</code>  | сохраняет файлы в соответствии с путями, указанными в директивах <i>alias</i> или <i>root</i> |
| <code>off</code> | запрещает сохранение файлов                                                                   |

Имя файла можно задать явно с помощью строки с переменными:

```
uwsgi_store /data/www$original_uri;
```

Время изменения файлов выставляется согласно полученному полю **Last-Modified** в заголовке ответа. Ответ сначала записывается во временный файл, а потом этот файл переименовывается. Временный файл и постоянное место хранения ответа могут располагаться на разных файловых системах. Однако нужно учитывать, что в этом случае вместо дешевой операции переименовывания в пределах одной файловой системы файл копируется с одной файловой системы на другую. Поэтому лучше, если сохраняемые файлы будут находиться на той же файловой системе, что и каталог с временными файлами, задаваемый директивой *uwsgi\_temp\_path* для данного *location*.

Директиву можно использовать для создания локальных копий статических неизменяемых файлов:

```
location /images/ {
 root /data/www;
 error_page 404 = /fetch$uri;
}

location /fetch/ {
 internal;

 uwsgi_pass backend:9000;
 ...

 uwsgi_store on;
 uwsgi_store_access user:rw group:rw all:r;
 uwsgi_temp_path /data/temp;

 alias /data/www/;
}
```

## uwsgi\_store\_access

|                  |                                                                |
|------------------|----------------------------------------------------------------|
| <i>Синтаксис</i> | <code>uwsgi_store_access <i>пользователи:права</i> ...;</code> |
| По умолчанию     | <code>uwsgi_store_access user:rw;</code>                       |
| <i>Контекст</i>  | http, server, location                                         |

Задаёт права доступа для создаваемых файлов и каталогов, например,

```
uwsgi_store_access user:rw group:rw all:r;
```

Если заданы какие-либо права для *group* или *all*, то права для *user* указывать необязательно:

```
uwsgi_store_access group:rw all:r;
```

## uwsgi\_temp\_file\_write\_size

|                  |                                                        |
|------------------|--------------------------------------------------------|
| <i>Синтаксис</i> | <code>uwsgi_temp_file_write_size <i>размер</i>;</code> |
| По умолчанию     | <code>uwsgi_temp_file_write_size 8k 16k;</code>        |
| <i>Контекст</i>  | http, server, location                                 |

Ограничивает размер данных, сбрасываемых во временный файл за один раз, при включенной буферизации ответов uwsgi-сервера во временные файлы. По умолчанию размер ограничен двумя буферами, заданными директивами *uwsgi\_buffer\_size* и *uwsgi\_buffers*. Максимальный размер временного файла задается директивой *uwsgi\_max\_temp\_file\_size*.

## uwsgi\_temp\_path

|                  |                                                                                                                  |
|------------------|------------------------------------------------------------------------------------------------------------------|
| <i>Синтаксис</i> | <code>uwsgi_temp_path <i>путь</i> [<i>уровень1</i> [<i>уровень2</i> [<i>уровень3</i>]]]`;</code>                 |
| По умолчанию     | <code>uwsgi_temp_path uwsgi_temp;</code> (путь зависит от параметра сборки <code>--http-uwsgi-temp-path</code> ) |
| <i>Контекст</i>  | http, server, location                                                                                           |

Задаёт имя каталога для хранения временных файлов с данными, полученными от uwsgi-серверов. В каталоге может использоваться иерархия подкаталогов до трех уровней. Например, при такой конфигурации

```
uwsgi_temp_path /spool/angie/uwsgi_temp 1 2;
```

временный файл будет следующего вида:

```
/spool/angie/uwsgi_temp/7/45/00000123457
```

См. также параметр *use\_temp\_path* директивы *uwsgi\_cache\_path*.

## HTTP/2

Обеспечивает поддержку HTTP/2.

При сборке из исходного кода модуль не собирается по умолчанию; его необходимо включить с помощью параметра сборки `--with-http_v2_module`. В пакетах и образах из наших репозиториях модуль включен в сборку.

### Пример конфигурации

```
server {
 listen 443 ssl;

 http2 on;

 ssl_certificate server.crt;
 ssl_certificate_key server.key;
}
```

#### Примечание

Чтобы принимать HTTP/2-соединения по TLS, необходимо наличие поддержки расширения "Application-Layer Protocol Negotiation" (ALPN) протокола TLS, появившейся в [OpenSSL](#) версии 1.0.2.

Если директива `ssl_prefer_server_ciphers` установлена в значение "on", *шифры* должны быть настроены таким образом, чтобы соответствовать черному списку [RFC 9113, Appendix A](#) а также поддерживаться клиентами.

## Директивы

### http2

|                  |                              |
|------------------|------------------------------|
| <i>Синтаксис</i> | <code>http2 on   off;</code> |
| По умолчанию     | <code>http2 off;</code>      |
| <i>Контекст</i>  | <code>http, server</code>    |

Разрешает протокол HTTP/2.

### http2\_body\_preload\_size

|                  |                                              |
|------------------|----------------------------------------------|
| <i>Синтаксис</i> | <code>http2_body_preload_size размер;</code> |
| По умолчанию     | —                                            |
| <i>Контекст</i>  | <code>http, server</code>                    |

Задаёт размер буфера для каждого запроса, в который может сохраняться тело запроса до того, как оно начнет обрабатываться.

## http2\_chunk\_size

|                  |                                       |
|------------------|---------------------------------------|
| <i>Синтаксис</i> | <code>http2_chunk_size размер;</code> |
| По умолчанию     | <code>http2_chunk_size 8k;</code>     |
| <i>Контекст</i>  | <code>http, server, location</code>   |

Задаёт максимальный размер частей, на которое будет разделяться тело ответа. Слишком маленькое значение может привести к росту накладных расходов. Слишком большое значение может негативно сказаться на приоритизации из-за блокировки очереди.

## http2\_max\_concurrent\_pushes

Устарело, начиная с версии 1.2.0.

|                  |                                                 |
|------------------|-------------------------------------------------|
| <i>Синтаксис</i> | <code>http2_max_concurrent_pushes число;</code> |
| По умолчанию     | <code>http2_max_concurrent_pushes 10;</code>    |
| <i>Контекст</i>  | <code>http, server</code>                       |

Ограничивает максимальное число параллельных *push*-запросов в соединении.

## http2\_max\_concurrent\_streams

|                  |                                                  |
|------------------|--------------------------------------------------|
| <i>Синтаксис</i> | <code>http2_max_concurrent_streams число;</code> |
| По умолчанию     | <code>http2_max_concurrent_streams 128;</code>   |
| <i>Контекст</i>  | <code>http, server</code>                        |

Задаёт максимальное число параллельных HTTP/2-потоков в соединении.

## http2\_push

Устарело, начиная с версии 1.2.0.

|                  |                                     |
|------------------|-------------------------------------|
| <i>Синтаксис</i> | <code>http2_push uri   off;</code>  |
| По умолчанию     | <code>http2_push off;</code>        |
| <i>Контекст</i>  | <code>http, server, location</code> |

Заблаговременно отправляет (*push*) запрос к заданному *uri* вместе с ответом на оригинальный запрос. Будут обработаны только относительные URI с абсолютными путями, например:

```
http2_push /static/css/main.css;
```

В значении *uri* допустимо использование переменных.

На одном уровне конфигурации можно указать несколько *http2\_push* директив. Параметр *off* отменяет действие унаследованных с предыдущего уровня конфигурации директив *http2\_push*.

## http2\_push\_preload

Устарело, начиная с версии 1.2.0.

|                  |                                           |
|------------------|-------------------------------------------|
| <i>Синтаксис</i> | <code>http2_push_preload on   off;</code> |
| По умолчанию     | <code>http2_push_preload off;</code>      |
| <i>Контекст</i>  | <code>http, server, location</code>       |

Разрешает автоматическое преобразование `preload links`, указанных в полях "Link" заголовка ответа, в `push`-запросы.

## http2\_recv\_buffer\_size

|                  |                                             |
|------------------|---------------------------------------------|
| <i>Синтаксис</i> | <code>http2_recv_buffer_size размер;</code> |
| По умолчанию     | <code>http2_recv_buffer_size 256k;</code>   |
| <i>Контекст</i>  | <code>http</code>                           |

Задаёт размер входного буфера для *рабочего процесса*.

## Встроенные переменные

Модуль `http_v2` поддерживает следующие встроенные переменные:

`$http2`

согласованный идентификатор протокола:

|                  |                                      |
|------------------|--------------------------------------|
| <code>h2</code>  | для HTTP/2 через TLS                 |
| <code>h2c</code> | для HTTP/2 через незашифрованный TCP |
| <code>" "</code> | пустая строка для остальных случаев  |

## HTTP/3

Обеспечивает поддержку протокола **HTTP/3** для соединений с клиентами, а также для соединений с проксируемыми серверами, настраиваемых при помощи следующих директив из модуля *Proxy*:

- `proxy_http3_hq`
- `proxy_http3_max_concurrent_streams`
- `proxy_http3_max_table_capacity`
- `proxy_http3_stream_buffer_size`
- `proxy_http_version`
- `proxy_pass`
- `proxy_quic_active_connection_id_limit`
- `proxy_quic_gso`
- `proxy_quic_host_key`

При сборке из исходного кода модуль не собирается по умолчанию; его необходимо включить с помощью параметра сборки `--with-http_v3_module`. В пакетах и образах из наших репозиториев модуль включен в сборку.

### Пример конфигурации

```
http {
 log_format quic '$remote_addr - $remote_user [$time_local] '
 '"$request" $status $body_bytes_sent '
 '"$http_referer" "$http_user_agent" "$http3"';

 access_log logs/access.log quic;

 server {
 # для лучшей совместимости рекомендуется
 # использовать одинаковый порт для http/3 и https
 listen 8443 quic reuseport;
 listen 8443 ssl;

 ssl_certificate certs/example.com.crt;
 ssl_certificate_key certs/example.com.key;

 location / {
 # используется для объявления о поддержке http/3
 add_header Alt-Svc 'h3=":8443"; ma=86400';
 }
 }
}
```

#### Примечание

Чтобы принимать HTTP/3-соединения по TLS, необходимо наличие поддержки протокола TLSv1.3, появившейся в [OpenSSL](#) версии 1.1.1.

Кроме того, опцию `reuseport` можно указать только в одной из директив `listen ... quic` на сервере. Остальные директивы `listen ... quic` должны быть указаны без нее.

### Директивы

#### http3

|                  |                              |
|------------------|------------------------------|
| <i>Синтаксис</i> | <code>http3 on   off;</code> |
| По умолчанию     | <code>http3 on;</code>       |
| <i>Контекст</i>  | <code>http, server</code>    |

Разрешает согласование протокола HTTP/3.

## http3\_hq

|                  |                    |
|------------------|--------------------|
| <i>Синтаксис</i> | http3_hq on   off; |
| По умолчанию     | http3_hq off;      |
| <i>Контекст</i>  | http, server       |

Разрешает согласование протокола HTTP/0.9, используемого в функциональных тестах QUIC.

### Предупреждение

Включайте этот режим только для запуска специализированных тестов, которым в явной форме необходим данный режим.

## http3\_max\_concurrent\_streams

|                  |                                     |
|------------------|-------------------------------------|
| <i>Синтаксис</i> | http3_max_concurrent_streams число; |
| По умолчанию     | http3_max_concurrent_streams 128;   |
| <i>Контекст</i>  | http, server                        |

Инициализирует настройки HTTP/3 и QUIC, а также задает максимальное число параллельных HTTP/3-потоков в соединении.

## http3\_max\_table\_capacity

|                       |                                 |
|-----------------------|---------------------------------|
| <i>Синтаксис</i>      | http3_max_table_capacity число; |
| Значение по умолчанию | http3_max_table_capacity 4096;  |
| <i>Контекст</i>       | http, server                    |

Определяет емкость динамической таблицы для серверных соединений.

### Примечание

Похожая директива `proxy_http3_max_table_capacity` задает это значение для прокси-соединений. Чтобы избежать ошибок, использование динамической таблицы отключается при включенном кэшировании в режиме проксирования.

## http3\_stream\_buffer\_size

|                  |                                          |
|------------------|------------------------------------------|
| <i>Синтаксис</i> | http3_stream_buffer_size <i>размер</i> ; |
| По умолчанию     | http3_stream_buffer_size 64k;            |
| <i>Контекст</i>  | http, server                             |

Задаёт *размер* буфера, используемого для чтения и записи QUIC-потоков.

## quic\_active\_connection\_id\_limit

|                  |                                                |
|------------------|------------------------------------------------|
| <i>Синтаксис</i> | quic_active_connection_id_limit <i>число</i> ; |
| По умолчанию     | quic_active_connection_id_limit 2;             |
| <i>Контекст</i>  | http, server                                   |

Устанавливает значение транспортного параметра QUIC `active_connection_id_limit`. Это максимальное значение ID соединений, возможное для хранения на сервере.

## quic\_bpf

|                  |                    |
|------------------|--------------------|
| <i>Синтаксис</i> | quic_bpf on   off; |
| По умолчанию     | quic_bpf off;      |
| <i>Контекст</i>  | main               |

Разрешает маршрутизацию пакетов QUIC при помощи **eBPF**. Если маршрутизация включена, то обеспечивается поддержка миграции QUIC-соединений.

### Примечание

Директива поддерживается только на Linux 5.7+.

## quic\_gso

|                  |                    |
|------------------|--------------------|
| <i>Синтаксис</i> | quic_gso on   off; |
| По умолчанию     | quic_gso off;      |
| <i>Контекст</i>  | http, server       |

Разрешает отправку оптимизированного пакетного режима при помощи segmentation offloading.

### Примечание

Оптимизированная отправка поддерживается только на Linux с поддержкой UDP\_SEGMENT.



## quic\_host\_key

|                  |                             |
|------------------|-----------------------------|
| <i>Синтаксис</i> | quic_host_key <i>файл</i> ; |
| По умолчанию     | —                           |
| <i>Контекст</i>  | http, server                |

Задаёт *файл* с секретным ключом, применяемым при шифровании stateless reset и address validation токенов. По умолчанию создается случайный ключ при каждой перезагрузке. Токены, созданные при помощи старых ключей, не принимаются.

## quic\_retry

|                  |                      |
|------------------|----------------------|
| <i>Синтаксис</i> | quic_retry on   off; |
| По умолчанию     | quic_retry off;      |
| <i>Контекст</i>  | http, server         |

Разрешает функциональность [QUIC Address Validation](#), в том числе отправку нового токена в *Retry*-пакете или *NEW\_TOKEN* frame и валидацию токена, полученного в *Initial*-пакете.

## Встроенные переменные

Модуль *http\_v3* поддерживает следующие встроенные переменные:

`$http3`

согласованный идентификатор протокола:

|    |                                     |
|----|-------------------------------------|
| h3 | для HTTP/3-соединений               |
| hq | для hq-соединений                   |
| "" | пустая строка для остальных случаев |

`$quic_connection`

порядковый номер QUIC-соединения

## Потоковые модули

|                       |                                                                                                                                               |
|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Stream</i>         | Основная функциональность потокового сервера для балансировки протоколов TCP и UDP на уровне L4.                                              |
| <i>Access</i>         | Контроль доступа на основе IP-адресов и диапазонов CIDR.                                                                                      |
| <i>ACME</i>           | Автоматическое получение и обновление SSL-сертификатов по протоколу ACME для потоковых серверов.                                              |
| <i>Geo</i>            | Преобразование IP-адресов в заданные значения переменных.                                                                                     |
| <i>Limit Conn</i>     | Ограничение числа одновременных соединений для защиты от перегрузки.                                                                          |
| <i>Log</i>            | Настройка журнала сессий для отслеживания обращений к ресурсам с целью мониторинга и анализа.                                                 |
| <i>Map</i>            | Преобразование переменных на основе predetermined пар "ключ-значение".                                                                        |
| <i>MQTT Preread</i>   | Чтение идентификатора клиента и имени пользователя из соединения по протоколу MQTT до момента принятия решения о балансировке.                |
| <i>Pass</i>           | Передача принятых соединений напрямую в настроенный слушающий сокет.                                                                          |
| <i>Proxy</i>          | Настройка проксирования к другим серверам.                                                                                                    |
| <i>RDP Preread</i>    | Чтение cookie из соединения по протоколу RDP до момента принятия решения о балансировке.                                                      |
| <i>RealIP</i>         | Определение адреса и порта клиента при работе за другим прокси-сервером.                                                                      |
| <i>Return</i>         | Отправка в ответ клиенту при его подключении заданного значения без дальнейшего проксирования.                                                |
| <i>Set</i>            | Установка заданных значений переменных.                                                                                                       |
| <i>Split Clients</i>  | Создание переменных для А/В-тестирования, канареечных релизов, шардинга и других сценариев, требующих разделения по пропорциональным группам. |
| <i>SSL</i>            | Терминация протоколов SSL/TLS и DTLS.                                                                                                         |
| <i>SSL Preread</i>    | Извлечение информации из сообщения ClientHello без терминации SSL/TLS и до момента принятия решения о балансировке.                           |
| <i>Upstream</i>       | Настройка групп проксируемых серверов для балансировки нагрузки.                                                                              |
| <i>Upstream Probe</i> | Настройка активных проверок работоспособности для групп проксируемых серверов.                                                                |

## Потоковый модуль

### Access

Модуль позволяет ограничить доступ для определенных адресов клиентов.

### Пример конфигурации

```
server {
 ...
 deny 192.168.1.1;
 allow 192.168.1.0/24;
 allow 10.1.1.0/16;
 allow 2001:0db8::/32;
 deny all;
}
```

Правила проверяются в порядке их записи до первого соответствия. В данном примере доступ разрешен только для IPv4-сетей 10.1.1.0/16 и 192.168.1.0/24, кроме адреса 192.168.1.1, и для IPv6-сети 2001:0db8::/32.

## Директивы

### allow

|                  |                                                |
|------------------|------------------------------------------------|
| <i>Синтаксис</i> | <code>allow адрес   CIDR   unix:   all;</code> |
| По умолчанию     | —                                              |
| <i>Контекст</i>  | stream, server                                 |

Разрешает доступ для указанной сети или адреса. Если указано специальное значение `unix:`, разрешает доступ для всех UNIX-сокетов.

### deny

|                  |                                               |
|------------------|-----------------------------------------------|
| <i>Синтаксис</i> | <code>deny адрес   CIDR   unix:   all;</code> |
| По умолчанию     | —                                             |
| <i>Контекст</i>  | stream, server                                |

Запрещает доступ для указанной сети или адреса. Если указано специальное значение `unix:`, запрещает доступ для всех UNIX-сокетов.

## АСМЕ

Позволяет автоматически получать сертификаты с использованием протокола [АСМЕ](#) для серверов, определенных в контексте `stream`.

При сборке из исходного кода модуль не собирается по умолчанию; его необходимо включить с помощью параметра сборки `--with-stream_acme_module` (также требуется `--with-http_acme_module`). В пакетах и образах из наших репозиториях модуль включен в сборку.

### Примечание

Для корректной работы блок `stream` должен располагаться после блока `http`. Это связано с тем, что потоковый модуль использует определения клиентов, созданные при разборе HTTP-конфигурации.

## Пример конфигурации

Примеры конфигурации и инструкции по настройке см. в разделе [АСМЕ в потоковом модуле](#).

## Директивы

### acme

|                  |                        |
|------------------|------------------------|
| <i>Синтаксис</i> | <code>acme имя;</code> |
| По умолчанию     | —                      |
| <i>Контекст</i>  | <code>server</code>    |

Для всех доменов, указанных в директивах `server_name` во всех блоках `server`, которые ссылаются на *клиент ACME* из HTTP-модуля с именем *имя*, будет получен единый сертификат; если изменится конфигурация `server_name`, сертификат будет обновлен для учета изменений.

При каждом запуске Angie для всех доменов, у которых отсутствует действующий сертификат, запрашиваются новые сертификаты. Возможные причины включают истечение срока действия сертификатов, отсутствие файлов или невозможность прочитать их, а также изменения в настройках сертификатов.

#### Примечание

Сейчас домены, заданные через регулярные выражения, не поддерживаются и будут пропускаться.

Домены со звездочкой поддерживаются только в режиме `challenge=dns` в `acme_client`.

Эта директива может быть указана несколько раз для загрузки сертификатов разных типов, например RSA и ECDSA:

```
server {
 listen 12345 ssl;
 server_name example.com www.example.com;

 ssl_certificate $acme_cert_rsa;
 ssl_certificate_key $acme_cert_key_rsa;

 ssl_certificate $acme_cert_ecdsa;
 ssl_certificate_key $acme_cert_key_ecdsa;

 acme rsa;
 acme ecdsa;
}
```

## Встроенные переменные

`$acme_cert_<имя>`

Содержимое последнего файла сертификата (если он есть), полученного клиентом с этим *именем*.

`$acme_cert_key_<имя>`

Содержимое файла ключа сертификата, используемого клиентом с этим *именем*.

#### **Примечание**

Файл сертификата доступен, только если клиент АСМЕ получил хотя бы один сертификат, а вот файл ключа доступен сразу после запуска.

## Geo

Создает переменные, значения которых зависят от IP-адреса клиента.

### Пример конфигурации

```
geo $geo {
 default 0;

 127.0.0.1 2;
 192.168.1.0/24 1;
 10.1.0.0/16 1;

 ::1 2;
 2001:0db8::/32 1;
}
```

## Директивы

### geo

|                  |                                                               |
|------------------|---------------------------------------------------------------|
| <i>Синтаксис</i> | <code>geo [<i>\$адрес</i>] <i>\$переменная</i> { ... }</code> |
| По умолчанию     | —                                                             |
| <i>Контекст</i>  | stream                                                        |

Описывает для указанной переменной зависимость значения от IP-адреса клиента. По умолчанию адрес берется из переменной *\$remote\_addr*, но его также можно получить из другой переменной, например:

```
geo $arg_remote_addr $geo {
 ...;
}
```

#### **Примечание**

Поскольку переменные вычисляются только в момент использования, само по себе наличие даже большого числа объявлений переменных `geo` не влечет за собой никаких дополнительных расходов на обработку соединений.

Если значение переменной не представляет из себя правильный IP-адрес, то используется адрес "255.255.255.255".

Адреса задаются либо префиксами в формате CIDR (включая одиночные адреса), либо в виде диапазонов.

Также поддерживаются следующие специальные параметры:

|                |                                                                                                                                                                                                                                                                           |
|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>delete</b>  | Удаляет описанную сеть.                                                                                                                                                                                                                                                   |
| <b>default</b> | Значение переменной, если адрес клиента не соответствует ни одному из заданных адресов. При задании адресов в формате CIDR вместо <b>default</b> можно использовать 0.0.0.0/0 и ::/0. Если параметр <b>default</b> не указан, значением по умолчанию будет пустая строка. |
| <b>include</b> | Включает файл с адресами и значениями. Включений может быть несколько.                                                                                                                                                                                                    |
| <b>ranges</b>  | Указывает, что адреса задаются в виде диапазонов. Этот параметр должен быть первым. Для ускорения загрузки гео-базы нужно располагать адреса в порядке возрастания.                                                                                                       |

Пример:

```
geo $country {
 default ZZ;
 include conf/geo.conf;
 delete 127.0.0.0/16;

 127.0.0.0/24 US;
 127.0.0.1/32 RU;
 10.1.0.0/16 RU;
 192.168.1.0/24 UK;
}
```

В файле `conf/geo.conf` могут быть такие строки:

```
10.2.0.0/16 RU;
192.168.2.0/24 RU;
```

В качестве значения выбирается максимальное совпадение, например, для адреса 127.0.0.1 будет выбрано значение RU, а не US.

Пример описания диапазонов:

```
geo $country {
 ranges;
 default ZZ;
 127.0.0.0-127.0.0.0 US;
 127.0.0.1-127.0.0.1 RU;
 127.0.0.2-127.0.0.255 US;
 10.1.0.0-10.1.255.255 RU;
 192.168.1.0-192.168.1.255 UK;
}
```

## Limit Conn

Позволяет ограничить число соединений по заданному ключу, в частности, число соединений с одного IP-адреса.

### Пример конфигурации

```
stream {
 limit_conn_zone $binary_remote_addr zone=addr:10m;

 ...

 server {

 ...

 limit_conn addr 1;
 limit_conn_log_level error;
 }
}
```

## Директивы

### limit\_conn

|                  |                        |
|------------------|------------------------|
| <i>Синтаксис</i> | limit_conn зона число; |
| По умолчанию     | —                      |
| <i>Контекст</i>  | stream, server         |

Задаёт зону разделяемой памяти и максимально допустимое число соединений для одного значения ключа. При превышении этого числа сервер закроет соединение. Например, директивы

```
limit_conn_zone $binary_remote_addr zone=addr:10m;

server {
 ...
 limit_conn addr 1;
}
```

разрешают одновременно обрабатывать не более одного соединения с одного IP-адреса.

Допустимо одновременное указание нескольких директив `limit_conn`, при этом будет срабатывать любое из ограничений.

Директивы наследуются с предыдущего уровня конфигурации при условии, что на данном уровне не описаны свои директивы `limit_conn`.

## limit\_conn\_dry\_run

|                  |                              |
|------------------|------------------------------|
| <i>Синтаксис</i> | limit_conn_dry_run on   off; |
| По умолчанию     | limit_conn_dry_run off;      |
| <i>Контекст</i>  | stream, server               |

Включает режим пробного запуска. В данном режиме число соединений не ограничивается, однако в *зоне разделяемой памяти* текущее число избыточных соединений учитывается как обычно.

## limit\_conn\_log\_level

|                  |                                                    |
|------------------|----------------------------------------------------|
| <i>Синтаксис</i> | limit_conn_log_level info   notice   warn   error; |
| По умолчанию     | limit_conn_log_level error;                        |
| <i>Контекст</i>  | stream, server                                     |

Задаёт желаемый уровень записи в лог случаев ограничения числа соединений.

## limit\_conn\_zone

|                  |                                                             |
|------------------|-------------------------------------------------------------|
| <i>Синтаксис</i> | limit_conn_zone <i>ключ</i> zone = <i>название:размер</i> ; |
| По умолчанию     | —                                                           |
| <i>Контекст</i>  | stream                                                      |

Задаёт параметры зоны разделяемой памяти, которая хранит состояние для разных значений ключа. Состояние в частности содержит текущее число соединений. В качестве ключа может использоваться текст, переменные и их комбинации. Запросы с пустым значением ключа не учитываются.

Пример использования:

```
limit_conn_zone $binary_remote_addr zone=addr:10m;
```

Здесь в качестве ключа используется IP-адрес клиента, задаваемый переменной \$binary\_remote\_addr.

Длина значения \$binary\_remote\_addr равна 4 байтам для IPv4-адресов или 16 байтам для IPv6-адресов. При этом размер состояния всегда равен 32 или 64 байтам на 32-битных платформах и 64 байтам на 64-битных. В зоне размером 1 мегабайт может разместиться около 32 тысяч состояний размером 32 байта или 16 тысяч состояний размером 64 байта. При переполнении зоны сервер закрывает соединение.



## Встроенные переменные

`$limit_conn_status`

хранит результат ограничения числа соединений: `PASSED`, `REJECTED` или `REJECTED_DRY_RUN`

## Log

Модуль записывает логи запросов в указанном формате.

## Пример конфигурации

```
log_format basic '$remote_addr [$time_local] '
 '$protocol $status $bytes_sent $bytes_received '
 '$session_time';

access_log /spool/logs/angie-access.log basic buffer=32k;
```

## Директивы

### `access_log`

|                  |                                                                                                                                    |
|------------------|------------------------------------------------------------------------------------------------------------------------------------|
| <i>Синтаксис</i> | <code>access_log путь [формат [buffer=размер] [gzip[=степень]] [flush=время] [if=условие]];</code><br><code>access_log off;</code> |
| По умолчанию     | <code>access_log off;</code>                                                                                                       |
| <i>Контекст</i>  | <code>stream, server</code>                                                                                                        |

Задаёт *путь*, *формат* и настройки буферизованной записи в лог. На одном уровне конфигурации может использоваться несколько логов. Запись в *syslog* настраивается указанием префикса *"syslog:"* в первом параметре. Специальное значение *off* отменяет все директивы *access\_log* для текущего уровня.

Если задан размер буфера с помощью параметра *buffer* или указан параметр *gzip*, то запись будет буферизованной.

### Предупреждение

Размер буфера должен быть не больше размера атомарной записи в дисковый файл. Для FreeBSD этот размер неограничен.

При включенной буферизации данные записываются в файл:

- если очередная строка лога не помещается в буфер;
- если данные в буфере находятся дольше интервала времени, заданного параметром *flush*;
- при переоткрытии лог-файла или завершении рабочего процесса.

Если задан параметр *gzip*, то буфер будет сжиматься перед записью в файл. Степень сжатия может быть задана в диапазоне от *1* (быстрее, но хуже сжатие) до *9* (медленнее, но лучше сжатие). По умолчанию используются буфер размером *64K* байт и степень сжатия *1*. Данные сжимаются

атомарными блоками, и в любой момент времени лог-файл может быть распакован или прочитан с помощью утилиты `"zcat"`.

Пример:

```
access_log /path/to/log.gz basic gzip flush=5m;
```

#### Примечание

Для поддержки gzip-сжатия логов Angie должен быть собран с библиотекой `zlib`.

В пути файла можно использовать переменные, но такие логи имеют некоторые ограничения:

- *пользователь*, с правами которого работают рабочие процессы, должен иметь права на создание файлов в каталоге с такимилогами;
- не работает буферизация;
- файл открывается для каждой записи в лог и сразу же после записи закрывается. Следует однако иметь в виду, что поскольку дескрипторы часто используемых файлов могут храниться в кэше, то при ротации логов в течение времени, заданного параметром `valid` директивы `open_log_file_cache`, запись может продолжаться в старый файл.

Параметр `if` включает условную запись в лог. Сессия не будет записываться в лог, если результатом вычисления условия является `"0"` или пустая строка.

## log\_format

|                  |                                                                        |
|------------------|------------------------------------------------------------------------|
| <i>Синтаксис</i> | <code>log_format имя [escape=default   json   none] строка ...;</code> |
| По умолчанию     | —                                                                      |
| <i>Контекст</i>  | stream                                                                 |

Задаёт формат лога, например:

```
log_format proxy '$remote_addr [$time_local] '
 '$protocol $status $bytes_sent $bytes_received '
 '$session_time "$upstream_addr" '
 '"$upstream_bytes_sent" "$upstream_bytes_received" "$upstream_
↪connect_time";
```

Параметр `escape` позволяет задать экранирование символов `json` или `default` в переменных, по умолчанию используется `default`. Значение `none` отключает экранирование символов.

При использовании `default` символы `"", \"`, а также символы со значениями меньше 32 или больше 126 экранируются как `"\xXX"`. Если значение переменной не найдено, то в качестве значения в лог будет записываться дефис `"-"`.

При использовании `json` экранируются все символы, недопустимые в JSON строках: символы `"", \"` и `\"` экранируются как `"\"` и `\"`, символы со значениями меньше 32 экранируются как `"\n", \"r", \"t", \"b", \"f` или `"u00XX"`.

## open\_log\_file\_cache

|                  |                                                                                                                                           |
|------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Синтаксис</i> | <code>open_log_file_cache max=N [inactive=время] [min_uses=N] [valid=время];</code><br>По умолчанию <code>open_log_file_cache off;</code> |
| <i>Контекст</i>  | http, server, location                                                                                                                    |

Задаёт кэш, в котором хранятся дескрипторы файлов часто используемых логов, имена которых заданы с использованием переменных. Параметры:

|                 |                                                                                                                                                                                            |
|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>max</b>      | Задаёт максимальное число дескрипторов в кэше; при переполнении кэша наименее востребованные (LRU) дескрипторы закрываются.                                                                |
| <b>inactive</b> | Задаёт время, после которого кэшированный дескриптор закрывается, если к нему не было обращений в течение этого времени; по умолчанию — 10 секунд.                                         |
| <b>min_uses</b> | Задаёт минимальное число использований файла в течение времени, заданного параметром <b>inactive</b> , после которого дескриптор файла будет оставаться открытым в кэше; по умолчанию — 1. |
| <b>valid</b>    | Указывает, через какое время нужно проверять, что файл ещё существует под тем же именем; по умолчанию — 60 секунд.                                                                         |
| <b>off</b>      | Запрещает кэширование.                                                                                                                                                                     |

Пример использования:

```
open_log_file_cache max=1000 inactive=20s valid=1m min_uses=2;
```

## Map

Создаёт переменные, значения которых зависят от значений других переменных.

### Пример конфигурации

```
map $remote_addr $limit {
 127.0.0.1 "";
 default $binary_remote_addr;
}

limit_conn_zone $limit zone=addr:10m;
limit_conn addr 1;
```

## Директивы

### map

|                  |                                               |
|------------------|-----------------------------------------------|
| <i>Синтаксис</i> | <code>map строка \$переменная { ... };</code> |
| По умолчанию     | —                                             |
| <i>Контекст</i>  | stream                                        |

Создает новую переменную. Ее значение зависит от первого параметра, заданного строкой с переменными, например:

```
set $var1 "foo";
set $var2 "bar";

map $var1$var2 $new_variable {
 default "foobar_value";
}
```

Здесь переменная `$new_variable` будет иметь значение, составленное из двух переменных `$var1` и `$var2`, или значение по умолчанию, если эти переменные не определены.

#### **Примечание**

Поскольку переменные вычисляются только в момент использования, само по себе наличие даже большого числа объявлений переменных `map` не влечет за собой никаких дополнительных расходов на обработку запросов.

Параметры внутри блока `map` задают соответствие между исходными и результирующими значениями.

Исходные значения задаются строками или регулярными выражениями.

Строки проверяются без учета регистра.

Перед регулярным выражением ставится символ `~`, если при сравнении следует учитывать регистр символов, либо символы `~*`, если регистр символов учитывать не нужно. Регулярное выражение может содержать именованные и позиционные группы захвата, которые могут затем использоваться в других директивах совместно с результирующей переменной.

Если исходное значение совпадает с именем одного из специальных параметров, описанных ниже, перед ним следует поставить символ `\`.

В качестве результирующего значения можно указать текст, переменную и их комбинации.

Также поддерживаются следующие специальные параметры:

|                                      |                                                                                                                                                                                                           |
|--------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>default</code> <i>значение</i> | задает результирующее значение, если исходное значение не совпадает ни с одним из перечисленных. Если параметр <code>default</code> не указан, результирующим значением по умолчанию будет пустая строка. |
| <code>hostnames</code>               | указывает, что в качестве исходных значений можно использовать маску для первой или последней части имени хоста. Этот параметр следует указывать перед списком значений.                                  |

Например,

```
*.example.com 1;
example.* 1;
```

Вместо двух записей

```
example.com 1;
*.example.com 1;
```

можно использовать одну:

```
.example.com 1;
```

|                                  |                                                              |
|----------------------------------|--------------------------------------------------------------|
| <code>include <i>файл</i></code> | включает файл со значениями. Включений может быть несколько. |
| <code>volatile</code>            | указывает, что переменная не кэшируется.                     |

Если исходному значению соответствует несколько из указанных вариантов, например, одновременно подходят и маска, и регулярное выражение, будет выбран первый подходящий вариант в следующем порядке приоритета:

1. Строковое значение без маски.
2. Самое длинное строковое значение с маской в начале, например `*.example.com`.
3. Самое длинное строковое значение с маской в конце, например `mail.*`.
4. Первое подходящее регулярное выражение (в порядке следования в конфигурационном файле).
5. Значение по умолчанию (`default`).

### map\_hash\_bucket\_size

|                  |                                                  |
|------------------|--------------------------------------------------|
| <i>Синтаксис</i> | <code>map_hash_bucket_size <i>размер</i>;</code> |
| По умолчанию     | <code>map_hash_bucket_size 32 64 128;</code>     |
| <i>Контекст</i>  | stream                                           |

Задаёт размер корзины в хэш-таблицах для переменных *map*. Значение по умолчанию зависит от размера строки кэша процессора. Подробнее настройка хэш-таблиц обсуждается *отдельно*.

### map\_hash\_max\_size

|                  |                                               |
|------------------|-----------------------------------------------|
| <i>Синтаксис</i> | <code>map_hash_max_size <i>размер</i>;</code> |
| По умолчанию     | <code>map_hash_max_size 2048;</code>          |
| <i>Контекст</i>  | stream                                        |

Задаёт максимальный размер хэш-таблиц для переменных *map*. Подробнее настройка хэш-таблиц обсуждается *отдельно*.

## MQTT Preread

Позволяет извлекать идентификатор клиента и имя пользователя из пакетов `CONNECT` протокола Message Queuing Telemetry Transport (MQTT) версий 3.1.1 и 5.0.

При сборке из исходного кода модуль необходимо включить с помощью параметра сборки `--with-stream-mqtt-preread-module`. В пакетах и образах из наших репозиториях модуль включен в сборку.

## Пример конфигурации

Выбор сервера в группе по идентификатору клиента:

```
stream {
 mqtt_preread on;

 upstream mqtt {
 hash $mqtt_preread_clientid;
 # ...
 }
}
```

## Директивы

### mqtt\_preread

|                  |                        |
|------------------|------------------------|
| <i>Синтаксис</i> | mqtt_preread on   off; |
| По умолчанию     | mqtt_preread off;      |
| <i>Контекст</i>  | stream, server         |

Управляет извлечением информации из пакета **CONNECT** на этапе *предварительного чтения*. Если параметр включен (**on**), то в контексте, где он задан, заполняются перечисленные ниже переменные.

### Встроенные переменные

Подробное описание семантики значений см. в спецификации протокола MQTT версий 3.1.1 и 5.0.

`$mqtt_preread_clientid`

Уникальный идентификатор клиента.

`$mqtt_preread_username`

Необязательное имя пользователя.

### Pass

Позволяет передавать принятое соединение напрямую на любой настроенный слушающий сокет в модуль HTTP, потоковый или почтовый модули.

Модуль допускает выборочную SSL-терминацию на основе SNI.

## Пример конфигурации

После того, как модуль `stream` завершит обработку SSL/TLS, он передает соединение в модуль `http`:

```
stream {
 server {
 listen 8000 default_server;
 ssl_preread on;
 # ...
 }

 server {
 listen 8000;
 server_name foo.example.com;
 pass 127.0.0.1:8001; # to HTTP
 }

 server {
 listen 8000;
 server_name bar.example.com;
 # ...
 }
}

http {
 server {
 listen 8001 ssl;
 # ...

 location / {
 root html;
 }
 }
}
```

## Директивы

### `pass`

|                  |                          |
|------------------|--------------------------|
| <i>Синтаксис</i> | <code>pass адрес;</code> |
| По умолчанию     | —                        |
| <i>Контекст</i>  | <code>server</code>      |

Эта директива задает адрес сервера, на который должно быть передано клиентское соединение. Адрес можно указать как IP-адрес и порт:

```
pass 127.0.0.1:12345;
```

Или как путь к UNIX-сокету:

```
pass unix:/tmp/stream.socket;
```

Также *адрес* можно задать с помощью переменных:

```
pass $upstream;
```

## Proxy

Позволяет проксировать потоки данных по TCP, UDP и UNIX-сокетах.

### Пример конфигурации

```
server {
 listen 127.0.0.1:12345;
 proxy_pass 127.0.0.1:8080;
}

server {
 listen 12345;
 proxy_connect_timeout 1s;
 proxy_timeout 1m;
 proxy_pass example.com:12345;
}

server {
 listen 53 udp reuseport;
 proxy_timeout 20s;
 proxy_pass dns.example.com:53;
}

server {
 listen [::1]:12345;
 proxy_pass unix:/tmp/stream.socket;
}
```

## Директивы

### proxy\_bind

|                  |                                                    |
|------------------|----------------------------------------------------|
| <i>Синтаксис</i> | <code>proxy_bind адрес [transparent]   off;</code> |
| По умолчанию     | —                                                  |
| <i>Контекст</i>  | stream, server                                     |

Задаёт локальный IP-адрес, который будет использоваться в исходящих соединениях с проксируемым сервером. В значении параметра допустимо использование переменных. Специальное значение `off` отменяет действие унаследованной с предыдущего уровня конфигурации директивы `proxy_bind`, позволяя системе самостоятельно выбирать локальный IP-адрес.



Параметр `transparent` позволяет задать нелокальный IP-адрес, который будет использоваться в исходящих соединениях с проксируемым сервером, например, реальный IP-адрес клиента:

```
proxy_bind $remote_addr transparent;
```

Для работы параметра обычно требуется запустить рабочие процессы Angie с привилегиями *супер-пользователя*. В Linux этого не требуется, так как если указан параметр *transparent*, то рабочие процессы наследуют capability *CAP\_NET\_RAW* из главного процесса.

#### **Примечание**

Необходимо настроить таблицу маршрутизации ядра для перехвата сетевого трафика с проксируемого сервера.

### **proxy\_buffer\_size**

|                  |                                        |
|------------------|----------------------------------------|
| <i>Синтаксис</i> | <code>proxy_buffer_size размер;</code> |
| По умолчанию     | <code>proxy_buffer_size 16k;</code>    |
| <i>Контекст</i>  | stream, server                         |

Задаёт размер буфера, в который будут читаться данные, получаемые от проксируемого сервера. Также задаёт размер буфера, в который будут читаться данные, получаемые от клиента.

### **proxy\_connect\_timeout**

|                  |                                           |
|------------------|-------------------------------------------|
| <i>Синтаксис</i> | <code>proxy_connect_timeout время;</code> |
| По умолчанию     | <code>proxy_connect_timeout 60s;</code>   |
| <i>Контекст</i>  | stream, server                            |

Задаёт таймаут для установления соединения с проксируемым сервером.

### **proxy\_connection\_drop**

|                  |                                                      |
|------------------|------------------------------------------------------|
| <i>Синтаксис</i> | <code>proxy_connection_drop время   on   off;</code> |
| По умолчанию     | <code>proxy_connection_drop off;</code>              |
| <i>Контекст</i>  | http, server, location                               |

Настраивает завершение всех соединений с проксируемым сервером, если он был удален из группы или помечен как постоянно недоступный в результате процесса *resolve* или команды *API DELETE*.

Сессия завершается, когда обрабатывается следующее событие чтения или записи для клиента или проксируемого сервера.

Установка *времени* включает *таймаут* до завершения сессии; при выборе значения *on* сессии завершаются немедленно.

## proxy\_download\_rate

|                  |                                       |
|------------------|---------------------------------------|
| <i>Синтаксис</i> | proxy_download_rate <i>скорость</i> ; |
| По умолчанию     | proxy_download_rate 0;                |
| <i>Контекст</i>  | stream, server                        |

Ограничивает скорость чтения данных от проксируемого сервера; **скорость** задается в байтах в секунду.

|   |                                |
|---|--------------------------------|
| 0 | отключает ограничение скорости |
|---|--------------------------------|

### Примечание

Ограничение устанавливается на соединение, поэтому, если Angie одновременно откроет два соединения к проксируемому серверу, суммарная скорость будет вдвое выше заданного ограничения.

В значении параметра можно использовать переменные. Это может быть полезно в случаях, когда скорость нужно ограничивать в зависимости от какого-либо условия:

```
map $slow $rate {
 1 4k;
 2 8k;
}

proxy_download_rate $rate
```

## proxy\_half\_close

|                  |                            |
|------------------|----------------------------|
| <i>Синтаксис</i> | proxy_half_close on   off; |
| По умолчанию     | proxy_half_close off;      |
| <i>Контекст</i>  | stream, server             |

Разрешает или запрещает независимое закрытие каждой из сторон проксируемого соединения TCP ("TCP half-close"). Если разрешено, то проксирование по TCP будет продолжаться, пока обе стороны не закроют соединение.

## proxy\_next\_upstream

|                  |                               |
|------------------|-------------------------------|
| <i>Синтаксис</i> | proxy_next_upstream on   off; |
| По умолчанию     | proxy_next_upstream on;       |
| <i>Контекст</i>  | stream, server                |

При невозможности установить соединение с проксируемым сервером определяет, будет ли клиентское соединение передано *следующему серверу*.

Передача соединения следующему серверу может быть ограничена по *количеству попыток* и по *времени*.

### proxy\_next\_upstream\_timeout

|                  |                                            |
|------------------|--------------------------------------------|
| <i>Синтаксис</i> | proxy_next_upstream_timeout <i>время</i> ; |
| По умолчанию     | proxy_next_upstream_timeout 0;             |
| <i>Контекст</i>  | stream, server                             |

Ограничивает время, в течение которого возможна передача запроса *следующему* серверу.

|   |                           |
|---|---------------------------|
| 0 | отключает это ограничение |
|---|---------------------------|

### proxy\_next\_upstream\_tries

|                  |                                          |
|------------------|------------------------------------------|
| <i>Синтаксис</i> | proxy_next_upstream_tries <i>число</i> ; |
| По умолчанию     | proxy_next_upstream_tries 0;             |
| <i>Контекст</i>  | stream, server                           |

Ограничивает число допустимых попыток для передачи запроса *следующему* серверу.

|   |                           |
|---|---------------------------|
| 0 | отключает это ограничение |
|---|---------------------------|

### proxy\_pass

|                  |                           |
|------------------|---------------------------|
| <i>Синтаксис</i> | proxy_pass <i>адрес</i> ; |
| По умолчанию     | —                         |
| <i>Контекст</i>  | server                    |

Задаёт адрес проксируемого сервера; **адрес** может быть указан в виде доменного имени или IP-адреса, за которым следует порт:

```
proxy_pass localhost:12345;
```

или в виде пути UNIX-сокета:

```
proxy_pass unix:/tmp/stream.socket;
```

Если доменному имени соответствует несколько адресов, то все они будут использоваться по очереди (round-robin). Кроме того, в качестве адреса можно указать *группу серверов*.

Адрес можно также задать с помощью переменных:

```
proxy_pass $upstream;
```

В этом случае имя сервера ищется среди описанных *групп серверов* и если не найдено, то определяется с помощью *resolver*'а.

## proxy\_protocol

|                  |                          |
|------------------|--------------------------|
| <i>Синтаксис</i> | proxy_protocol on   off; |
| По умолчанию     | proxy_protocol off;      |
| <i>Контекст</i>  | stream, server           |

Включает протокол **PROXY** для соединений с проксируемым сервером.

## proxy\_requests

|                  |                       |
|------------------|-----------------------|
| <i>Синтаксис</i> | proxy_requests число; |
| По умолчанию     | proxy_requests 0;     |
| <i>Контекст</i>  | stream, server        |

Задаёт число датаграмм, полученных от клиента, по достижении которого удаляется привязка между клиентом и существующей UDP-сессией. После получения указанного количества датаграмм следующая датаграмма, полученная от того же клиента, начинает новую сессию. Сессия завершится после отправки всех принятых датаграмм на проксируемый сервер и получения указанного количества *ответов* или после *таймаута*.

## proxy\_responses

|                  |                        |
|------------------|------------------------|
| <i>Синтаксис</i> | proxy_responses число; |
| По умолчанию     | —                      |
| <i>Контекст</i>  | stream, server         |

Задаёт количество датаграмм, ожидаемых от проксируемого сервера в ответ на датаграмму клиента в случае, если используется протокол **UDP**. Задаваемое число служит подсказкой для завершения сессии. По умолчанию количество датаграмм не ограничено.

Если указано нулевое значение, то ответ не ожидается. Однако если ответ получен и сессия еще не завершилась, то ответ будет обработан.

## proxy\_socket\_keepalive

|                  |                                  |
|------------------|----------------------------------|
| <i>Синтаксис</i> | proxy_socket_keepalive on   off; |
| По умолчанию     | proxy_socket_keepalive off;      |
| <i>Контекст</i>  | stream, server                   |

Конфигурирует поведение "TCP keepalive" для исходящих соединений к проксируемому серверу.

|    |                                                                   |
|----|-------------------------------------------------------------------|
| "" | По умолчанию для сокета действуют настройки операционной системы. |
| on | для сокета включается параметр <i>SO_KEEPALIVE</i>                |

## proxy\_ssl

|                  |                                  |
|------------------|----------------------------------|
| <i>Синтаксис</i> | <code>proxy_ssl on   off;</code> |
| По умолчанию     | <code>proxy_ssl off;</code>      |
| <i>Контекст</i>  | stream, server                   |

Включает протоколы SSL/TLS для соединений с проксируемым сервером.

## proxy\_ssl\_certificate

|                  |                                                 |
|------------------|-------------------------------------------------|
| <i>Синтаксис</i> | <code>proxy_ssl_certificate файл [файл];</code> |
| По умолчанию     | —                                               |
| <i>Контекст</i>  | stream, server                                  |

Задаёт файл с сертификатом в формате PEM для аутентификации на проксируемом сервере. В имени файла можно использовать переменные.

При включённом `proxy_ssl_ntls` директива принимает два аргумента вместо одного:

```
server {
 proxy_ssl_ntls on;

 proxy_ssl_certificate sign.crt enc.crt;
 proxy_ssl_certificate_key sign.key enc.key;

 proxy_ssl_ciphers "ECC-SM2-WITH-SM4-SM3:ECDHE-SM2-WITH-SM4-SM3:RSA";

 proxy_pass backend:12345;
}
```

## proxy\_ssl\_certificate\_key

|                  |                                                     |
|------------------|-----------------------------------------------------|
| <i>Синтаксис</i> | <code>proxy_ssl_certificate_key файл [файл];</code> |
| По умолчанию     | —                                                   |
| <i>Контекст</i>  | stream, server                                      |

Задаёт файл с секретным ключом в формате PEM для аутентификации на проксируемом сервере. В имени файла можно использовать переменные.

При включённом `proxy_ssl_ntls` директива принимает два аргумента вместо одного:

```
server {
 proxy_ssl_ntls on;

 proxy_ssl_certificate sign.crt enc.crt;
 proxy_ssl_certificate_key sign.key enc.key;

 proxy_ssl_ciphers "ECC-SM2-WITH-SM4-SM3:ECDHE-SM2-WITH-SM4-SM3:RSA";
}
```

```
proxy_pass backend:12345;
}
```

## proxy\_ssl\_ciphers

|              |                                  |
|--------------|----------------------------------|
| Синтаксис    | proxy_ssl_ciphers <i>шифры</i> ; |
| По умолчанию | proxy_ssl_ciphers DEFAULT;       |
| Контекст     | stream, server                   |

Описывает разрешенные шифры для запросов к проксируемому серверу. Шифры задаются в формате, поддерживаемом библиотекой OpenSSL.

Список шифров зависит от установленной версии OpenSSL. Полный список можно посмотреть с помощью команды `openssl ciphers`.

### ⚠ Предупреждение

Директива `proxy_ssl_ciphers` *не* настраивает шифры для TLS 1.3 при использовании OpenSSL. Для настройки шифров TLS 1.3 в OpenSSL используйте директиву `proxy_ssl_conf_command`, добавленную для расширенной конфигурации SSL.

- В LibreSSL шифры TLS 1.3 *можно* настраивать с помощью `proxy_ssl_ciphers`.
- В BoringSSL шифры TLS 1.3 настроить невозможно.

## proxy\_ssl\_conf\_command

|              |                                              |
|--------------|----------------------------------------------|
| Синтаксис    | proxy_ssl_conf_command <i>имя значение</i> ; |
| По умолчанию | —                                            |
| Контекст     | stream, server                               |

Задаёт произвольные конфигурационные *команды* OpenSSL при установлении соединения с проксируемым сервером.

### i Примечание

Директива поддерживается при использовании OpenSSL 1.0.2 и выше. Чтобы настроить шифры TLS 1.3 в OpenSSL, используйте команду `ciphersuites`.

На одном уровне может быть указано несколько директив `proxy_ssl_conf_command`. Директивы наследуются с предыдущего уровня конфигурации при условии, что на данном уровне не описаны свои директивы `proxy_ssl_conf_command`.

### ⚠ Предупреждение

Следует учитывать, что изменение настроек OpenSSL напрямую может привести к неожиданному поведению.

## proxy\_ssl\_crl

|                  |                             |
|------------------|-----------------------------|
| <i>Синтаксис</i> | proxy_ssl_crl <i>файл</i> ; |
| По умолчанию     | —                           |
| <i>Контекст</i>  | stream, server              |

Указывает файл с отозванными сертификатами (CRL) в формате PEM, используемыми при *проверке* сертификата проксируемого сервера.

## proxy\_ssl\_name

|                  |                                           |
|------------------|-------------------------------------------|
| <i>Синтаксис</i> | proxy_ssl_name <i>имя</i> ;               |
| По умолчанию     | proxy_ssl_name <i>хост</i> из proxy_pass; |
| <i>Контекст</i>  | stream, server                            |

Позволяет переопределить имя сервера, используемое при *проверке* сертификата проксируемого сервера, а также для *передачи его через SNI* при установлении соединения с проксируемым сервером. Имя сервера можно также задать с помощью переменных.

По умолчанию используется имя хоста из адреса, заданного директивой *proxy\_pass*.

## proxy\_ssl\_ntls

|                  |                          |
|------------------|--------------------------|
| <i>Синтаксис</i> | proxy_ssl_ntls on   off; |
| По умолчанию     | proxy_ssl_ntls off;      |
| <i>Контекст</i>  | stream, server           |

Включает клиентскую поддержку NTLS при использовании TLS библиотеки TongSuo.

```
server {
 proxy_ssl_ntls on;

 proxy_ssl_certificate sign.crt enc.crt;
 proxy_ssl_certificate_key sign.key enc.key;

 proxy_ssl_ciphers "ECC-SM2-WITH-SM4-SM3:ECDHE-SM2-WITH-SM4-SM3:RSA";

 proxy_pass backend:12345;
}
```

### Примечание

Angie необходимо собрать с использованием параметра конфигурации `--with-ntls`, с соответствующей SSL библиотекой с поддержкой NTLS

```
./configure --with-openssl=../Tongsuo-8.3.0 \
 --with-openssl-opt=enable-ntls \
 --with-ntls
```

## proxy\_ssl\_password\_file

|                  |                                       |
|------------------|---------------------------------------|
| <i>Синтаксис</i> | proxy_ssl_password_file <i>файл</i> ; |
| По умолчанию     | —                                     |
| <i>Контекст</i>  | stream, server                        |

Задаёт файл с паролями от *секретных ключей*, где каждый пароль указан на отдельной строке. Пароли применяются по очереди в момент загрузки ключа.

## proxy\_ssl\_protocols

|                  |                                                                            |
|------------------|----------------------------------------------------------------------------|
| <i>Синтаксис</i> | proxy_ssl_protocols [SSLv2] [SSLv3] [TLSv1] [TLSv1.1] [TLSv1.2] [TLSv1.3]; |
| По умолчанию     | proxy_ssl_protocols TLSv1.2 TLSv1.3;                                       |
| <i>Контекст</i>  | stream, server                                                             |

Изменено в версии 1.2.0: Параметр TLSv1.3 добавлен к используемым по умолчанию.

Разрешает указанные протоколы для соединений с проксируемым сервером.

## proxy\_ssl\_server\_name

|                  |                                 |
|------------------|---------------------------------|
| <i>Синтаксис</i> | proxy_ssl_server_name on   off; |
| По умолчанию     | proxy_ssl_server_name off;      |
| <i>Контекст</i>  | stream, server                  |

Разрешает или запрещает передачу имени сервера, заданного директивой *proxy\_ssl\_name*, через расширение Server Name Indication протокола TLS (SNI, RFC 6066) при установлении соединения с проксируемым сервером.

## proxy\_ssl\_session\_reuse

|                  |                                   |
|------------------|-----------------------------------|
| <i>Синтаксис</i> | proxy_ssl_session_reuse on   off; |
| По умолчанию     | proxy_ssl_session_reuse on;       |
| <i>Контекст</i>  | stream, server                    |

Определяет, использовать ли повторно SSL-сессии при работе с проксируемым сервером. Если в логах появляются ошибки "*SSL3\_GET\_FINISHED:digest check failed*", то можно попробовать включить повторное использование сессий.



### proxy\_ssl\_trusted\_certificate

|                  |                                             |
|------------------|---------------------------------------------|
| <i>Синтаксис</i> | proxy_ssl_trusted_certificate <i>файл</i> ; |
| По умолчанию     | —                                           |
| <i>Контекст</i>  | stream, server                              |

Задаёт файл с доверенными сертификатами СА в формате PEM, используемыми при *проверке* сертификата проксируемого HTTPS-сервера.

### proxy\_ssl\_verify

|                  |                            |
|------------------|----------------------------|
| <i>Синтаксис</i> | proxy_ssl_verify on   off; |
| По умолчанию     | proxy_ssl_verify off;      |
| <i>Контекст</i>  | stream, server             |

Разрешает или запрещает проверку сертификата проксируемого сервера.

### proxy\_ssl\_verify\_depth

|                  |                                       |
|------------------|---------------------------------------|
| <i>Синтаксис</i> | proxy_ssl_verify_depth <i>число</i> ; |
| По умолчанию     | proxy_ssl_verify_depth 1;             |
| <i>Контекст</i>  | stream, server                        |

Устанавливает глубину проверки в цепочке сертификатов проксируемого сервера.

### proxy\_timeout

|                  |                              |
|------------------|------------------------------|
| <i>Синтаксис</i> | proxy_timeout <i>время</i> ; |
| По умолчанию     | proxy_timeout 10m;           |
| <i>Контекст</i>  | stream, server               |

Задаёт таймаут между двумя идущими подряд операциями чтения или записи на клиентском соединении или соединении с проксируемым сервером. Если по истечении этого времени данные не передавались, соединение закрывается.

## proxy\_upload\_rate

|                  |                                     |
|------------------|-------------------------------------|
| <i>Синтаксис</i> | proxy_upload_rate <i>скорость</i> ; |
| По умолчанию     | proxy_upload_rate 0;                |
| <i>Контекст</i>  | stream, server                      |

Ограничивает скорость чтения данных от клиента. Скорость задается в байтах в секунду.

|   |                                |
|---|--------------------------------|
| 0 | отключает ограничение скорости |
|---|--------------------------------|

### Примечание

Ограничение устанавливается на соединение, поэтому, если клиент одновременно откроет два соединения, суммарная скорость будет вдвое выше заданного ограничения.

В значении параметра можно использовать переменные. Это может быть полезно в случаях, когда скорость нужно ограничивать в зависимости от какого-либо условия:

```
map $slow $rate {
 1 4k;
 2 8k;
}

proxy_upload_rate $rate;
```

## RDP Preread

При использовании протокола RDP позволяет извлекать cookie, которые используются для идентификации и управления сессиями, до момента принятия решения о балансировке.

При сборке из исходного кода модуль необходимо включить с помощью параметра сборки `--with-stream_rdp_preread_module`. В пакетах и образах из наших репозиториях модуль включен в сборку.

### Пример конфигурации

#### Привязка к выдавшему cookie серверу

Конфигурация использует режим `learn` директивы `sticky`:

```
stream {

 rdp_preread on;

 upstream rdp {

 server 127.0.0.1:3390 sid=a;
 server 127.0.0.1:3391 sid=b;

 sticky learn lookup=$rdp_cookie create=$rdp_cookie zone=sessions:1m;
```

```
}
}
```

## Директивы

### rdp\_preread

|                  |                       |
|------------------|-----------------------|
| <i>Синтаксис</i> | rdp_preread on   off; |
| По умолчанию     | rdp_preread off;      |
| <i>Контекст</i>  | stream, server        |

Управляет извлечением информации из cookie протокола RDP на этапе *предварительного чтения*. Если параметр включен (on), то в контексте, где он задан, заполняются перечисленные ниже переменные.

### Встроенные переменные

Семантика значений в составе cookie зависит от версии протокола RDP.

`$rdp_cookie`

Значение cookie целиком.

`$rdp_cookie_<имя>`

Значение поля cookie с заданным именем.

### RealIP

Позволяет менять адрес и порт клиента на переданные в заголовке протокола PROXY. Протокол PROXY должен быть предварительно включен при помощи установки параметра `proxy_protocol` в директиве *listen*.

При сборке из исходного кода модуль не собирается по умолчанию; его необходимо включить с помощью параметра сборки `--with-stream_realip_module`. В пакетах и образах из наших репозиторий модуль включен в сборку.

### Пример конфигурации

```
listen 12345 proxy_protocol;

set_real_ip_from 192.168.1.0/24;
set_real_ip_from 192.168.2.1;
set_real_ip_from 2001:0db8::/32;
```

## Директивы

### set\_real\_ip\_from

|                  |                                                      |
|------------------|------------------------------------------------------|
| <i>Синтаксис</i> | <code>set_real_ip_from адрес   CIDR   unix::;</code> |
| По умолчанию     | —                                                    |
| <i>Контекст</i>  | stream, server                                       |

Задаёт доверенные адреса, которые передают верный адрес для замены. Если указано специальное значение `unix::`, доверенными будут считаться все UNIX-сокеты.

## Встроенные переменные

`$realip_remote_addr`

хранит исходный адрес клиента

`$realip_remote_port`

хранит исходный порт клиента

## Return

Позволяет отправить заданное значение клиенту и после этого закрыть соединение.

## Пример конфигурации

```
server {
 listen 12345;
 return $time_iso8601;
}
```

## Директивы

### return

|                  |                               |
|------------------|-------------------------------|
| <i>Синтаксис</i> | <code>return значение;</code> |
| По умолчанию     | —                             |
| <i>Контекст</i>  | server                        |

Задаёт значение, отправляемое клиенту. В качестве значения можно использовать текст, переменные и их комбинации.

## Set

Позволяет устанавливать значение переменной.

### Пример конфигурации

```
server {
 listen 12345;
 set $true 1;
}
```

## Директивы

### set

|                  |                                         |
|------------------|-----------------------------------------|
| <i>Синтаксис</i> | <code>set \$переменная значение;</code> |
| По умолчанию     | —                                       |
| <i>Контекст</i>  | server                                  |

Устанавливает значение указанной переменной. В качестве значения можно использовать текст, переменные и их комбинации.

## Split Clients

Модуль генерирует переменные для А/В-тестирования, канареечных релизов и других сценариев, которые направляют определенный процент клиентов на один сервер или конфигурацию, а остальных — куда-то еще.

### Пример конфигурации

```
stream {
 # ...
 split_clients "${remote_addr}AAA" $upstream {
 0.5% feature_test1;
 2.0% feature_test2;
 * production;
 }

 server {
 # ...
 proxy_pass $upstream;
 }
}
```

## Директивы

### split\_clients

|                  |                                                        |
|------------------|--------------------------------------------------------|
| <i>Синтаксис</i> | <code>split_clients строка \$переменная { ... }</code> |
| По умолчанию     | —                                                      |
| <i>Контекст</i>  | stream                                                 |

Создает *\$переменную*, хэшируя *строку*; переменные в *строке* подставляются, результат хэшируется, затем по значению хэша выбирается строковое значение *\$переменной*.

Функция хэширования использует [MurmurHash2](#) (32 бит), и весь диапазон ее значений (с 0 по 4294967295) сопоставляется с корзинами в порядке появления; процентные величины определяют размер корзин. В конце может стоять метасимвол (\*); хэши, не попавшие в другие корзины, сопоставляются с приданным ему значением.

Пример:

```
split_clients "${remote_addr}AAA" $variant {
 0.5% .one;
 2.0% .two;
 * "";
}
```

Здесь после подстановки в строке *\$remote\_addrAAA* значения хэша распределяются следующим образом:

- значения от 0 до 21474835 (0,5%) дают *.one*;
- значения от 21474836 до 107374180 (2%) дают *.two*;
- значения от 107374181 до 4294967295 (все остальные) дают *""* (пустую строку).

## SSL

Обеспечивает необходимую поддержку для работы прокси-сервера по протоколу SSL/TLS.

При сборке из исходного кода модуль не собирается по умолчанию; его необходимо включить с помощью параметра сборки `--with-stream_ssl_module`.

В пакетах и образах из наших репозиториях модуль включен в сборку.

### Примечание

Для этого модуля нужна библиотека OpenSSL.

## Пример конфигурации

Для уменьшения загрузки процессора рекомендуется

- установить число *рабочих процессов* равным числу процессоров,
- включить *разделяемый кэш сессий*,
- выключить *встроенный кэш сессий*
- и, возможно, увеличить *время жизни* сессии (по умолчанию 5 минут):

```
worker_processes auto;

stream {

 #...

 server {
 listen 12345 ssl;

 ssl_protocols TLSv1.2 TLSv1.3;
 ssl_ciphers AES128-SHA:AES256-SHA:RC4-SHA:DES-CBC3-SHA:RC4-MD5;
 ssl_certificate /usr/local/angie/conf/cert.pem;
 ssl_certificate_key /usr/local/angie/conf/cert.key;
 ssl_session_cache shared:SSL:10m;
 ssl_session_timeout 10m;

 # ...
 }
}
```

## Директивы

### ssl\_alpn

|                  |                                |
|------------------|--------------------------------|
| <i>Синтаксис</i> | ssl_alpn <i>протокол ...</i> ; |
| По умолчанию     | —                              |
| <i>Контекст</i>  | stream, server                 |

Задаёт список поддерживаемых протоколов [ALPN](#). Один из протоколов должен быть *согласован*, если клиент использует ALPN:

```
map $ssl_alpn_protocol $proxy {
 h2 127.0.0.1:8001;
 http/1.1 127.0.0.1:8002;
}

server {
 listen 12346;
 proxy_pass $proxy;
 ssl_alpn h2 http/1.1;
}
```

## ssl\_certificate

|                  |                                    |
|------------------|------------------------------------|
| <i>Синтаксис</i> | <code>ssl_certificate файл;</code> |
| По умолчанию     | —                                  |
| <i>Контекст</i>  | stream, server                     |

Указывает файл с сертификатом в формате PEM для данного сервера. Если вместе с основным сертификатом нужно указать промежуточные, то они должны находиться в этом же файле в следующем порядке — сначала основной сертификат, а затем промежуточные. В этом же файле может находиться секретный ключ в формате PEM.

Директива может быть указана несколько раз для загрузки сертификатов разных типов, например RSA и ECDSA:

```
server {
 listen 12345 ssl;

 ssl_certificate example.com.rsa.crt;
 ssl_certificate_key example.com.rsa.key;

 ssl_certificate example.com.ecdsa.crt;
 ssl_certificate_key example.com.ecdsa.key;

 # ...
}
```

Возможность задавать отдельные цепочки сертификатов для разных сертификатов есть только в OpenSSL 1.0.2 и выше. Для более старых версий следует указывать только одну цепочку сертификатов.

### Примечание

В имени файла можно использовать переменные при использовании OpenSSL 1.0.2 и выше:

```
ssl_certificate $ssl_server_name.crt;
ssl_certificate_key $ssl_server_name.key;
```

При использовании переменных сертификат загружается при каждой операции SSL-рукопожатия, что может отрицательно влиять на производительность.

Вместо **файла** можно указать значение "data:\$переменная", при котором сертификат загружается из переменной без использования промежуточных файлов.

Ненадлежащее использование подобного синтаксиса может быть небезопасно, например данные секретного ключа могут попасть в *лог ошибок*.



## ssl\_certificate\_key

|                  |                                   |
|------------------|-----------------------------------|
| <i>Синтаксис</i> | ssl_certificate_key <i>файл</i> ; |
| По умолчанию     | —                                 |
| <i>Контекст</i>  | stream, server                    |

Указывает файл с секретным ключом в формате PEM для данного виртуального сервера.

### Примечание

В имени файла можно использовать переменные при использовании OpenSSL 1.0.2 и выше.

Вместо файла можно указать значение "engine:*имя*:id", которое загружает ключ с указанным *id* из OpenSSL engine с заданным именем.

Вместо файла также можно указать значение "data:*\$переменная*", при котором секретный ключ загружается из переменной без использования промежуточных файлов. При этом следует учитывать, что ненадлежащее использование подобного синтаксиса может быть небезопасно, например данные секретного ключа могут попасть в *лог ошибок*.

## ssl\_ciphers

|                  |                               |
|------------------|-------------------------------|
| <i>Синтаксис</i> | ssl_ciphers <i>шифры</i> ;    |
| По умолчанию     | ssl_ciphers HIGH:!aNULL:!MD5; |
| <i>Контекст</i>  | stream, server                |

Описывает разрешенные шифры. Шифры задаются в формате, поддерживаемом библиотекой OpenSSL, например:

```
ssl_ciphers ALL:!aNULL:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP;
```

Список шифров зависит от установленной версии OpenSSL. Полный список можно посмотреть с помощью команды `openssl ciphers`.

### Предупреждение

Директива `ssl_ciphers` не настраивает шифры для TLS 1.3 при использовании OpenSSL. Для настройки шифров TLS 1.3 в OpenSSL используйте директиву `ssl_conf_command`, добавленную для расширенной конфигурации SSL.

- В LibreSSL шифры TLS 1.3 можно настраивать с помощью `ssl_ciphers`.
- В BoringSSL шифры TLS 1.3 настроить невозможно.

## ssl\_client\_certificate

|                  |                                      |
|------------------|--------------------------------------|
| <i>Синтаксис</i> | ssl_client_certificate <i>файл</i> ; |
| По умолчанию     | —                                    |
| <i>Контекст</i>  | stream, server                       |

Задаёт файл с доверенными сертификатами CA в формате PEM, которые используются для *проверки* клиентских сертификатов и ответов OCSP, если включен *ssl\_stapling*.

Список сертификатов будет отправляться клиентам. Если это нежелательно, можно воспользоваться директивой *ssl\_trusted\_certificate*.

## ssl\_conf\_command

|                  |                                        |
|------------------|----------------------------------------|
| <i>Синтаксис</i> | ssl_conf_command <i>имя значение</i> ; |
| По умолчанию     | —                                      |
| <i>Контекст</i>  | stream, server                         |

Задаёт произвольные *конфигурационные команды* OpenSSL.

### Примечание

Директива поддерживается при использовании OpenSSL 1.0.2 и выше. Чтобы настроить шифры TLS 1.3 в OpenSSL, используйте команду *ciphersuites*.

На одном уровне может быть указано несколько директив *ssl\_conf\_command*:

```
ssl_conf_command Options PrioritizeChaCha;
ssl_conf_command Ciphersuites TLS_CHACHA20_POLY1305_SHA256;
```

Директивы наследуются с предыдущего уровня конфигурации при условии, что на данном уровне не описаны свои директивы *ssl\_conf\_command*.

### Предупреждение

Изменение настроек OpenSSL напрямую может привести к неожиданному поведению.

## ssl\_crl

|                  |                       |
|------------------|-----------------------|
| <i>Синтаксис</i> | ssl_crl <i>файл</i> ; |
| По умолчанию     | —                     |
| <i>Контекст</i>  | stream, server        |

Указывает файл с отозванными сертификатами (CRL) в формате PEM, используемыми для *проверки* клиентских сертификатов.

## ssl\_dhparam

|                  |                                |
|------------------|--------------------------------|
| <i>Синтаксис</i> | <code>ssl_dhparam файл;</code> |
| По умолчанию     | —                              |
| <i>Контекст</i>  | stream, server                 |

Указывает файл с параметрами для DHE-шифров.

### Предупреждение

По умолчанию параметры не заданы, и соответственно DHE-шифры не будут использоваться.

## ssl\_early\_data

|                  |                                       |
|------------------|---------------------------------------|
| <i>Синтаксис</i> | <code>ssl_early_data on   off;</code> |
| По умолчанию     | <code>ssl_early_data off;</code>      |
| <i>Контекст</i>  | stream, server                        |

Разрешает или запрещает TLS 1.3 [early data](#).

### Примечание

Директива поддерживается при использовании OpenSSL 1.1.1 и выше или [BoringSSL](#).

## ssl\_ecdh\_curve

|                  |                                     |
|------------------|-------------------------------------|
| <i>Синтаксис</i> | <code>ssl_ecdh_curve кривая;</code> |
| По умолчанию     | <code>ssl_ecdh_curve auto;</code>   |
| <i>Контекст</i>  | stream, server                      |

Задаёт кривую для ECDHE-шифров.

### Примечание

При использовании OpenSSL 1.0.2 и выше можно указывать несколько кривых, например:

```
ssl_ecdh_curve prime256v1:secp384r1;
```

Специальное значение `auto` соответствует встроенному в библиотеку OpenSSL списку кривых для OpenSSL 1.0.2 и выше, или `prime256v1` для более старых версий.

### Примечание

При использовании OpenSSL 1.0.2 и выше директива задает список кривых, поддерживаемых сервером. Поэтому для работы ECDSA-сертификатов важно, чтобы список включал кривые, используемые в сертификатах.

### ssl\_handshake\_timeout

|                  |                                           |
|------------------|-------------------------------------------|
| <i>Синтаксис</i> | <code>ssl_handshake_timeout время;</code> |
| По умолчанию     | <code>ssl_handshake_timeout 60s;</code>   |
| <i>Контекст</i>  | <code>stream, server</code>               |

Задаёт таймаут для завершения операции SSL-рукопожатия.

### ssl\_ocsp

|                  |                                        |
|------------------|----------------------------------------|
| <i>Синтаксис</i> | <code>ssl_ocsp on   off   leaf;</code> |
| По умолчанию     | <code>ssl_ocsp off;</code>             |
| <i>Контекст</i>  | <code>http, server</code>              |

Включает проверку OCSP для цепочки клиентских сертификатов. Параметр `leaf` включает проверку только клиентского сертификата.

Для работы проверки OCSP необходимо дополнительно установить значение директивы `ssl_verify_client` в `on` или `optional`.

Для преобразования имени хоста OCSP-респондера в адрес необходимо дополнительно задать директиву `resolver`.

Пример:

```
ssl_verify_client on;
ssl_ocsp on;
resolver 127.0.0.53;
```

### ssl\_ocsp\_cache

|                  |                                                        |
|------------------|--------------------------------------------------------|
| <i>Синтаксис</i> | <code>ssl_ocsp_cache off   [shared:имя:размер];</code> |
| По умолчанию     | <code>ssl_ocsp_cache off;</code>                       |
| <i>Контекст</i>  | <code>http, server</code>                              |

Задаёт имя и размер кэша, который хранит статус клиентских сертификатов для проверки OCSP-ответов. Кэш разделяется между всеми рабочими процессами. Кэш с одинаковым названием может использоваться в нескольких виртуальных серверах.

Параметр `off` запрещает использование кэша.

## ssl\_ocsp\_responder

|                  |                                      |
|------------------|--------------------------------------|
| <i>Синтаксис</i> | <code>ssl_ocsp_responder uri;</code> |
| По умолчанию     | —                                    |
| <i>Контекст</i>  | http, server                         |

Переопределяет URI OCSP-респондера, указанный в расширении сертификата "Authority Information Access" для *проверки* клиентских сертификатов.

Поддерживаются только OCSP-респондеры на основе `http://`:

```
ssl_ocsp_responder http://ocsp.example.com/;
```

## ssl\_ntls

|                  |                                 |
|------------------|---------------------------------|
| <i>Синтаксис</i> | <code>ssl_ntls on   off;</code> |
| По умолчанию     | <code>ssl_ntls off;</code>      |
| <i>Контекст</i>  | stream, server                  |

Включает серверную поддержку NTLS при использовании TLS библиотеки [TongSuo](#)

```
listen ... ssl;
ssl_ntls on;
```

### Примечание

Angie необходимо собрать с использованием параметра конфигурации `--with-ntls`, с соответствующей SSL библиотекой с поддержкой NTLS

```
./configure --with-openssl=../Tongsuo-8.3.0 \
 --with-openssl-opt=enable-ntls \
 --with-ntls
```

## ssl\_password\_file

|                  |                                      |
|------------------|--------------------------------------|
| <i>Синтаксис</i> | <code>ssl_password_file файл;</code> |
| По умолчанию     | —                                    |
| <i>Контекст</i>  | stream, server                       |

Задаёт файл с паролями от *секретных ключей*, где каждый пароль указан на отдельной строке. Пароли применяются по очереди в момент загрузки ключа.

Пример:

```
stream {
 ssl_password_file /etc/keys/global.pass;
 ...
}
```

```
server {
 listen 127.0.0.1:12345;
 ssl_certificate_key /etc/keys/first.key;
}

server {
 listen 127.0.0.1:12346;

 # вместо файла можно указать именованный канал
 ssl_password_file /etc/keys/fifo;
 ssl_certificate_key /etc/keys/second.key;
}
```

### ssl\_prefer\_server\_ciphers

|                  |                                     |
|------------------|-------------------------------------|
| <i>Синтаксис</i> | ssl_prefer_server_ciphers on   off; |
| По умолчанию     | ssl_prefer_server_ciphers off;      |
| <i>Контекст</i>  | stream, server                      |

При использовании протоколов SSLv3 и TLS устанавливает приоритет серверных шифров над клиентскими.

### ssl\_protocols

|                  |                                                                      |
|------------------|----------------------------------------------------------------------|
| <i>Синтаксис</i> | ssl_protocols [SSLv2] [SSLv3] [TLSv1] [TLSv1.1] [TLSv1.2] [TLSv1.3]; |
| По умолчанию     | ssl_protocols TLSv1.2 TLSv1.3;                                       |
| <i>Контекст</i>  | stream, server                                                       |

Изменено в версии 1.2.0: Параметр TLSv1.3 добавлен к используемым по умолчанию.

Разрешает указанные протоколы.

#### Примечание

Параметры TLSv1.1 и TLSv1.2 работают только при использовании OpenSSL 1.0.1 и выше.

Параметр TLSv1.3 работает только при использовании OpenSSL 1.1.1 и выше.

### ssl\_session\_cache

|                  |                                                                             |
|------------------|-----------------------------------------------------------------------------|
| <i>Синтаксис</i> | ssl_session_cache off   none   [builtin[:размер]] [shared:название:размер]; |
| По умолчанию     | ssl_session_cache none;                                                     |
| <i>Контекст</i>  | stream, server                                                              |

Задаёт тип и размеры кэшей для хранения параметров сессий. Тип кэша может быть следующим:

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>off</code>     | жесткое запрещение использования кэша сессий: Angie явно сообщает клиенту, что сессии не могут использоваться повторно.                                                                                                                                                                                                                                                                                                                                                             |
| <code>none</code>    | мягкое запрещение использования кэша сессий: Angie сообщает клиенту, что сессии могут использоваться повторно, но на самом деле не хранит параметры сессии в кэше.                                                                                                                                                                                                                                                                                                                  |
| <code>builtin</code> | встроенный в OpenSSL кэш, используется в рамках только одного рабочего процесса. Размер кэша задается в сессиях. Если размер не задан, то он равен 20480 сессиям. Использование встроенного кэша может вести к фрагментации памяти.                                                                                                                                                                                                                                                 |
| <code>shared</code>  | кэш, разделяемый между всеми рабочими процессами. Размер кэша задается в байтах, в 1 мегабайт может поместиться около 4000 сессий. У каждого разделяемого кэша должно быть произвольное название. Кэш с одинаковым названием может использоваться в нескольких серверах. Также он используется для автоматического создания, хранения и периодического обновления ключей сессионных билетов TLS, если они не указаны явно с помощью директивы <code>ssl_session_ticket_key</code> . |

Можно использовать одновременно оба типа кэша, например:

```
ssl_session_cache builtin:1000 shared:SSL:10m;
```

однако использование только разделяемого кэша без встроенного должно быть более эффективным.

### `ssl_session_ticket_key`

|                  |                                                   |
|------------------|---------------------------------------------------|
| <i>Синтаксис</i> | <code>ssl_session_ticket_key</code> <i>файл</i> ; |
| По умолчанию     | —                                                 |
| <i>Контекст</i>  | stream, server                                    |

Задаёт файл с секретным ключом, применяемым при шифровании и расшифровке сессионных билетов TLS. Директива необходима, если один и тот же ключ нужно использовать на нескольких серверах. По умолчанию используется случайно сгенерированный ключ.

Если указано несколько ключей, то только первый ключ используется для шифрования сессионных билетов TLS. Это позволяет настроить ротацию ключей, например:

```
ssl_session_ticket_key current.key;
ssl_session_ticket_key previous.key;
```

Файл должен содержать 80 или 48 байт случайных данных и может быть создан следующей командой:

```
openssl rand 80 > ticket.key
```

В зависимости от размера файла для шифрования будет использоваться либо AES256 (для 80-байтных ключей), либо AES128 (для 48-байтных ключей).

## ssl\_session\_tickets

|                  |                                            |
|------------------|--------------------------------------------|
| <i>Синтаксис</i> | <code>ssl_session_tickets on   off;</code> |
| По умолчанию     | <code>ssl_session_tickets on;</code>       |
| <i>Контекст</i>  | stream, server                             |

Разрешает или запрещает возобновление сессий при помощи [сессионных билетов TLS](#).

## ssl\_session\_timeout

|                  |                                         |
|------------------|-----------------------------------------|
| <i>Синтаксис</i> | <code>ssl_session_timeout время;</code> |
| По умолчанию     | <code>ssl_session_timeout 5m;</code>    |
| <i>Контекст</i>  | stream, server                          |

Задаёт время, в течение которого клиент может повторно использовать параметры сессии.

## ssl\_stapling

|                  |                                     |
|------------------|-------------------------------------|
| <i>Синтаксис</i> | <code>ssl_stapling on   off;</code> |
| По умолчанию     | <code>ssl_stapling off;</code>      |
| <i>Контекст</i>  | http, server                        |

Разрешает или запрещает [прикрепление OCSP-ответов сервером](#). Пример:

```
ssl_stapling on;
resolver 127.0.0.53;
```

Для работы OCSP-прикрепления должен быть известен сертификат издателя сертификата сервера. Если в заданном директивой `ssl_certificate` файле не содержится промежуточных сертификатов, то сертификат издателя сертификата сервера следует поместить в файл, заданный директивой `ssl_trusted_certificate`.

### Предупреждение

Для преобразования имени хоста OCSP-респондера в адрес необходимо дополнительно задать директиву `resolver`.



## ssl\_stapling\_file

|                  |                                      |
|------------------|--------------------------------------|
| <i>Синтаксис</i> | <code>ssl_stapling_file файл;</code> |
| По умолчанию     | —                                    |
| <i>Контекст</i>  | http, server                         |

Если значение задано, то вместо опроса OCSP-респондера, указанного в сертификате сервера, ответ берется из указанного файла.

Ответ должен быть в формате DER и может быть сгенерирован командой `openssl ocsp`.

## ssl\_stapling\_responder

|                  |                                          |
|------------------|------------------------------------------|
| <i>Синтаксис</i> | <code>ssl_stapling_responder uri;</code> |
| По умолчанию     | —                                        |
| <i>Контекст</i>  | http, server                             |

Переопределяет URI OCSP-респондера, указанный в расширении сертификата "Authority Information Access".

Поддерживаются только OCSP-респондеры на основе `http://`:

```
ssl_stapling_responder http://ocsp.example.com/;
```

## ssl\_stapling\_verify

|                  |                                            |
|------------------|--------------------------------------------|
| <i>Синтаксис</i> | <code>ssl_stapling_verify on   off;</code> |
| По умолчанию     | <code>ssl_stapling_verify off;</code>      |
| <i>Контекст</i>  | http, server                               |

Разрешает или запрещает проверку ответов OCSP сервером.

Для работоспособности проверки сертификат издателя сертификата сервера, корневой сертификат и все промежуточные сертификаты должны быть указаны как доверенные с помощью директивы `ssl_trusted_certificate`.

## ssl\_trusted\_certificate

|                  |                                            |
|------------------|--------------------------------------------|
| <i>Синтаксис</i> | <code>ssl_trusted_certificate файл;</code> |
| По умолчанию     | —                                          |
| <i>Контекст</i>  | stream, server                             |

Задаёт файл с доверенными сертификатами CA в формате PEM, которые используются для *про-верки* клиентских сертификатов.

В отличие от `ssl_client_certificate`, список этих сертификатов не будет отправляться клиентам.

## ssl\_verify\_client

|                  |                                                                      |
|------------------|----------------------------------------------------------------------|
| <i>Синтаксис</i> | <code>ssl_verify_client on   off   optional   optional_no_ca;</code> |
| По умолчанию     | <code>ssl_verify_client off;</code>                                  |
| <i>Контекст</i>  | stream, server                                                       |

Разрешает проверку клиентских сертификатов. Результат проверки доступен через переменную `$ssl_client_verify`. Если при проверке клиентского сертификата произошла ошибка или клиент не предоставил требуемый сертификат, соединение закрывается.

|                             |                                                                                                                                                                                                                              |
|-----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>optional</code>       | запрашивает клиентский сертификат, и если сертификат был предоставлен, проверяет его                                                                                                                                         |
| <code>optional_no_ca</code> | запрашивает сертификат клиента, но не требует, чтобы он был подписан доверенным сертификатом СА. Это предназначено для случаев, когда фактическая проверка сертификата осуществляется внешним по отношению к Angie сервисом. |

## ssl\_verify\_depth

|                  |                                      |
|------------------|--------------------------------------|
| <i>Синтаксис</i> | <code>ssl_verify_depth число;</code> |
| По умолчанию     | <code>ssl_verify_depth 1;</code>     |
| <i>Контекст</i>  | stream, server                       |

Устанавливает глубину проверки в цепочке клиентских сертификатов.

## Встроенные переменные

Модуль `stream_ssl` поддерживает встроенные переменные:

`$ssl_alpn_protocol`

возвращает протокол, выбранный при помощи ALPN во время SSL-рукопожатия, либо пустую строку.

`$ssl_cipher`

возвращает название используемого шифра для установленного SSL-соединения.

## `$ssl_ciphers`

возвращает список шифров, поддерживаемых клиентом. Известные шифры указаны по имени, неизвестные указаны в шестнадцатеричном виде, например:

AES128-SHA:AES256-SHA:0x00ff

### **Примечание**

Переменная полностью поддерживается при использовании OpenSSL версии 1.0.2 и выше. При использовании более старых версий переменная доступна только для новых сессий и может содержать только известные шифры.

## `$ssl_client_cert`

возвращает клиентский сертификат для установленного SSL-соединения в формате PEM перед каждой строкой которого, кроме первой, вставляется символ табуляции.

## `$ssl_client_fingerprint`

возвращает SHA1-отпечаток клиентского сертификата для установленного SSL-соединения.

## `$ssl_client_i_dn`

возвращает строку "issuer DN" клиентского сертификата для установленного SSL-соединения согласно RFC 2253.

## `$ssl_client_raw_cert`

возвращает клиентский сертификат для установленного SSL-соединения в формате PEM.

## `$ssl_client_s_dn`

возвращает строку "subject DN" клиентского сертификата для установленного SSL-соединения согласно RFC 2253.

## `$ssl_client_serial`

возвращает серийный номер клиентского сертификата для установленного SSL-соединения.

## `$ssl_client_v_end`

возвращает дату окончания срока действия клиентского сертификата.

`$ssl_client_v_remain`

возвращает число дней, оставшихся до истечения срока действия клиентского сертификата.

`$ssl_client_v_start`

возвращает дату начала срока действия клиентского сертификата.

`$ssl_client_verify`

возвращает результат проверки клиентского сертификата: `SUCCESS`, `FAILED:reason` и, если сертификат не был предоставлен, `NONE`.

`$ssl_curve`

возвращает согласованную кривую, использованную для обмена ключами во время SSL-рукопожатия. Известные кривые указаны по имени, неизвестные указаны в шестнадцатеричном виде, например:

`prime256v1`

#### **Примечание**

Переменная поддерживается при использовании OpenSSL версии 3.0 и выше. При использовании более старых версий значением переменной будет пустая строка.

`$ssl_curves`

возвращает список кривых, поддерживаемых клиентом. Известные кривые указаны по имени, неизвестные указаны в шестнадцатеричном виде, например:

`0x001d:prime256v1:secp521r1:secp384r1`

#### **Примечание**

Переменная поддерживается при использовании OpenSSL версии 1.0.2 и выше. При использовании более старых версий значением переменной будет пустая строка.

Переменная доступна только для новых сессий.

`$ssl_early_data`

возвращает 1, если используется TLS 1.3 *early data* и операция SSL-рукопожатия не завершена, иначе "".

`$ssl_protocol`

возвращает протокол установленного SSL-соединения.

`$ssl_server_cert_type`

принимает значения RSA, DSA, ECDSA, ED448, ED25519, SM2, RSA-PSS или `unknown` в зависимости от типа сертификата и ключа сервера.

`$ssl_server_name`

возвращает имя сервера, запрошенное через [SNI](#).

`$ssl_session_id`

возвращает идентификатор сессии установленного SSL-соединения.

`$ssl_session_reused`

возвращает `r`, если сессия была использована повторно, иначе `"."`.

## SSL Preread

Позволяет извлекать информацию из сообщения [ClientHello](#) без терминации TLS, например имя сервера, запрошенное через [SNI](#), или протоколы, указанные в [ALPN](#).

При сборке из исходного кода модуль не собирается по умолчанию; его необходимо включить с помощью параметра сборки `--with-stream_ssl_preread_module`. В пакетах и образах из наших репозиториях модуль включен в сборку.

## Пример конфигурации

### Выбор апстрима по имени сервера

```
map $ssl_preread_server_name $name {
 backend.example.com backend;
 default backend2;
}

upstream backend {
 server 192.168.0.1:12345;
 server 192.168.0.2:12345;
}

upstream backend2 {
 server 192.168.0.3:12345;
 server 192.168.0.4:12345;
}

server {
 listen 12346;
 proxy_pass $name;
```

```
ssl_preread on;
}
```

### Выбор сервера по протоколу

```
map $ssl_preread_alpn_protocols $proxy {
 ~\bh2\b 127.0.0.1:8001;
 ~\bhttp/1.1\b 127.0.0.1:8002;
 ~\bxmlpp-client\b 127.0.0.1:8003;
}

server {
 listen 9000;
 proxy_pass $proxy;
 ssl_preread on;
}
```

### Выбор сервера по версии протокола SSL

```
map $ssl_preread_protocol $upstream {
 "" ssh.example.com:22;
 "TLSv1.2" new.example.com:443;
 default tls.example.com:443;
}

ssh и https на одном порту
server {
 listen 192.168.0.1:443;
 proxy_pass $upstream;
 ssl_preread on;
}
```

## Директивы

### ssl\_preread

|                  |                       |
|------------------|-----------------------|
| <i>Синтаксис</i> | ssl_preread on   off; |
| По умолчанию     | ssl_preread off;      |
| <i>Контекст</i>  | stream, server        |

Разрешает извлекать информацию из сообщения ClientHello на этапе *предварительного чтения*.

## Встроенные переменные

`$ssl_preread_protocol`

Максимальная версия протокола SSL, поддерживаемая клиентом.

`$ssl_preread_server_name`

Имя сервера, запрошенное через SNI.

`$ssl_preread_alpn_protocols`

Список протоколов, переданный клиентом через ALPN. Значения разделяются запятыми.

## Upstream

Предоставляет контекст для описания группы серверов, которые могут использоваться в директиве *proxy\_pass*.

## Пример конфигурации

```
upstream backend {
 hash $remote_addr consistent;
 zone backend 1m;

 server backend1.example.com:1935 weight=5;
 server unix:/tmp/backend3;
 server backend3.example.com service=_example._tcp resolve;

 server backup1.example.com:1935 backup;
 server backup2.example.com:1935 backup;
}

resolver 127.0.0.53 status_zone=resolver;

server {
 listen 1936;
 proxy_pass backend;
}
```

## Директивы

### upstream

|              |                                   |
|--------------|-----------------------------------|
| Синтаксис    | <code>upstream имя { ... }</code> |
| По умолчанию | —                                 |
| Контекст     | stream                            |

Описывает группу серверов. Серверы могут слушать на разных портах. Кроме того, можно одновременно использовать серверы, слушающие на TCP- и UNIX-сокетах.

Пример:

```
upstream backend {
 server backend1.example.com:1935 weight=5;
 server 127.0.0.1:1935 max_fails=3 fail_timeout=30s;
 server unix:/tmp/backend2;
 server backend3.example.com:1935 resolve;

 server backup1.example.com:1935 backup;
}
```

По умолчанию соединения распределяются по серверам циклически (в режиме round-robin) с учетом весов серверов. В вышеприведенном примере каждые 7 соединений будут распределены так: 5 соединений на backend1.example.com:1935 и по одному соединению на второй и третий серверы.

Если при попытке работы с сервером происходит ошибка, то соединение передается следующему серверу, и так далее до тех пор, пока не будут опробованы все работающие серверы. Если связь с серверами не удалась, соединение будет закрыто.

## server

|              |                                           |
|--------------|-------------------------------------------|
| Синтаксис    | <b>server</b> адрес [ <i>параметры</i> ]; |
| По умолчанию | —                                         |
| Контекст     | upstream                                  |

Задаёт адрес и другие параметры сервера. Адрес может быть указан в виде доменного имени или IP-адреса, и обязательного порта, или в виде пути UNIX-сокета, который указывается после префикса **unix:**. Доменное имя, которому соответствует несколько IP-адресов, задаёт сразу несколько серверов.

Могут быть заданы следующие параметры:

|                        |                                                                                                                                                                                                                                                                                      |
|------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>weight=число</b>    | Задаёт вес сервера. По умолчанию — 1.                                                                                                                                                                                                                                                |
| <b>max_conns=число</b> | Ограничивает максимальное число одновременных активных соединений к проксируемому серверу. Значение по умолчанию равно 0 и означает, что ограничения нет. Если группа не находится в <i>зоне</i> разделяемой памяти, то ограничение работает отдельно для каждого рабочего процесса. |

**max\_fails=число** — задаёт число неудачных попыток связи с сервером, которые должны произойти в течение заданного *fail\_timeout* времени для того, чтобы сервер считался недоступным; после этого он будет повторно проверен через то же самое время.

В данном случае неудачной попыткой считается ошибка или таймаут при установке соединения с сервером.

### Примечание

Если директива **server** в группе разрешается в несколько серверов, ее настройка **max\_fails** применяется к каждому серверу отдельно.

Если после разрешения всех директив **server** в апстриме остается только один сервер, настройка **max\_fails** не действует и будет проигнорирована.



|                          |                             |
|--------------------------|-----------------------------|
| <code>max_fails=1</code> | Число попыток по умолчанию. |
| <code>max_fails=0</code> | Отключает учет попыток.     |

`fail_timeout=время` — задает период времени, в течение которого должно произойти определенное число неудачных попыток связи с сервером (*max\_fails*), чтобы сервер считался недоступным. Затем сервер остается недоступным в течение того же самого времени, прежде чем будет проверен повторно.

Значение по умолчанию — 10 секунд.

#### **Примечание**

Если директива **server** в группе разрешается в несколько серверов, ее настройка **fail\_timeout** применяется к каждому серверу отдельно.

Если после разрешения всех директив **server** в апстриме остается только один сервер, настройка **fail\_timeout** не действует и будет проигнорирована.

|               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>backup</b> | <p>Помечает сервер как запасной. На него будут передаваться запросы в случае, если не работают основные серверы.</p> <p>Можно создать несколько групп резервирования серверов и задать уровень для каждой группы с помощью <code>backup=&lt;уровень_группы&gt;</code>. Например, <code>backup=1</code> (или просто <code>backup</code>) — группа резервных серверов первого уровня, (в API отображается как <code>true</code>), <code>backup=2</code> — группа резервных серверов второго уровня, (в API отображается как соответствующее число).</p> <p>Если директива <b>server</b> использует SRV-записи DNS для получения адресов проксируемых серверов, то уровень группы резервирования можно задать с помощью параметра <code>backup=prio</code>. В этом случае абсолютное значение приоритета <code>priority=n</code> из SRV-записи будет использовано в качестве уровня резервной группы: <code>backup=n</code>.</p> <p>Если параметр <code>backup=prio</code> не задан, уровень резервирования будет определяться через относительное значение приоритета <code>priority=n</code> из SRV-записи. В этом случае директива <code>backup=n</code> задает базовый уровень отсчета (например <code>backup=3</code>), а относительные уровни резервирования отсчитываются от этого уровня. Сервер с наименьшим значением <code>priority</code> получит уровень <code>backup=3</code>, следующий за ним — <code>backup=4</code> и так далее. Если уровень отсчета <code>backup=n</code> не задан, отсчет будет идти от нуля: сервер с наименьшим значением <code>priority</code> получит <code>backup=0</code> и станет основным, за ним — <code>backup=1</code>, резервный, и так далее.</p> <p>Если задана директива <code>backup_switch</code>, также применяется ее логика активного резервирования.</p> |
| <b>down</b>   | Помечает сервер как постоянно недоступный.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <b>drain</b>  | Помечает сервер как разгружаемый ( <code>draining</code> ); это значит, что он получает только запросы сессий, привязанных ранее через <i>sticky</i> . В остальном поведение такое же, как в режиме <b>down</b> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |

#### **Предупреждение**

Параметр **backup** нельзя использовать совместно с методами балансировки нагрузки *hash* и *random*.

Параметры **down** и **drain** взаимно исключают.

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>resolve</b>     | Позволяет отслеживать изменения списка IP-адресов, соответствующего доменному имени, и обновлять его без перезагрузки конфигурации. При этом группа должна находиться в <i>зоне разделяемой памяти</i> ; также должен быть определен <i>преобразователь имен в адреса</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| <b>service=имя</b> | <p>Включает преобразование SRV-записей DNS и задает имя сервиса. При указании этого параметра необходимо также задать параметр <i>resolve</i>, не указывая порт сервера при имени хоста.</p> <p>Если в имени службы нет точек, формируется имя по стандарту RFC: к имени службы добавляется префикс <code>_</code>, затем через точку добавляется <code>_tcp</code>. Так, имя службы <code>http</code> даст в результате <code>_http._tcp</code>.</p> <p>Angie разрешает SRV-записи, объединяя нормализованное имя службы и имя хоста и получая список серверов для полученной комбинации через DNS, вместе с их приоритетами и весами.</p> <ul style="list-style-type: none"> <li>• SRV-записи с наивысшим приоритетом (те, которые имеют минимальное значение приоритета) разрешаются как основные серверы, а прочие записи становятся запасными серверами. Если <code>backup</code> установлено с <code>server</code>, SRV-записи с наивысшим приоритетом разрешаются как запасные серверы, а прочие записи игнорируются.</li> <li>• Вес аналогичен параметру <code>weight</code> директивы <code>server</code>. Если вес задан как в самой директиве, так и в SRV-записи, используется вес, установленный в директиве.</li> </ul> |

В этом примере выполняется поиск записи `_http._tcp.backend.example.com`:

```
server backend.example.com service=http resolve;
```

|                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|-------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>sid=id</b>           | Задаёт ID сервера в группе. Если параметр не задан, то ID задается как шестнадцатеричный MD5-хэш IP-адреса и порта или пути UNIX-сокета.                                                                                                                                                                                                                                                                                                                                                                  |
| <b>slow_start=время</b> | <p>Задаёт <i>время</i> восстановления веса сервера, возвращающегося к работе при балансировке нагрузки методом <i>round-robin</i> или <i>least_conn</i>.</p> <p>Если параметр задан и сервер после сбоя снова считается работающим с точки зрения <i>max_fails</i> и <i>upstream_probe</i>, то такой сервер равномерно набирает указанный для него вес в течение заданного времени.</p> <p>Если параметр не задан, то в аналогичной ситуации сервер сразу начинает работу с указанным для него весом.</p> |

#### **Примечание**

Если в апстриме задан только один `server`, `slow_start` не работает и будет игнорироваться.

## state

|                  |                          |
|------------------|--------------------------|
| <i>Синтаксис</i> | <code>state файл;</code> |
| По умолчанию     | —                        |
| <i>Контекст</i>  | upstream                 |

Указывает *файл*, где постоянно хранится список серверов апстрима. Для хранения таких файлов специально создается каталог `/var/lib/angie/state/` (`/var/db/angie/state/` во FreeBSD) с соответствующими правами доступа, и в конфигурации остается добавить лишь имя файла:

```
upstream backend {
 zone backend 1m;
 state /var/lib/angie/state/<ИМЯ ФАЙЛА>;
}
```

Список серверов здесь имеет формат, аналогичный *server*. Содержимое файла изменяется при любом изменении серверов в разделе `/config/stream/upstreams/` через API конфигурации. Файл считывается при запуске Angie или перезагрузке конфигурации.

### Предупреждение

Чтобы использовать директиву `state` в блоке `upstream`, в нем не должно быть директив `server`, но нужна зона разделяемой памяти (*zone*).

## zone

|                  |                                 |
|------------------|---------------------------------|
| <i>Синтаксис</i> | <code>zone имя [размер];</code> |
| По умолчанию     | —                               |
| <i>Контекст</i>  | upstream                        |

Задаёт имя и размер зоны разделяемой памяти, в которой хранятся конфигурация группы и ее рабочее состояние, разделяемые между рабочими процессами. В одной и той же зоне могут быть сразу несколько групп. В этом случае достаточно указать размер только один раз.

## backup\_switch

|                  |                                               |
|------------------|-----------------------------------------------|
| <i>Синтаксис</i> | <code>backup_switch permanent[=время];</code> |
| По умолчанию     | —                                             |
| <i>Контекст</i>  | upstream                                      |

Директива включает возможность начать выбор серверов не с основной группы, а с *активной*, то есть той, где в предыдущий раз был успешно найден сервер. Если найти сервер в активной группе для очередного запроса не удастся, и поиск переходит к резервной группе, то уже эта группа становится активной, и последующие запросы сначала направляются на серверы этой группы.

Если параметр `permanent` определен без значения *времени*, группа остается активной после выбора, и автоматическая перепроверка групп с меньшим уровнем не происходит. Если *время* задано, то

активный статус группы истекает через указанный интервал, и балансировщик снова проверяет группы с меньшим уровнем, возвращаясь к ним, если серверы работают нормально.

Пример:

```
upstream media_backend {
 server primary1.example.com:1935;
 server primary2.example.com:1935;

 server reserve1.example.com:1935 backup;
 server reserve2.example.com:1935 backup;

 backup_switch permanent=2m;
}
```

Если балансировщик переключается с основных серверов на резервную группу, все последующие запросы обрабатываются этой резервной группой в течение 2 минут. По истечении 2 минут балансировщик повторно проверяет основные серверы и снова делает их активными, если они работают нормально.

## feedback

| <i>Синтаксис</i> | <code>feedback</code> <i>переменная</i><br><code>[account=условная_переменная];</code> | <code>[inverse]</code> | <code>[factor=число]</code> |
|------------------|----------------------------------------------------------------------------------------|------------------------|-----------------------------|
| По умолчанию     | —                                                                                      |                        |                             |
| <i>Контекст</i>  | upstream                                                                               |                        |                             |

Задаёт в `upstream` механизм балансировки нагрузки по обратной связи. Он динамически корректирует решения при балансировке, умножая вес каждого проксируемого сервера на среднее значение обратной связи, которое меняется с течением времени в зависимости от значения *переменной* и подчиняется необязательному условию.

Могут быть заданы следующие параметры:

|            |                                                                                                                                                                                                                                                                                                                                                                                          |
|------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| переменная | Переменная, из которой берется значение обратной связи. Она должна представлять собой метрику производительности или состояния; предполагается, что ее передает сервер.<br>Значение оценивается при каждом ответе от сервера и учитывается в скользящем среднем согласно настройкам <b>inverse</b> и <b>factor</b> .                                                                     |
| inverse    | Если параметр задан, значение обратной связи интерпретируется наоборот: более низкие значения указывают на лучшую производительность.                                                                                                                                                                                                                                                    |
| factor     | Коэффициент, по которому значение обратной связи учитывается при расчете среднего. Допустимы целые числа от 0 до 99. По умолчанию — 90.<br>Среднее рассчитывается по формуле <a href="#">экспоненциального сглаживания</a> .<br>Чем больше коэффициент, тем меньше новые значения влияют на среднее; если указать 90, то будет взято 90 % от предыдущего значения и лишь 10 % от нового. |
| account    | Указывает условную переменную, которая контролирует, как соединения учитываются при расчете. Среднее значение обновляется с учетом значения обратной связи, только если условная переменная не равна "" или "0".                                                                                                                                                                         |

**Примечание**

По умолчанию трафик от *активных проверок* не включается в расчет; комбинация переменной *\$upstream\_probe* с **account** позволяет включить и их или даже исключить все остальное.

Пример:

```
upstream backend {
 zone backend 1m;

 feedback $feedback_value factor=80 account=$condition_value;

 server backend1.example.com:1935 weight=1;
 server backend2.example.com:1935 weight=2;
}

map $protocol $feedback_value {
 "TCP" 100;
 "UDP" 75;
 default 10;
}

map $upstream_probe $condition_value {
 "high_priority" "1";
 "low_priority" "0";
 default "1";
}
```

Эта конфигурация категоризирует серверы по уровням обратной связи на основе протоколов, используемых в отдельных сессиях, а также добавляет условие на *\$upstream\_probe*, чтобы учитывать только активную проверку **high\_priority** или обычные клиентские сессии.

## hash

|                  |                                      |
|------------------|--------------------------------------|
| <i>Синтаксис</i> | <code>hash ключ [consistent];</code> |
| По умолчанию     | —                                    |
| <i>Контекст</i>  | upstream                             |

Задаёт метод балансировки нагрузки для группы, при котором соответствие клиента серверу определяется при помощи хэшированного значения ключа. В качестве ключа может использоваться текст, переменные и их комбинации. Следует отметить, что любое добавление или удаление серверов в группе может привести к перераспределению большинства ключей на другие серверы. Метод совместим с библиотекой Perl `Cache::Memcached`.

Пример использования:

```
hash $remote_addr;
```

При использовании доменных имен, разрешающихся в несколько IP-адресов (например, с параметром `resolve`), сервер не сортирует полученные адреса, поэтому их порядок может различаться на разных серверах, что влияет на распределение клиентов. Чтобы обеспечить одинаковое распределение, используйте параметр `consistent`.

Если задан параметр `consistent`, то вместо вышеописанного метода будет использоваться метод консистентного хэширования `ketama`. Метод гарантирует, что при добавлении сервера в группу или его удалении на другие серверы будет перераспределено минимальное число ключей. Применение метода для кэширующих серверов обеспечивает больший процент попаданий в кэш. Метод совместим с библиотекой Perl `Cache::Memcached::Fast` при значении параметра `ketama_points`, равном 160.

## least\_bandwidth

|                  |                                                                            |
|------------------|----------------------------------------------------------------------------|
| <i>Синтаксис</i> | <code>least_bandwidth both   upstream   downstream [factor=number];</code> |
| По умолчанию     | both                                                                       |
| <i>Контекст</i>  | upstream                                                                   |

Модуль балансировки нагрузки `least_bandwidth` в потоковом модуле выбирает проксируемый сервер, который использует наименьший объем пропускной способности, стремясь равномерно распределить сетевой трафик, направляя новые сессии на сервер с наименьшим потреблением пропускной способности. Соответственно, чем меньше трафика, тем сервер считается менее загруженным и тем больше запросов будет на него отправляться.

|                         |                                                                                                                                                                                                |
|-------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>upstream</code>   | Директива учитывает только пропускную способность, используемую <b>от клиента к серверу</b> .                                                                                                  |
| <code>downstream</code> | Директива учитывает только пропускную способность, используемую <b>от сервера к клиенту</b> .                                                                                                  |
| <code>both</code>       | Директива учитывает сумму пропускной способности, используемой вверх (от клиента к серверу) и вниз по потоку (от сервера к клиенту). Режим по умолчанию.                                       |
| <code>factor</code>     | Необязательный коэффициент сглаживания для расчета скользящего среднего, в диапазоне от 0 до 100 (по умолчанию 90). Более высокий коэффициент означает более медленную адаптацию к изменениям. |

Модуль периодически вычисляет среднее использование пропускной способности для каждого проксируемого сервера, используя формулу скользящего среднего для сглаживания колебаний со

временем. Когда начинается новая сессия, он выбирает проксируемый сервер с наименьшим средним использованием пропускной способности, скорректированным с учетом веса сервера, если он указан. Если несколько серверов имеют одинаковое минимальное среднее использование пропускной способности, применяется взвешенный метод round-robin.

### least\_conn

|                  |                          |
|------------------|--------------------------|
| <i>Синтаксис</i> | <code>least_conn;</code> |
| По умолчанию     | —                        |
| <i>Контекст</i>  | upstream                 |

Задаёт для группы метод балансировки нагрузки, при котором соединение передаётся серверу с наименьшим числом активных соединений, с учетом весов серверов. Если подходит сразу несколько серверов, они выбираются циклически (в режиме round-robin) с учетом их весов.

### least\_packets

|              |                                                                         |
|--------------|-------------------------------------------------------------------------|
| Синтаксис    | <code>least_packets both   upstream   downstream [factor=number]</code> |
| По умолчанию | both                                                                    |
| Контекст     | upstream                                                                |

Модуль балансировки нагрузки **least\_packets** в потоковом модуле выбирает проксируемый сервер, обрабатывающий наименьшее количество трафика на основе количества переданных пакетов, стремясь равномерно распределить сетевую нагрузку, направляя новые сеансы на сервер с наименьшей скоростью передачи пакетов.

|                   |                                                                                                                                                                                                |
|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>upstream</b>   | Директива учитывает только пакеты, отправленные <b>от клиента к серверу</b> .                                                                                                                  |
| <b>downstream</b> | Директива учитывает только пакеты, отправленные <b>от сервера к клиенту</b> .                                                                                                                  |
| <b>both</b>       | Директива учитывает сумму пакетов, отправленных вверх (от клиента к серверу) и вниз по потоку (от сервера к клиенту).                                                                          |
| <b>factor</b>     | Необязательный коэффициент сглаживания для расчета скользящего среднего, в диапазоне от 0 до 100 (по умолчанию 90). Более высокий коэффициент означает более медленную адаптацию к изменениям. |

Модуль периодически обновляет среднюю скорость пакетов для каждого проксируемого сервера, используя формулу скользящего среднего для сглаживания колебаний со временем. Когда инициируется новый сеанс, он выбирает проксируемый сервер с наименьшей средней скоростью передачи пакетов, скорректированной с учетом веса сервера. Если несколько серверов имеют одно и то же минимальное значение, применяется взвешенный метод round-robin.

## least\_time

|                  |                                                                                                            |
|------------------|------------------------------------------------------------------------------------------------------------|
| <i>Синтаксис</i> | <code>least_time connect   first_byte   last_byte [factor=number]<br/>[account=condition_variable];</code> |
| По умолчанию     | —                                                                                                          |
| <i>Контекст</i>  | upstream                                                                                                   |

Задаёт для группы метод балансировки нагрузки, при котором вероятность передачи соединения активному серверу обратно пропорциональна среднему времени его ответа; чем оно меньше, тем больше соединений будет получать сервер.

|                         |                                                                    |
|-------------------------|--------------------------------------------------------------------|
| <code>connect</code>    | Директива учитывает среднее время установки соединения.            |
| <code>first_byte</code> | Директива использует среднее время получения первого байта ответа. |
| <code>last_byte</code>  | Директива использует среднее время получения полного ответа.       |

|                      |                                                                                                                                                                                              |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>factor</code>  | Выполняет ту же функцию, что и <code>response_time_factor</code> , и переопределяет его, если параметр задан.                                                                                |
| <code>account</code> | Указывает условную переменную, которая контролирует, какие соединения учитываются при расчете. Среднее значение обновляется, только если условная переменная соединения не равна "" или "0". |

### Примечание

По умолчанию *активные проверки* не включаются в расчет; комбинация переменной `$upstream_probe` с `account` позволяет включить их или даже исключить все остальное.

Текущие значения представлены как `connect_time`, `first_byte_time` и `last_byte_time` в объекте `health` сервера среди *метрик апстрима* в API.

## random

|                  |                            |
|------------------|----------------------------|
| <i>Синтаксис</i> | <code>random [two];</code> |
| По умолчанию     | —                          |
| <i>Контекст</i>  | upstream                   |

Задаёт для группы метод балансировки нагрузки, при котором соединение передается случайно выбранному серверу, с учетом весов серверов.

Если указан необязательный параметр `two`, Angie случайным образом выбирает два сервера, из которых выбирает сервер, используя указанный метод. Методом по умолчанию является `least_conn`, при котором соединение передается на сервер с наименьшим количеством активных соединений.



## response\_time\_factor

|                  |                                          |
|------------------|------------------------------------------|
| <i>Синтаксис</i> | <code>response_time_factor число;</code> |
| По умолчанию     | <code>response_time_factor 90;</code>    |
| <i>Контекст</i>  | <code>upstream</code>                    |

Задаёт для метода балансировки нагрузки *least\_time* коэффициент сглаживания **предыдущего** значения при вычислении среднего времени ответа по формуле **экспоненциально взвешенного скользящего среднего**.

Чем больше указанное *число*, тем меньше новые значения влияют на среднее; если указать 90, то будет взято 90 % от предыдущего значения и лишь 10 % от нового. Допустимые значения — от 0 до 99 включительно.

Текущие результаты вычислений представлены как `connect_time` (время установления соединения), `first_byte_time` (время получения первого байта ответа) и `last_byte_time` (время получения ответа целиком) в объекте `health` сервера среди *метрик апстрима* в API.

### Примечание

При подсчете учитываются только успешные ответы; что считать неуспешным ответом, определяют директивы *proxy\_next\_upstream*.

## sticky

|                  |                                                                                                                                                                                                                                                                                                                           |
|------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Синтаксис</i> | <code>sticky route \$variable...;</code><br><code>sticky learn zone=zone create=\$create_var1... lookup=\$lookup_var1...</code><br><code>[connect] [norefresh] [timeout=time];</code><br><code>sticky learn lookup=\$lookup_var1... remote_action=uri</code><br><code>remote_result=\$remote_var [remote_uri=uri];</code> |
| По умолчанию     | —                                                                                                                                                                                                                                                                                                                         |
| <i>Контекст</i>  | <code>upstream</code>                                                                                                                                                                                                                                                                                                     |

Настраивает привязку клиентских сессий к проксируемым серверам в режиме, заданном первым параметром; для разгрузки серверов, у которых задана директива `sticky`, можно использовать опцию `drain` в блоке *server*.

### Предупреждение

Директива `sticky` должна использоваться после всех директив, задающих тот или иной метод балансировки нагрузки, иначе она не будет работать.

### Режим `route`

Этот режим использует предопределенные идентификаторы маршрутов, которые могут быть встроены в свойства соединения, доступные Angie. Он менее гибок, так как зависит от предопределенных значений, но лучше подходит, если такие идентификаторы уже используются.

Здесь при установлении соединения проксируемый сервер может назначить клиенту маршрут и вернуть его идентификатор способом, известным им обоим. В качестве идентификатора маршрута

должно использоваться значение параметра *sid* директивы *server*. Учтите, что параметр дополнительно хэшируется, если задана директива *sticky\_secret*.

Последующие соединения от клиентов, желающих использовать этот маршрут, должны содержать выданный сервером идентификатор, причем так, чтобы он попал в переменные Angie.

В параметрах директивы указываются переменные для маршрутизации. Чтобы выбрать сервер, куда направляется входящее соединение, используется первая непустая переменная; она затем сравнивается с параметром *sid* директивы *server*. Если выбрать сервер не удастся или выбранный сервер не может принять соединение, то будет выбран другой сервер согласно настроенному методу балансировки.

Здесь Angie ищет идентификатор маршрута в переменной *\$route*, получающей значение на основе *\$ssl\_preread\_server\_name* (обратите внимание, что нужно включить *ssl\_preread*):

```
stream {
 map $ssl_preread_server_name $route {
 a.example.com a;
 b.example.com b;
 default "";
 }

 upstream backend {
 server 127.0.0.1:8081 sid=a;
 server 127.0.0.1:8082 sid=b;

 sticky route $route;
 }

 server {
 listen 127.0.0.1:8080;

 ssl_preread on;

 proxy_pass backend;
 }
}
```

### Режим learn

В этом режиме для привязки клиента к конкретному проксируемому серверу используется динамически генерируемый ключ; этот режим более гибок, так как назначает серверы на ходу, хранит сеансы в зоне разделяемой памяти и поддерживает различные способы передачи идентификаторов сессий.

Здесь сессия создается на основе свойств соединения, идущих от проксируемого сервера. С параметрами *create* и *lookup* перечисляются переменные, указывающие, как создаются новые и ищутся существующие сессии. Оба параметра можно использовать по несколько раз.

Идентификатором сессии служит значение первой непустой переменной, указанной с *create*; например, это может быть *имя проксируемого сервера*.

Сессии хранятся в зоне разделяемой памяти; ее имя и размер задаются параметром *zone*. Если к сессии не было обращений в течение *времени* *timeout*, она удаляется. Значение по умолчанию — 1 час.

По умолчанию Angie продлевает срок действия сессии, обновляя метку времени последнего обращения при каждом ее использовании. Параметр *norefresh* меняет это поведение: сессия истекает

строго по таймауту, даже если используется.

Последующие соединения от клиентов, желающих использовать сессию, должны содержать ее идентификатор. Параметр `lookup` ищет идентификатор сессии в соединении по заданному для него списку переменных, останавливаясь на первой непустой. Если ничего не найдено — запрос считается новым. Значение найденного идентификатора сопоставляется с сессиями в разделяемой памяти. Если выбрать сервер не удастся или выбранный сервер не может обработать соединение, то будет выбран другой сервер согласно настроенному методу балансировки.

Параметр `connect` позволяет создать сессию сразу после установления соединения с проксируемым сервером. Без него сессия создается только после завершения обработки соединения. (В случае UDP-соединений сессии создаются сразу после выбора сервера.)

В примере Angie создает и ищет сессии, используя переменную `$rdp_cookie`:

```
stream {
 upstream backend {
 server 127.0.0.1:3390;
 server 127.0.0.1:3391;

 sticky learn lookup=$rdp_cookie create=$rdp_cookie zone=sessions:1m;
 }

 server {
 listen 127.0.0.1:3389;

 rdp_preread on;

 proxy_pass backend;
 }
}
```

#### Режим `learn` с `remote_action`

Параметры `remote_action` и `remote_result` позволяют динамически назначать идентификаторы сессий и управлять ими с использованием удаленного хранилища сессий.

В отличие от режима `learn` с `zone`, данный режим не кэширует сессии локально и обращается к удаленному хранилищу при каждом соединении.

Параметр `remote_action` должен указывать на `location` в контексте *client*. Параметр `remote_uri` задает URI клиентского HTTP-запроса к указанному `location`. По умолчанию он равен `/`. Значение `remote_uri` может содержать переменные.

Общий принцип работы режима таков: если идентификатор сессии не найден локально, Angie отправляет синхронный подзапрос в некое удаленное хранилище, заданное параметром `remote_action`.

При поступлении нового соединения Angie выполняет следующие действия:

- Сначала извлекается идентификатор сессии из первой непустой переменной в списке `lookup`. Если все переменные пустые, используется обычный алгоритм балансировки нагрузки без привязки.
- Затем Angie отправляет в удаленное хранилище, заданное параметром `remote_action`, синхронный HTTP-подзапрос, который должен содержать в понятном хранилищу виде:
  - идентификатор *сессии* из параметра `lookup` (в конфигурации это переменная `$sticky_sessid`);

- идентификатор предварительно выбранного *сервера*: значение параметра `sid=` из директивы `server`, если оно задано, либо MD5-хэш имени сервера (в конфигурации это переменная `$sticky_sid`).

Переменные `$sticky_sessid` и `$sticky_sid` автоматически экспортируются в HTTP-контекст с префиксом `stream_`: `$stream_sticky_sessid`, `$stream_sticky_sid`. Это позволяет напрямую использовать их в HTTP-директивах, например через HTTP-заголовки с помощью `proxy_set_header`.

- Удаленное хранилище обрабатывает запрос и возвращает HTTP-ответ:

Ответ с кодом 200, 201 или 204 подтверждает выбранный сервер. Удаленное хранилище может одновременно с этим вернуть альтернативный идентификатор сервера в HTTP-заголовке; извлечь его можно через `remote_result`.

При получении от хранилища любого другого HTTP-кода (включая ошибки сети и таймауты) либо несуществующего идентификатора сервера Angie использует изначально выбранный сервер.

Идентификатор сервера извлекается из ответа удаленного хранилища через параметр `remote_result`: в нем можно указывать переменные с префиксом `upstream_http_`, которые создаются Angie автоматически для доступа к заголовкам HTTP-ответов от удаленного хранилища. Например, заголовок `X-Sid: server1` в таком ответе становится доступным в переменной `$upstream_http_x_sid` со значением `server1`.

Ниже показан упрощенный пример конфигурации. Удаленное хранилище возвращает идентификатор сервера в заголовке `X-Sticky-Sid` и таким образом подтверждает или переопределяет выбор Angie:

```
http {

 client {

 location @sticky_client1 {

 # используем переменные из потокового upstream;
 # он добавляет эти переменные в HTTP-контекст с префиксом stream_*
 proxy_set_header X-Sticky-Sessid $stream_sticky_sessid;
 proxy_set_header X-Sticky-Sid $stream_sticky_sid;
 proxy_set_header X-Sticky-Last $msec;
 proxy_pass http://127.0.0.1:8080;

 proxy_cache remote;
 proxy_cache_valid 200 1d;
 proxy_cache_key $scheme$proxy_host$request_uri$stream_sticky_sessid;

 }
 }

 stream {

 upstream u {

 server 127.0.0.1:8081 sid=backend-01;
 server 127.0.0.1:8082 sid=backend-02;

 sticky learn lookup=$remote_addr # переменная stream
 remote_action=@sticky_client1 # location из блока client
 remote_result=$upstream_http_x_sticky_sid # HTTP-переменная
 remote_uri=/foo; # по умолчанию - /

 }

 }

}
```

```
server {
 listen 127.0.0.1:8080;
 proxy_pass u;
}
```

Здесь при следующем ответе от удаленного хранилища:

```
HTTP/1.1 200 OK
...
X-Sticky-Sid: backend-01
X-Session-Info: active
```

Становятся доступными две переменные:

- `$upstream_http_x_sticky_sid`, со значением `backend-01`;
- `$upstream_http_x_session_info`, со значением `active`.

Так как переменная `$upstream_http_x_sticky_sid` указана в параметре `remote_result`, то ее значение будет использовано для выбора сервера с `sid=backend-01`.

Директива `sticky` учитывает состояние серверов в *upstream*:

- Серверы, помеченные как `down` или временно недоступные из-за сбоев, исключаются из выбора.
- Серверы, которые достигли максимального количества соединений (при использовании `max_conns`), временно пропускаются.
- Серверы с опцией `drain` (PRO) могут быть выбраны для создания новых сессий в режиме `sticky` при совпадении идентификаторов.
- Если ранее недоступный сервер восстанавливается, `sticky` автоматически возобновляет его использование.

Поведение `sticky` можно дополнительно настроить директивами *sticky\_secret* и *sticky\_strict*. Если в ходе работы `sticky` выбрать сервер не удастся или он недоступен, запрос будет обработан согласно выбранному методу балансировки нагрузки, если только не включена директива `sticky_strict`. В режиме `sticky_strict on`; запрос отклоняется с ошибкой.

Зоны разделяемой памяти, указываемые в параметре `zone` директивы `sticky`, не могут использоваться совместно различными *upstream*; каждая группа должна использовать свою собственную зону.

## sticky\_secret

|              |                                    |
|--------------|------------------------------------|
| Синтаксис    | <code>sticky_secret строка;</code> |
| По умолчанию | —                                  |
| Контекст     | <code>upstream</code>              |

Добавляет *строку* как соль в функцию MD5-хэширования для директивы *sticky* в режиме `route`. *Строка* может содержать переменные, например `$remote_addr`:

```
upstream backend {
 server 127.0.0.1:8081 sid=a;
 server 127.0.0.1:8082 sid=b;
```

```
sticky route $route;
sticky_secret my_secret.$remote_addr;
}
```

Соль добавляется после хэшируемого значения; чтобы независимо проверить механизм хэширования:

```
$ echo -n "<VALUE><SALT>" | md5sum
```

## sticky\_strict

|                  |                                      |
|------------------|--------------------------------------|
| <i>Синтаксис</i> | <code>sticky_strict on   off;</code> |
| По умолчанию     | <code>sticky_strict off;</code>      |
| <i>Контекст</i>  | <code>upstream</code>                |

При включении Angie будет возвращать клиенту ошибку соединения, если желаемый сервер недоступен, вместо использования любого другого доступного сервера, как это происходит, когда в группе нет доступных серверов.

## Встроенные переменные

Модуль `stream_upstream` поддерживает следующие встроенные переменные:

`$sticky_sessid`

Используется с `remote_action` в *sticky*; хранит начальный идентификатор сессии, взятый из `lookup`.

`$sticky_sid`

Используется с `remote_action` в *sticky*; хранит идентификатор сервера, предварительно связанный с сессией.

В `sticky_sid` содержится значение параметра `sid=` из блока *upstream* директивы `server`, если оно задано, либо MD5-хэш имени сервера.

`$upstream_addr`

хранит IP-адрес и порт или путь к UNIX-сокету сервера группы. Если при проксировании были сделаны обращения к нескольким серверам, то их адреса разделяются запятой, например:

```
192.168.1.1:1935, 192.168.1.2:1935, unix:/tmp/sock"
```

Если сервер не может быть выбран, то переменная хранит *имя группы серверов*.

#### `$upstream_bytes_received`

число байт, полученных от сервера группы. Значения нескольких соединений разделяются запятыми и двоеточиями подобно адресам в переменной `$upstream_addr`.

#### `$upstream_bytes_sent`

число байт, переданных на сервер группы. Значения нескольких соединений разделяются запятыми и двоеточиями подобно адресам в переменной `$upstream_addr`.

#### `$upstream_connect_time`

хранит время, затраченное на установление соединения с сервером группы; время хранится в секундах с точностью до миллисекунд. Времена нескольких соединений разделяются запятыми и двоеточиями подобно адресам в переменной `$upstream_addr`.

#### `$upstream_first_byte_time`

время получения первого байта данных; время хранится в секундах с точностью до миллисекунд. Времена нескольких соединений разделяются запятыми подобно адресам в переменной `$upstream_addr`.

#### `$upstream_session_time`

длительность сессии в секундах с точностью до миллисекунд. Времена нескольких соединений разделяются запятыми подобно адресам в переменной `$upstream_addr`.

#### `$upstream_sticky_status`

Статус соединений с привязкой.

|      |                                                                                   |
|------|-----------------------------------------------------------------------------------|
| ""   | Соединение направлено в группу серверов, где привязка не используется.            |
| NEW  | Соединение не содержит информации о привязке к серверу.                           |
| HIT  | Соединение с привязкой направлено на желаемый сервер.                             |
| MISS | Соединение с привязкой направлено на сервер, выбранный по алгоритму балансировки. |

Статусы из нескольких соединений разделяются запятыми и двоеточиями аналогично адресам в переменной `$upstream_addr`.

### Upstream Probe

Реализует активные проверки работоспособности (health probes) для `Upstream`.

## Пример конфигурации

```
server {
 listen ...;

 # ...
 proxy_pass backend;
 upstream_probe_timeout 1s;

 upstream_probe backend_probe
 port=12345
 interval=5s
 test=$good
 essential
 fails=3
 passes=3
 max_response=512k
 mode=onfail
 "send=data:GET / HTTP/1.0\r\n\r\n";
}
```

### Примечание

Согласно спецификации RFC 2616 (HTTP/1.1) и RFC 9110 (HTTP Semantics), заголовки HTTP должны разделяться последовательностью CRLF (\r\n), а не просто \n.

## Директивы

### upstream\_probe

|                  |                                                                                                                                                                                                                                         |
|------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Синтаксис</i> | <code>upstream_probe имя [port=число] [interval=время] [test=условие] [ping [ping_timeout=время]] [essential [persistent]] [fails=число] [passes=число] [max_response=размер] [mode=always   idle   onfail] [udp] [send=строка];</code> |
| По умолчанию     | —                                                                                                                                                                                                                                       |
| <i>Контекст</i>  | server                                                                                                                                                                                                                                  |

Задаёт активную проверку работоспособности серверов *апстрима*, указанного в директиве *proxy\_pass* в том же контексте **server**, где находится директива **upstream\_probe**.

Сервер проходит проверку, если запрос к нему успешно выполняется с учетом всех параметров самой директивы **upstream\_probe** и всех параметров, влияющих на использование апстримов тем контекстом **server**, где она задана, в том числе директивы *proxy\_next\_upstream*.

Чтобы использовать проверки, в апстриме необходима зона разделяемой памяти (*zone*). Для одного апстрима можно определить несколько проверок.

Могут быть заданы следующие параметры:



|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| имя                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | Обязательное имя проверки.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| port                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | Альтернативный порт для запроса.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| interval                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | <i>Интервал</i> между проверками. По умолчанию — 5s.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| ping                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | Использовать проверку методом ICMP Echo вместо TCP- или UDP-пакета. При выборе этого режима Angie ADC отправляет эхо-запрос (ICMP Echo Request) на IP-адрес сервера и ожидает ответ (Echo Reply). Проверка осуществляется по IP-адресу, а номер порта из <b>upstream</b> игнорируется.                                                                                                                                                                                                                                                                                                                                                                  |
| <div>  <b>Примечание</b> </div> <p>Параметры <b>test</b>, <b>port</b>, <b>udp</b> и <b>send</b> несовместимы с <b>ping</b> и игнорируются. Для отправки ICMP-пакетов необходимы соответствующие права: на Linux — запуск процессов от имени группы, указанной в <code>/proc/sys/net/ipv4/ping_group_range</code>; на FreeBSD — права суперпользователя или наличие <i>capability</i> на использование raw-сокеты. При использовании ICMP-проверок переменные, основанные на данных ответа сервера или отправленных пакетах, всегда пусты; результат определяется только фактом получения ICMP Echo Reply.</p> |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| ping_timeout                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | <i>Время</i> ожидания ICMP-ответа; по умолчанию — 1s. Используется только вместе с <b>ping</b> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| test                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | Проверяемое при запросе условие; задается строкой из переменных. Если результат подстановки всех переменных — "" или "0", проверка не пройдена.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| essential                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | Если параметр задан, то изначально состояние сервера подлежит уточнению и клиентские запросы не передаются ему, пока проверка не будет пройдена.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| persistent                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | Установка этого параметра требует сначала включить <b>essential</b> ; серверы с <b>persistent</b> , работавшие до <i>перезагрузки конфигурации</i> , начинают получать запросы без необходимости сначала пройти эту проверку.                                                                                                                                                                                                                                                                                                                                                                                                                           |
| fails                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | Число последовательных неуспешных запросов, при котором проверка считает сервер неработающим. По умолчанию — 1.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| passes                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | Число последовательных успешных запросов, при котором проверка считает сервер работающим. По умолчанию — 1.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| max_response                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | Максимальный объем памяти для ответа. Если задано нулевое <i>значение</i> , ожидание ответа отключается. По умолчанию — 256k.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| mode                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | Режим проверки в зависимости от работоспособности серверов: <ul style="list-style-type: none"> <li>• <b>always</b> — серверы проверяются независимо от состояния;</li> <li>• <b>idle</b> — проверяются неработающие серверы, а также серверы, где с последнего клиентского запроса прошло время <b>interval</b>.</li> <li>• <b>onfail</b> — проверяются серверы только в неработающем состоянии.</li> </ul> По умолчанию — <b>always</b> .                                                                                                                                                                                                              |
| udp                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | Если параметр указан, используется протокол UDP. По умолчанию для проверок используется TCP.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| send                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | Отправляемые для проверки данные: встроенные данные с префиксом <b>data:</b> или путь к файлу (абсолютный или относительно <code>/usr/local/angie/</code> ). При использовании файла: <ul style="list-style-type: none"> <li>• <i>Рабочий процесс</i> открывает и читает файл при каждом обращении; содержимое не сохраняется в памяти.</li> <li>• Перезагрузка конфигурации при изменении файла не требуется; новое содержимое будет прочитано при следующем обращении.</li> <li>• Необходимые права доступа: 644 для файла, 755 для директории.</li> <li>• Обновляйте файлы командой перемещения (<b>mv</b>), а не прямым редактированием.</li> </ul> |

Пример:

```
upstream backend {
 zone backend 1m;

 server a.example.com;
 server b.example.com;
}

map $upstream_probe_response $good {
 ~200 "1";
 default "";
}

server {
 listen ...;

 # ...
 proxy_pass backend;
 upstream_probe_timeout 1s;

 upstream_probe backend_probe
 port=12345
 interval=5s
 test=$good
 essential
 persistent
 fails=3
 passes=3
 max_response=512k
 mode=onfail
 "send=data:GET / HTTP/1.0\r\n\r\n";
}
```

Детали работы:

- Изначально сервер не получает клиентские запросы, пока не пройдет *все* заданные для него проверки с параметром **essential** (пропуская помеченные как **persistent**, если конфигурация перезагружена и до этого сервер считался работающим). Если таких проверок нет, сервер считается работающим.
- Сервер считается неработающим и не получает клиентские запросы, если *какая-либо* заданная для него проверка достигает своего порога **fails** или сам сервер достигает порога *max\_fails*.
- Чтобы неработающий сервер снова мог считаться работающим, *все* заданные для него проверки должны достичь своего порога **passes**; после этого учитывается порог *max\_fails*.

### upstream\_probe\_timeout

|                  |                                            |
|------------------|--------------------------------------------|
| <i>Синтаксис</i> | <code>upstream_probe_timeout время;</code> |
| По умолчанию     | <code>upstream_probe_timeout 50s;</code>   |
| <i>Контекст</i>  | <code>server</code>                        |

Задаёт максимальное *время* бездействия установленного с сервером соединения для проверок, настроенных с помощью директивы *upstream\_probe*; при превышении этого предела соединение будет закрыто.

## Встроенные переменные

Модуль `stream_upstream` поддерживает следующие встроенные переменные:

`$upstream_probe`

Имя активной сейчас проверки `upstream_probe`.

`$upstream_probe_response`

Содержимое ответа, полученного в ходе активной проверки `upstream_probe`.

Базовый потоковый модуль реализует основную функциональность для обработки TCP- и UDP-соединений: это определение серверных блоков, маршрутизация трафика, настройка проксирования, поддержка SSL/TLS и управление подключениями для потоковых сервисов, таких как базы данных, DNS и другие протоколы, работающие на основе TCP и UDP.

Остальные модули этого раздела расширяют эту функциональность, позволяя гибко настраивать и оптимизировать работу потокового сервера под различные сценарии и требования.

При сборке из исходного кода модуль не собирается по умолчанию; его необходимо включить с помощью параметра сборки `--with-stream`. В пакетах и образах из наших репозиториев модуль включен в сборку.

## Пример конфигурации

```
worker_processes auto;

error_log /var/log/angie/error.log info;

events {
 worker_connections 1024;
}

stream {
 upstream backend {
 hash $remote_addr consistent;

 server backend1.example.com:12345 weight=5;
 server 127.0.0.1:12345 max_fails=3 fail_timeout=30s;
 server unix:/tmp/backend3;
 }

 upstream dns {
 server 192.168.0.1:53535;
 server dns.example.com:53;
 }

 server {
 listen 12345;
 proxy_connect_timeout 1s;
 proxy_timeout 3s;
 proxy_pass backend;
 }

 server {
```

```
listen 127.0.0.1:53 udp reuseport;
proxy_timeout 20s;
proxy_pass dns;
}

server {
 listen [::1]:12345;
 proxy_pass unix:/tmp/stream.socket;
}
}
```

## Директивы

### listen

|                  |                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Синтаксис</i> | <code>listen</code> <i>адрес[:порт]</i> [ <i>ssl</i> ] [ <i>udp</i> ] [ <i>proxy_protocol</i> ] [ <i>setfib=число</i> ] [ <i>fastopen=число</i> ] [ <i>backlog=число</i> ] [ <i>rcvbuf=размер</i> ] [ <i>sndbuf=размер</i> ] [ <i>accept_filter=фильтр</i> ] [ <i>deferred</i> ] [ <i>bind</i> ] [ <i>ipv6only=on   off</i> ] [ <i>reuseport</i> ] [ <i>so_keepalive=on off</i> ][ <i>keepidle</i> ]:[ <i>keepintvl</i> ]:[ <i>keepcnt</i> ]; |
| По умолчанию     | —                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| <i>Контекст</i>  | server                                                                                                                                                                                                                                                                                                                                                                                                                                        |

Задаёт *адрес* и *порт* для сокета, на котором сервер будет принимать соединения. Можно указать только *порт*, и тогда Angie будет слушать на всех доступных IPv4-интерфейсах (и IPv6, если он включен). Кроме того, *адрес* может быть именем хоста, например:

```
listen 127.0.0.1:12345;
listen *:12345;
listen 12345; # то же, что и *:12345
listen localhost:12345;
```

IPv6-адреса задаются в квадратных скобках:

```
listen [::1]:12345;
listen [::]:12345;
```

UNIX-сокеты задаются префиксом `unix`:

```
listen unix:/var/run/angie.sock;
```

Диапазоны портов задаются при помощи указания первого и последнего порта через дефис:

```
listen 127.0.0.1:12345-12399;
listen 12345-12399;
```

#### Примечание

Разные серверы должны слушать на разных парах *адрес:порт*.

|                             |                                                                                                                                                                                     |
|-----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>ssl</code>            | указывает на то, что все соединения, принимаемые на данном слушающем сокете, должны работать в режиме SSL.                                                                          |
| <code>udp</code>            | конфигурирует слушающий сокет для работы с датаграммами. Для обработки пакетов с одного адреса и порта в рамках одной сессии необходимо также указывать параметр <i>reuseport</i> . |
| <code>proxy_protocol</code> | указывает на то, что все соединения, принимаемые на данном порту, должны использовать протокол PROXY.                                                                               |

В директиве `listen` можно также указать несколько дополнительных параметров, специфичных для связанных с сокетами системных вызовов.

|                                                                                                                                                                                                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>setfib=число</code>                                                                                                                                                                                 | устанавливает связанную таблицу маршрутизации, FIB (параметр <code>SO_SETFIB</code> ) для слушающего сокета. Пока это работает только на FreeBSD.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| <code>fastopen=число</code>                                                                                                                                                                               | включает "TCP Fast Open" для слушающего сокета и <b>ограничивает</b> максимальную длину очереди соединений, которые еще не завершили процесс трехстороннего рукопожатия.<br>Не включайте "TCP Fast Open", не убедившись, что сервер может адекватно обрабатывать <b>многократное получение</b> одного и того же SYN-пакета с данными.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| <code>backlog=число</code>                                                                                                                                                                                | задает параметр <i>backlog</i> в вызове <i>listen()</i> , который ограничивает максимальный размер очереди ожидающих приема соединений. По умолчанию <i>backlog</i> устанавливается равным -1 для FreeBSD, DragonFly BSD и macOS, и 511 для других платформ.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| <code>rcvbuf=размер</code>                                                                                                                                                                                | задает размер буфера приема (параметр <code>SO_RCVBUF</code> ) для слушающего сокета.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| <code>sndbuf=размер</code>                                                                                                                                                                                | задает размер буфера передачи (параметр <code>SO_SNDBUF</code> ) для слушающего сокета.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| <code>accept_filter=фи</code>                                                                                                                                                                             | задает имя принимающего фильтра (параметр <code>SO_ACCEPTFILTER</code> ) для слушающего сокета, который фильтрует входящие соединения перед их передачей в <code>accept()</code> . Работает только на FreeBSD и NetBSD 5.0+. Допустимые значения: <code>dataready</code> и <code>httpready</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <code>deferred</code>                                                                                                                                                                                     | указывает использовать отложенный <code>accept()</code> (параметр <code>TCP_DEFER_ACCEPT</code> ) на Linux.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <code>bind</code>                                                                                                                                                                                         | указывает, что для данного слушающего сокета нужно делать <i>bind()</i> отдельно. Это нужно потому, что если описаны несколько директив <code>listen</code> с одинаковым портом, но разными адресами, и одна из директив <code>listen</code> слушает на всех адресах для данного <i>порта</i> ( <i>*:порт</i> ), то Angie сделает <i>bind()</i> только на <i>*:порт</i> . Необходимо заметить, что в этом случае для определения адреса, на который пришло соединение, делается системный вызов <i>getsockname()</i> . Если же используются параметры <i>setfib</i> , <i>fastopen</i> , <i>backlog</i> , <i>rcvbuf</i> , <i>sndbuf</i> , <i>accept_filter</i> , <i>deferred</i> , <i>ipv6only</i> , <i>reuseport</i> или <i>so_keepalive</i> , то для данной пары <i>адрес:порт</i> всегда делается отдельный вызов <i>bind()</i> . |
| <code>ipv6only=on   off</code>                                                                                                                                                                            | определяет (через параметр сокета <code>IPV6_V6ONLY</code> ), будет ли слушающий на wildcard-адресе <code>:::</code> IPv6-сокеты принимать только IPv6-соединения, или же одновременно IPv6- и IPv4-соединения. По умолчанию параметр включен. Установить его можно только один раз на старте.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <code>reuseport</code>                                                                                                                                                                                    | указывает, что нужно создавать отдельный слушающий сокет для каждого рабочего процесса (через параметр сокета <code>SO_REUSEPORT</code> для Linux 3.9+ и DragonFly BSD или <code>SO_REUSEPORT_LB</code> для FreeBSD 12+), позволяя ядру распределять входящие соединения между рабочими процессами. В настоящий момент это работает только на Linux 3.9+, DragonFly BSD и FreeBSD 12+.                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| <div>  <b>Предупреждение</b><br/> Ненадлежащее использование параметра <i>reuseport</i> может быть небезопасно. </div> |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| <code>multipath</code>                                                                                                                                                                                    | включает прием соединений по протоколу <b>Multipath TCP</b> (MPTCP), поддерживаемому в ядре Linux с версии 5.6. Параметр <b>несовместим</b> с <code>udp</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |

`so_keepalive=on | off | [keepidle]:[keepintvl]:[keepcnt]`

Конфигурирует для слушающего сокета поведение "TCP keepalive".

|                  |                                                                                   |
|------------------|-----------------------------------------------------------------------------------|
| <code>''</code>  | если параметр опущен, для сокета будут действовать настройки операционной системы |
| <code>on</code>  | для сокета включается параметр <code>SO_KEEPALIVE</code>                          |
| <code>off</code> | для сокета параметр <code>SO_KEEPALIVE</code> выключается                         |

Некоторые операционные системы поддерживают настройку параметров "TCP keepalive" на уровне сокета посредством параметров `TCP_KEEPIDLE`, `TCP_KEEPINTVL` и `TCP_KEEPCNT`. На таких системах их можно сконфигурировать с помощью параметров `keepidle`, `keepintvl` и `keepcnt`. Один или два параметра могут быть опущены, в таком случае для соответствующего параметра сокета будут действовать стандартные системные настройки.

Например,

```
so_keepalive=30m:10
```

установит таймаут бездействия (`TCP_KEEPIDLE`) в 30 минут, для интервала проб (`TCP_KEEPINTVL`) будет действовать стандартная системная настройка, а счетчик проб (`TCP_KEEPCNT`) будет равен 10.

### `preread_buffer_size`

|                  |                                                  |
|------------------|--------------------------------------------------|
| <i>Синтаксис</i> | <code>preread_buffer_size</code> <i>размер</i> ; |
| По умолчанию     | <code>preread_buffer_size</code> 16k;            |
| <i>Контекст</i>  | stream, server                                   |

Задаёт размер буфера *предварительного чтения*.

### `preread_timeout`

|                  |                                             |
|------------------|---------------------------------------------|
| <i>Синтаксис</i> | <code>preread_timeout</code> <i>время</i> ; |
| По умолчанию     | <code>preread_timeout</code> 30s;           |
| <i>Контекст</i>  | stream, server                              |

Задаёт время фазы *предварительного чтения*.

### `proxy_protocol_timeout`

|                  |                                                    |
|------------------|----------------------------------------------------|
| <i>Синтаксис</i> | <code>proxy_protocol_timeout</code> <i>время</i> ; |
| По умолчанию     | <code>proxy_protocol_timeout</code> 30s;           |
| <i>Контекст</i>  | stream, server                                     |

Задаёт время для завершения операции чтения заголовка протокола PROXY. Если по истечении этого времени заголовок полностью не получен, соединение закрывается.

## resolver

|                  |                                                                                                   |
|------------------|---------------------------------------------------------------------------------------------------|
| <i>Синтаксис</i> | <code>resolver адрес ... [valid=время] [ipv4=on   off] [ipv6=on   off] [status_zone=зона];</code> |
| По умолчанию     | —                                                                                                 |
| <i>Контекст</i>  | stream, server, upstream                                                                          |

Задаёт серверы DNS, используемые для преобразования имен вышестоящих серверов в адреса, например:

```
resolver 127.0.0.53 [::1]:5353;
```

Адрес может быть указан в виде доменного имени или IP-адреса, и необязательного порта. Если порт не указан, используется порт 53. Серверы DNS опрашиваются циклически.

### Примечание

Значение директивы наследуется вложенными блоками и может быть переопределено в них при необходимости. В пределах одного блока допустимо указывать директиву только один раз. Если она повторяется, действует последнее определение.

По умолчанию Angie кэширует ответы, используя значение TTL из ответа DNS. Если директива **resolver** не указана и не выполняются динамические DNS-запросы (например, при использовании фиксированных имен в *Proxy* без переменных), указание резолвера не требуется: имена будут разрешены при запуске с помощью системного резолвера. Необязательный параметр **valid** позволяет это переопределить:

|              |                                                                           |
|--------------|---------------------------------------------------------------------------|
| <b>valid</b> | необязательный параметр, позволяет переопределить срок кэширования ответа |
|--------------|---------------------------------------------------------------------------|

```
resolver 127.0.0.53 [::1]:5353 valid=30s;
```

По умолчанию Angie будет искать как IPv4-, так и IPv6-адреса при преобразовании имен в адреса.

|                 |                              |
|-----------------|------------------------------|
| <b>ipv4=off</b> | запрещает поиск IPv4-адресов |
| <b>ipv6=off</b> | запрещает поиск IPv6-адресов |

|                    |                                                                                                                                               |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| <b>status_zone</b> | необязательный параметр; включает сбор метрик запросов и ответов DNS-сервера ( <code>/status/resolvers/&lt;зона&gt;</code> ) в указанной зоне |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|

### Совет

Для предотвращения DNS-спуфинга рекомендуется использовать DNS-серверы в защищенной доверенной локальной сети.

### Совет



При запуске в Docker используйте соответствующий внутренний адрес DNS-сервера, например 127.0.0.11.

## resolver\_timeout

|                  |                                      |
|------------------|--------------------------------------|
| <i>Синтаксис</i> | <code>resolver_timeout время;</code> |
| По умолчанию     | <code>resolver_timeout 30s;</code>   |
| <i>Контекст</i>  | stream, server, upstream             |

Задаёт таймаут для преобразования имени в адрес, например:

```
resolver_timeout 5s;
```

## server

|                  |                             |
|------------------|-----------------------------|
| <i>Синтаксис</i> | <code>server { ... }</code> |
| По умолчанию     | —                           |
| <i>Контекст</i>  | stream                      |

Задаёт конфигурацию для сервера.

## server\_name

|                  |                                   |
|------------------|-----------------------------------|
| <i>Синтаксис</i> | <code>server_name имя ...;</code> |
| По умолчанию     | <code>server_name "";</code>      |
| <i>Контекст</i>  | server                            |

Задаёт имена виртуального сервера.

### Предупреждение

В модуле `stream` директива `server_name` основана на Server Name Indication (SNI) и работает только с TLS-соединениями. Для использования необходимо настроить TLS-терминацию или включить *TLS preread* в соответствующем блоке `server`.

Пример конфигурации:

```
server {
 listen 443 ssl;
 server_name example.com www.example.com;
 ssl_certificate /etc/angie/cert.pem;
 ssl_certificate_key /etc/angie/key.pem;
}
```

Первое указанное имя становится основным именем сервера.

Имена серверов могут включать звездочку (\*), заменяющую первую или последнюю часть имени:

```
server {
 server_name example.com *.example.com www.example.*;
}
```

Такие имена называются подстановочными (wildcard).

Также можно использовать регулярные выражения в именах серверов, предваряя имя тильдой (~):

```
server {
 server_name www.example.com ~www\d+\.example\.com$;
}
```

Регулярные выражения могут включать захватываемые группы, которые можно использовать в других директивах:

```
server {
 server_name ~^(www\.)?(.+)$;

 proxy_pass www.$2:12345;
}
```

Именованные захваты в регулярных выражениях создают переменные, которые можно использовать в других директивах:

```
server {
 server_name ~^(www\.)?(?<domain>.+)$;

 proxy_pass www.$domain:12345;
}
```

Если параметр директивы установлен на \$hostname, будет вставлено имя хоста машины.

При поиске виртуального сервера по имени, если имя совпадает с более чем одним из указанных вариантов (например, совпадают и шаблонное имя, и регулярное выражение), будет выбрано первое совпавшее имя в следующем порядке приоритета:

- Точное имя
- Самое длинное шаблонное имя, начинающееся с звездочки, например, \*.example.com
- Самое длинное шаблонное имя, заканчивающееся звездочкой, например, mail.\*
- Первое совпавшее регулярное выражение (в порядке появления в конфигурационном файле)

## server\_names\_hash\_bucket\_size

|                  |                                               |
|------------------|-----------------------------------------------|
| <i>Синтаксис</i> | server_names_hash_bucket_size <i>размер</i> ; |
| По умолчанию     | server_names_hash_bucket_size 32 64 128;      |
| <i>Контекст</i>  | stream                                        |

Задаёт размер корзины для хэш-таблиц имен серверов. Значение по умолчанию зависит от размера кэш-линии процессора.

## server\_names\_hash\_max\_size

|                  |                                                         |
|------------------|---------------------------------------------------------|
| <i>Синтаксис</i> | <code>server_names_hash_max_size</code> <i>размер</i> ; |
| По умолчанию     | <code>server_names_hash_max_size 512</code> ;           |
| <i>Контекст</i>  | <code>stream</code>                                     |

Задаёт максимальный размер хэш-таблиц имен серверов.

## status\_zone

|                  |                                                                                     |
|------------------|-------------------------------------------------------------------------------------|
| <i>Синтаксис</i> | <code>status_zone</code> <i>зона</i>   <i>ключ</i> <code>zone=зона[:число]</code> ; |
| По умолчанию     | —                                                                                   |
| <i>Контекст</i>  | <code>server</code>                                                                 |

Выделяет зону разделяемой памяти для сбора метрик `/status/stream/server_zones/<зона>`.

Несколько контекстов `server` могут совместно использовать одну и ту же зону для сбора данных.

Синтаксис с одним значением *зоны* объединяет все метрики для текущего контекста в одну зону разделяемой памяти:

```
server {
 listen 80;
 server_name *.example.com;

 status_zone single;
 # ...
}
```

Альтернативный синтаксис позволяет задавать следующие параметры:

|                               |                                                                                                                                                                                                                                                   |
|-------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>значение</i>               | Строка с переменными, значение которой определяет группировку подключений в зоне. Все подключения, дающие одинаковые значения после подстановки, объединяются в одну группу. Если подстановка возвращает пустое значение, метрики не обновляются. |
| <i>зона</i>                   | Имя зоны разделяемой памяти.                                                                                                                                                                                                                      |
| <i>число</i> (необязательный) | Максимальное количество отдельных групп для сбора метрик. Если новые значения <i>ключа</i> превышают этот лимит, они объединяются в группу <i>zone</i> . Значение по умолчанию — 1.                                                               |

В следующем примере все соединения с одинаковым значением `$server_addr` группируются в `host_zone`. Метрики собираются отдельно для каждого уникального значения `$server_addr` до тех пор, пока количество групп метрик не достигнет 10. После этого любые новые значения `$server_addr` будут добавляться в группу `server_zone`:

```
stream {
 upstream backend {
 server 192.168.0.1:3306;
 server 192.168.0.2:3306;
 # ...
 }
}
```

```

 }

 server {

 listen 3306;
 proxy_pass backend;

 status_zone $server_addr zone=server_zone:10;
 }
}

```

Результирующие метрики разделяются по отдельным серверам в выводе API.

### stream

|                  |                             |
|------------------|-----------------------------|
| <i>Синтаксис</i> | <code>stream { ... }</code> |
| По умолчанию     | —                           |
| <i>Контекст</i>  | main                        |

Предоставляет контекст конфигурационного файла, в котором указываются директивы stream-сервера.

### tcp\_nodelay

|                  |                                    |
|------------------|------------------------------------|
| <i>Синтаксис</i> | <code>tcp_nodelay on   off;</code> |
| По умолчанию     | <code>tcp_nodelay on;</code>       |
| <i>Контекст</i>  | stream, server                     |

Разрешает или запрещает использование параметра `TCP_NODELAY`. Параметр включается как для клиентских соединений, так и для соединений с проксируемыми серверами.

### variables\_hash\_bucket\_size

|                  |                                                 |
|------------------|-------------------------------------------------|
| <i>Синтаксис</i> | <code>variables_hash_bucket_size размер;</code> |
| По умолчанию     | <code>variables_hash_bucket_size 64;</code>     |
| <i>Контекст</i>  | stream                                          |

Задаёт размер корзины в хэш-таблице переменных. Подробнее настройка хэш-таблиц обсуждается *отдельно*.

## variables\_hash\_max\_size

|                  |                                                      |
|------------------|------------------------------------------------------|
| <i>Синтаксис</i> | <code>variables_hash_max_size</code> <i>размер</i> ; |
| По умолчанию     | <code>variables_hash_max_size 1024</code> ;          |
| <i>Контекст</i>  | <code>stream</code>                                  |

Задаёт максимальный размер хэш-таблиц переменных. Подробнее настройка хэш-таблиц обсуждается *отдельно*.

## Встроенные переменные

Базовый потоковый модуль поддерживает встроенные переменные:

`$angie_version`

версия Angie

`$binary_remote_addr`

адрес клиента в бинарном виде, длина значения всегда 4 байта для IPv4-адресов или 16 байт для IPv6-адресов

`$bytes_received`

число байт, полученных от клиента

`$bytes_sent`

число байт, переданных клиенту

`$connection`

порядковый номер соединения

`$hostname`

имя хоста

`$msec`

текущее время в секундах с точностью до миллисекунд

`$pid`

номер (PID) рабочего процесса

`$protocol`

протокол, используемый для работы с клиентом: *TCP* или *UDP*

`$proxy_protocol_addr`

адрес клиента, полученный из заголовка протокола PROXY. Протокол PROXY должен быть предварительно включен при помощи установки параметра `proxy_protocol` в директиве *listen*.

`$proxy_protocol_port`

порт клиента, полученный из заголовка протокола PROXY. Протокол PROXY должен быть предварительно включен при помощи установки параметра `proxy_protocol` в директиве *listen*.

`$proxy_protocol_server_addr`

адрес сервера, полученный из заголовка протокола PROXY. Протокол PROXY должен быть предварительно включен при помощи установки параметра `proxy_protocol` в директиве *listen*.

`$proxy_protocol_server_port`

порт сервера, полученный из заголовка протокола PROXY. Протокол PROXY должен быть предварительно включен при помощи установки параметра `proxy_protocol` в директиве *listen*.

`$proxy_protocol_tlv_<имя>`

TLV, полученный из заголовка протокола PROXY. *Имя* может быть именем типа TLV или его числовым значением. В последнем случае значение задается в шестнадцатеричном виде и должно начинаться с *0x*:

```
$proxy_protocol_tlv_alpn
$proxy_protocol_tlv_0x01
```

SSL TLV могут также быть доступны как по имени типа TLV, так и по его числовому значению, оба должны начинаться с `ssl_`:

```
$proxy_protocol_tlv_ssl_version
$proxy_protocol_tlv_ssl_0x21
```

Поддерживаются следующие имена типов TLV:

- `alpn` (0x01) - протокол более высокого уровня, используемый поверх соединения
- `authority` (0x02) - значение имени хоста, передаваемое клиентом

- `unique_id` (0x05) - уникальный идентификатор соединения
- `netns` (0x30) - имя пространства имен
- `ssl` (0x20) - структура SSL TLV в бинарном виде

Поддерживаются следующие имена типов SSL TLV:

- `ssl_version` (0x21) - версия SSL, используемая в клиентском соединении
- `ssl_cn` (0x22) - Common Name сертификата
- `ssl_cipher` (0x23) - имя используемого шифра
- `ssl_sig_alg` (0x24) - алгоритм, используемый для подписи сертификата
- `ssl_key_alg` (0x25) - алгоритм публичного ключа

Также поддерживается следующее специальное имя типа SSL TLV:

- `ssl_verify` - результат проверки клиентского сертификата: 0, если клиент предоставил сертификат и он был успешно верифицирован, либо ненулевое значение

Протокол PROXY должен быть предварительно включен при помощи установки параметра `proxy_protocol` в директиве *listen*.

`$remote_addr`

адрес клиента

`$remote_port`

порт клиента

`$server_addr`

адрес сервера, принявшего соединение. Получение значения этой переменной обычно требует одного системного вызова. Чтобы избежать системного вызова, в директивах *listen* следует указывать адреса и использовать параметр `bind`.

`$server_port`

порт сервера, принявшего соединение

`$session_time`

длительность сессии в секундах с точностью до миллисекунд

`$status`

статус сессии, может принимать одно из следующих значений:

|     |                                                                                            |
|-----|--------------------------------------------------------------------------------------------|
| 200 | сессия завершена успешно                                                                   |
| 400 | невозможно разобрать данные, полученные от клиента, например заголовок протокола PROXY     |
| 403 | доступ запрещен, например при ограничении доступа для <i>определенных адресов клиентов</i> |
| 500 | внутренняя ошибка сервера                                                                  |
| 502 | плохой шлюз, например если невозможно выбрать сервер группы или сервер недоступен          |
| 503 | сервис недоступен, например при ограничении по <i>числу соединений</i>                     |

`$time_iso8601`

локальное время в формате по стандарту ISO 8601

`$time_local`

локальное время в Common Log Format

## Почтовые модули

|               |                                                                                                                         |
|---------------|-------------------------------------------------------------------------------------------------------------------------|
| <i>Mail</i>   | Основная функциональность почтового прокси-сервера.                                                                     |
| <i>Auth</i>   | Аутентификация пользователей и выбор сервера для последующего проксирования с помощью HTTP-запросов к внешнему серверу. |
| <i>HTTP</i>   | Поддержка протокола IMAP.                                                                                               |
| <i>IMAP</i>   | Поддержка протокола POP3.                                                                                               |
| <i>POP3</i>   | Настройка проксирования к другим серверам.                                                                              |
| <i>Proxy</i>  | Определение адреса и порта клиента при работе за другим прокси-сервером.                                                |
| <i>RealIP</i> | Поддержка протокола SMTP.                                                                                               |
| <i>SMTP</i>   | Поддержка протоколов SSL/TLS и StartTLS.                                                                                |
| <i>SSL</i>    |                                                                                                                         |

## Почтовый модуль

Базовый почтовый модуль реализует основную функциональность почтового прокси-сервера: это поддержка протоколов SMTP, IMAP и POP3, настройка серверных блоков, маршрутизация почтовых запросов, аутентификация пользователей и поддержка SSL/TLS для защиты почтовых соединений.

Остальные модули этого раздела расширяют эту функциональность, позволяя гибко настраивать и оптимизировать работу почтового сервера под различные сценарии и требования.

В образах из наших репозиториях модуль уже включен в сборку.



## Пример конфигурации

```
worker_processes auto;

error_log /var/log/angie/error.log info;

events {
 worker_connections 1024;
}

mail {
 server_name mail.example.com;
 auth_http localhost:9000/cgi-bin/auth.cgi;

 imap_capabilities IMAP4rev1 UIDPLUS IDLE LITERAL+ QUOTA;

 pop3_auth plain apop cram-md5;
 pop3_capabilities LAST TOP USER PIPELINING UIDL;

 smtp_auth login plain cram-md5;
 smtp_capabilities "SIZE 10485760" ENHANCEDSTATUSCODES 8BITMIME DSN;
 xclient off;

 server {
 listen 25;
 protocol smtp;
 }
 server {
 listen 110;
 protocol pop3;
 proxy_pass_error_message on;
 }
 server {
 listen 143;
 protocol imap;
 }
 server {
 listen 587;
 protocol smtp;
 }
}
```

## Директивы

### listen

|                  |                                                                                                                                                                                                                                                                                                                                |
|------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Синтаксис</i> | <code>listen</code> <i>адрес[:порт]</i> [ <i>ssl</i> ] [ <i>proxy_protocol</i> ] [ <i>backlog=число</i> ]<br>[ <i>rcvbuf=размер</i> ] [ <i>sndbuf=размер</i> ] [ <i>bind</i> ] [ <i>ipv6only=on   off</i> ] [ <i>reuseport</i> ]<br>[ <i>so_keepalive=on off</i> ][ <i>keepidle</i> ][: <i>keepintvl</i> ][: <i>keepcnt</i> ]; |
| По умолчанию     | —                                                                                                                                                                                                                                                                                                                              |
| <i>Контекст</i>  | server                                                                                                                                                                                                                                                                                                                         |

Задаёт *адрес* и *порт* для сокета, на котором сервер будет принимать соединения. Можно указать

только *порт*, и тогда Angie будет слушать на всех доступных IPv4-интерфейсах (и IPv6, если он включен). Кроме того, *адрес* может быть именем хоста, например:

```
listen 127.0.0.1:110;
listen *:110;
listen 110; # то же, что и *:110
listen localhost:110;
```

IPv6-адреса задаются в квадратных скобках:

```
listen [::1]:110;
listen [::]:110;
```

UNIX-сокеты задаются префиксом `unix:`:

```
listen unix:/var/run/angie.sock;
```

#### **Примечание**

Разные серверы должны слушать на разных парах *адрес:порт*.

|                       |                                                                                                                                                                                                                                      |
|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>ssl</b>            | указывает на то, что все соединения, принимаемые на данном слушающем сокете, должны работать в режиме SSL.                                                                                                                           |
| <b>proxy_protocol</b> | указывает на то, что все соединения, принимаемые на данном порту, должны использовать протокол PROXY. Полученная информация передается <i>серверу аутентификации</i> и может быть использована для <i>изменения адреса клиента</i> . |

В директиве `listen` можно также указать несколько дополнительных параметров, специфичных для связанных с сокетами системных вызовов.

|                        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>backlog=число</b>   | задает параметр <i>backlog</i> в вызове <i>listen()</i> , который ограничивает максимальный размер очереди ожидающих приема соединений. По умолчанию <i>backlog</i> устанавливается равным <i>-1</i> для FreeBSD, DragonFly BSD и macOS, и <i>511</i> для других платформ.                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| <b>rcvbuf=размер</b>   | задает размер буфера приема (параметр <i>SO_RCVBUF</i> ) для слушающего сокета.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| <b>sndbuf=размер</b>   | задает размер буфера передачи (параметр <i>SO_SNDBUF</i> ) для слушающего сокета.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| <b>bind</b>            | указывает, что для данного слушающего сокета нужно делать <i>bind()</i> отдельно. Это нужно потому, что если описаны несколько директив <code>listen</code> с одинаковым портом, но разными адресами, и одна из директив <code>listen</code> слушает на всех адресах для данного <i>порта</i> ( <i>*:порт</i> ), то Angie сделает <i>bind()</i> только на <i>*:порт</i> . Необходимо заметить, что в этом случае для определения адреса, на который пришло соединение, делается системный вызов <i>getsockname()</i> . Если же используются параметры <i>backlog</i> , <i>rcvbuf</i> , <i>sndbuf</i> , <i>ipv6only</i> , <i>reuseport</i> или <i>so_keepalive</i> , то для данной пары <i>адрес:порт</i> всегда делается отдельный вызов <i>bind()</i> . |
| <b>ipv6only=on/off</b> | определяет (через параметр сокета <i>IPV6_V6ONLY</i> ), будет ли слушающий на wildcard-адресе <i>:::</i> IPv6-сокеты принимать только IPv6-соединения, или же одновременно IPv6- и IPv4-соединения. По умолчанию параметр включен. Установить его можно только один раз на старте.                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| <b>multipath</b>       | включает прием соединений по протоколу <b>Multipath TCP (MPTCP)</b> , поддерживаемому в ядре Linux с версии 5.6.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |

`so_keepalive=on | off | [keepidle]:[keepintvl]:[keepcnt]`

Конфигурирует для слушающего сокета поведение "TCP keepalive".

|                  |                                                                                   |
|------------------|-----------------------------------------------------------------------------------|
| <code>''</code>  | если параметр опущен, для сокета будут действовать настройки операционной системы |
| <code>on</code>  | для сокета включается параметр <code>SO_KEEPALIVE</code>                          |
| <code>off</code> | для сокета параметр <code>SO_KEEPALIVE</code> выключается                         |

Некоторые операционные системы поддерживают настройку параметров "TCP keepalive" на уровне сокета посредством параметров `TCP_KEEPIDLE`, `TCP_KEEPINTVL` и `TCP_KEEPCNT`. На таких системах их можно сконфигурировать с помощью параметров `keepidle`, `keepintvl` и `keepcnt`. Один или два параметра могут быть опущены, в таком случае для соответствующего параметра сокета будут действовать стандартные системные настройки.

Например,

```
so_keepalive=30m:10
```

установит таймаут бездействия (`TCP_KEEPIDLE`) в 30 минут, для интервала проб (`TCP_KEEPINTVL`) будет действовать стандартная системная настройка, а счетчик проб (`TCP_KEEPCNT`) будет равен 10.

Разные серверы должны слушать на разных парах *адрес:порт*.

## mail

|                  |                           |
|------------------|---------------------------|
| <i>Синтаксис</i> | <code>mail { ... }</code> |
| По умолчанию     | —                         |
| <i>Контекст</i>  | <code>main</code>         |

Предоставляет контекст конфигурационного файла, в котором указываются директивы почтового сервера.

## max\_commands

|                  |                                  |
|------------------|----------------------------------|
| <i>Синтаксис</i> | <code>max_commands число;</code> |
| По умолчанию     | <code>max_commands 1000;</code>  |
| <i>Контекст</i>  | <code>mail, server</code>        |

Задаёт максимальное количество команд, отправляемых во время аутентификации, для усиления защиты от DoS-атак.

## max\_errors

|                  |                                |
|------------------|--------------------------------|
| <i>Синтаксис</i> | <code>max_errors число;</code> |
| По умолчанию     | <code>max_errors 5;</code>     |
| <i>Контекст</i>  | mail, server                   |

Задаёт число ошибок протокола, по достижении которого соединение закрывается.

## protocol

|                  |                                           |
|------------------|-------------------------------------------|
| <i>Синтаксис</i> | <code>protocol imap   pop3   smtp;</code> |
| По умолчанию     | —                                         |
| <i>Контекст</i>  | server                                    |

Задаёт протокол проксируемого сервера. Поддерживаются протоколы *IMAP*, *POP3* и *SMTP*.

Если директива не указана, то протокол может быть определен автоматически по общеизвестному порту, указанному в директиве `listen`:

```
imap: 143, 993
pop3: 110, 995
smtp: 25, 587, 465
```

## resolver

|                  |                                                                                                   |
|------------------|---------------------------------------------------------------------------------------------------|
| <i>Синтаксис</i> | <code>resolver адрес ... [valid=время] [ipv4=on   off] [ipv6=on   off] [status_zone=зона];</code> |
| По умолчанию     | <code>resolver off;</code>                                                                        |
| <i>Контекст</i>  | mail, server                                                                                      |

Задаёт серверы DNS, используемые для преобразования имени хоста клиента для передачи его на сервер аутентификации и в команде *XCLIENT* при проксировании SMTP, например:

```
resolver 127.0.0.53 [::1]:5353;
```

Адрес может быть указан в виде доменного имени или IP-адреса, и необязательного порта. Если порт не указан, используется порт 53. Серверы DNS опрашиваются циклически.

### **Примечание**

Значение директивы наследуется вложенными блоками и может быть переопределено в них при необходимости. В пределах одного блока допустимо указывать директиву только один раз. Если она повторяется, действует последнее определение.

По умолчанию Angie кэширует ответы, используя значение TTL из ответа DNS. Если директива `resolver` не указана и не выполняются динамические DNS-запросы (например, при использовании

фиксированных имен в *Proxy* без переменных), указание резолвера не требуется: имена будут разрешены при запуске с помощью системного резолвера. Необязательный параметр `valid` позволяет это переопределить:

|                    |                                                                           |
|--------------------|---------------------------------------------------------------------------|
| <code>valid</code> | необязательный параметр, позволяет переопределить срок кэширования ответа |
|--------------------|---------------------------------------------------------------------------|

```
resolver 127.0.0.53 [::1]:5353 valid=30s;
```

По умолчанию Angie будет искать как IPv4-, так и IPv6-адреса при преобразовании имен в адреса.

|                          |                                                                                                     |
|--------------------------|-----------------------------------------------------------------------------------------------------|
| <code>ipv4=off</code>    | запрещает поиск IPv4-адресов                                                                        |
| <code>ipv6=off</code>    | запрещает поиск IPv6-адресов                                                                        |
| <code>status_zone</code> | необязательный параметр, включает сбор информации о запросах и ответах сервера DNS в указанной зоне |

#### 💡 Совет

Для предотвращения DNS-спуфинга рекомендуется использовать DNS-серверы в защищенной доверенной локальной сети.

### resolver\_timeout

|                  |                                      |
|------------------|--------------------------------------|
| <i>Синтаксис</i> | <code>resolver_timeout время;</code> |
| По умолчанию     | <code>resolver_timeout 30s;</code>   |
| <i>Контекст</i>  | mail, server                         |

Задаёт таймаут для преобразования имени в адрес, например:

```
resolver_timeout 5s;
```

### server

|                  |                             |
|------------------|-----------------------------|
| <i>Синтаксис</i> | <code>server { ... }</code> |
| По умолчанию     | —                           |
| <i>Контекст</i>  | mail                        |

Задаёт конфигурацию для сервера.

## server\_name

|                  |                                    |
|------------------|------------------------------------|
| <i>Синтаксис</i> | <code>server_name имя;</code>      |
| По умолчанию     | <code>server_name hostname;</code> |
| <i>Контекст</i>  | mail, server                       |

Задаёт имя сервера, используемое:

- в начальном приветствии POP3/SMTP-сервера;
- в salt при аутентификации SASL-методом CRAM-MD5;
- в команде EHLO при подключении к SMTP-бэкенду, если разрешена передача команды *XCLIENT*.

Если директива не указана, используется имя хоста (*hostname*) машины.

## timeout

|                  |                             |
|------------------|-----------------------------|
| <i>Синтаксис</i> | <code>timeout время;</code> |
| По умолчанию     | <code>timeout 60s;</code>   |
| <i>Контекст</i>  | mail, server                |

Задаёт таймаут, который используется до начала проксирования на бэкенд.

## Auth HTTP

Модуль позволяет выполнять аутентификацию на основе дополнительного HTTP-запроса перед обработкой основного запроса. Если подзапрос возвращает статус 2xx, основной запрос продолжается; если возвращается 401 или 403, пользователю отправляется соответствующая ошибка, а при любом другом статусе возвращается ошибка 500. Такой подход обычно используется для передачи аутентификации внешним сервисам, объединения аутентификации в разных приложениях или интеграции со сторонними системами, такими как OAuth или LDAP.

## Директивы

### auth\_http

|                  |                             |
|------------------|-----------------------------|
| <i>Синтаксис</i> | <code>auth_http uri;</code> |
| По умолчанию     | —                           |
| <i>Контекст</i>  | mail, server                |

Задаёт URL HTTP-сервера аутентификации. Протокол описан *ниже*.

## auth\_http\_header

|                  |                                              |
|------------------|----------------------------------------------|
| <i>Синтаксис</i> | auth_http_header <i>заголовок значение</i> ; |
| По умолчанию     | —                                            |
| <i>Контекст</i>  | mail, server                                 |

Добавляет указанный заголовок к запросам, посылаемым на сервер аутентификации. Заголовок можно использовать в качестве shared secret для проверки, что запрос поступил от Angie. Например:

```
auth_http_header X-Auth-Key "secret_string";
```

## auth\_http\_pass\_client\_cert

|                  |                                      |
|------------------|--------------------------------------|
| <i>Синтаксис</i> | auth_http_pass_client_cert on   off; |
| По умолчанию     | auth_http_pass_client_cert off;      |
| <i>Контекст</i>  | mail, server                         |

Добавляет заголовок Auth-SSL-Cert с *клиентским сертификатом* в формате PEM (закодирован в формате *urlencode*) к запросам, посылаемым на сервер аутентификации.

## auth\_http\_timeout

|                  |                                  |
|------------------|----------------------------------|
| <i>Синтаксис</i> | auth_http_timeout <i>время</i> ; |
| По умолчанию     | auth_http_timeout 60s;           |
| <i>Контекст</i>  | mail, server                     |

Задаёт таймаут общения с сервером аутентификации.

## Протокол

Для общения с сервером аутентификации используется протокол HTTP. Данные в теле ответа игнорируются, информация передается только в заголовках.

## Примеры запросов и ответов:

Запрос:

```
GET /auth HTTP/1.0
Host: localhost
Auth-Method: plain # plain/apop/cram-md5/external
Auth-User: user
Auth-Pass: password
Auth-Protocol: imap # imap/pop3/smtp
Auth-Login-Attempt: 1
Client-IP: 192.0.2.42
Client-Host: client.example.org
```

Хороший ответ:

```
HTTP/1.0 200 OK
Auth-Status: OK
Auth-Server: 198.51.100.1
Auth-Port: 143
```

Плохой ответ:

```
HTTP/1.0 200 OK
Auth-Status: Invalid login or password
Auth-Wait: 3
```

Если заголовок **Auth-Wait** нет, то после выдачи ошибки соединение будет закрыто. В текущей реализации на каждую попытку аутентификации выделяется память, которая освобождается только при завершении сессии. Поэтому число неудачных попыток аутентификации в рамках одной сессии должно быть ограничено — после 10-20 попыток (номер попытки передается в заголовке **Auth-Login-Attempt**) сервер должен выдать ответ без заголовка **Auth-Wait**.

При использовании APOP или CRAM-MD5 запрос и ответ будут выглядеть так:

```
GET /auth HTTP/1.0
Host: localhost
Auth-Method: apop
Auth-User: user
Auth-Salt: <238188073.1163692009@mail.example.com>
Auth-Pass: auth_response
Auth-Protocol: imap
Auth-Login-Attempt: 1
Client-IP: 192.0.2.42
Client-Host: client.example.org
```

Хороший ответ:

```
HTTP/1.0 200 OK
Auth-Status: OK
Auth-Server: 198.51.100.1
Auth-Port: 143
Auth-Pass: plain-text-pass
```

Если в ответе есть заголовок **Auth-User**, то он переопределяет имя пользователя, используемое для аутентификации с бэкендом.

Для SMTP в ответе дополнительно учитывается заголовок **Auth-Error-Code** — если он есть, то используется как код ответа в случае ошибки. Если его нет, то по умолчанию к **Auth-Status** будет добавлен код *535 5.7.0*.

Например, если от сервера аутентификации будет получен ответ:

```
HTTP/1.0 200 OK
Auth-Status: Temporary server problem, try again later
Auth-Error-Code: 451 4.3.0
Auth-Wait: 3
```

то по SMTP клиенту будет выдана ошибка

```
451 4.3.0 Temporary server problem, try again later
```

Если при проксировании SMTP не требуется аутентификация, запрос будет выглядеть так:



```
GET /auth HTTP/1.0
Host: localhost
Auth-Method: none
Auth-User:
Auth-Pass:
Auth-Protocol: smtp
Auth-Login-Attempt: 1
Client-IP: 192.0.2.42
Client-Host: client.example.org
Auth-SMTP-Helo: client.example.org
Auth-SMTP-From: MAIL FROM: <>
Auth-SMTP-To: RCPT TO: <postmaster@mail.example.com>
```

Для клиентского соединения по протоколу SSL/TLS добавляется заголовок **Auth-SSL**, и если директива `ssl_verify_client` включена, заголовок **Auth-SSL-Verify** содержит результат проверки клиентского сертификата: **SUCCESS**, **FAILED:reason** и, если сертификат не был предоставлен, **NONE**.

Если клиентский сертификат был предоставлен, информация о нем передается в следующих заголовках запроса: **Auth-SSL-Subject**, **Auth-SSL-Issuer**, **Auth-SSL-Serial** и **Auth-SSL-Fingerprint**. Если директива `auth_http_pass_client_cert` включена, сам сертификат передается в заголовке **Auth-SSL-Cert**. Протокол и шифр установленного соединения передаются в заголовках **Auth-SSL-Protocol** и **Auth-SSL-Cipher**. Запрос будет выглядеть так:

```
GET /auth HTTP/1.0
Host: localhost
Auth-Method: plain
Auth-User: user
Auth-Pass: password
Auth-Protocol: imap
Auth-Login-Attempt: 1
Client-IP: 192.0.2.42
Auth-SSL: on
Auth-SSL-Protocol: TLSv1.3
Auth-SSL-Cipher: TLS_AES_256_GCM_SHA384
Auth-SSL-Verify: SUCCESS
Auth-SSL-Subject: /CN=example.com
Auth-SSL-Issuer: /CN=example.com
Auth-SSL-Serial: C07AD56B846B5BFF
Auth-SSL-Fingerprint: 29d6a80a123d13355ed16b4b04605e29cb55a5ad
```

При использовании *протокола* **PROXY**, информация о нем передается в следующих заголовках запроса: **Proxy-Protocol-Addr**, **Proxy-Protocol-Port**, **Proxy-Protocol-Server-Addr** и **Proxy-Protocol-Server-Port**.

## IMAP

Модуль обеспечивает поддержку почтового протокола IMAP, позволяя серверу взаимодействовать с системами хранения почты. Он устанавливает соединения с серверами IMAP, обрабатывает основные команды, такие как список папок и получение сообщений, а также обеспечивает безопасную аутентификацию и управление статусами сообщений.

### Директивы

#### imap\_auth

|                  |                                   |
|------------------|-----------------------------------|
| <i>Синтаксис</i> | <code>imap_auth метод ...;</code> |
| По умолчанию     | <code>imap_auth plain;</code>     |
| <i>Контекст</i>  | mail, server                      |

Задаёт разрешённые методы аутентификации IMAP-клиентов. Поддерживаемые методы:

|                       |                                                                                                      |
|-----------------------|------------------------------------------------------------------------------------------------------|
| <code>plain</code>    | <code>LOGIN, AUTH=PLAIN</code>                                                                       |
| <code>login</code>    | <code>AUTH=LOGIN</code>                                                                              |
| <code>cram-md5</code> | <code>AUTH=CRAM-MD5</code> . Для работы этого метода пароль должен храниться в незашифрованном виде. |
| <code>external</code> | <code>AUTH=EXTERNAL</code>                                                                           |

Методы аутентификации с передачей пароля открытым текстом (команда `LOGIN, AUTH=PLAIN` и `AUTH=LOGIN`) включены всегда, однако если методы `plain` и `login` не указаны, то `AUTH=PLAIN` и `AUTH=LOGIN` не будут автоматически добавляться в `imap_capabilities`.

#### imap\_capabilities

|                  |                                                         |
|------------------|---------------------------------------------------------|
| <i>Синтаксис</i> | <code>imap_capabilities расширение ...;</code>          |
| По умолчанию     | <code>imap_capabilities IMAP4 IMAP4rev1 UIDPLUS;</code> |
| <i>Контекст</i>  | mail, server                                            |

Позволяет указать список расширений протокола IMAP, выдаваемый клиенту по команде CAPABILITY. В зависимости от значения директивы `starttls` к этому списку автоматически добавляются методы аутентификации, указанные в директиве `imap_auth`, и `STARTTLS`.

В данной директиве имеет смысл указать расширения, поддерживаемые IMAP-серверами, на которые проксируются клиенты (если эти расширения относятся к командам, используемым после аутентификации, когда Angie прозрачно проксирует подключение клиента на бэкенд).

## imap\_client\_buffer

|                  |                                         |
|------------------|-----------------------------------------|
| <i>Синтаксис</i> | <code>imap_client_buffer размер;</code> |
| По умолчанию     | <code>imap_client_buffer 4k 8k;</code>  |
| <i>Контекст</i>  | mail, server                            |

Задаёт размер буфера для чтения IMAP-команд. По умолчанию размер одного буфера равен размеру страницы. В зависимости от платформы это или 4К, или 8К.

## POP3

Модуль обеспечивает поддержку почтового протокола POP3, позволяя серверу загружать сообщения с почтовых серверов. Он подключается к серверам POP3, получает заголовки и содержимое сообщений, обеспечивает безопасную аутентификацию и управляет статусами сообщений, такими как загружено или удалено.

### Директивы

#### pop3\_auth

|                  |                                   |
|------------------|-----------------------------------|
| <i>Синтаксис</i> | <code>pop3_auth метод ...;</code> |
| По умолчанию     | <code>pop3_auth plain;</code>     |
| <i>Контекст</i>  | mail, server                      |

Задаёт разрешённые методы аутентификации POP3-клиентов. Поддерживаемые методы:

|                 |                                                                                                |
|-----------------|------------------------------------------------------------------------------------------------|
| <b>plain</b>    | <b>USER/PASS, AUTH PLAIN, AUTH LOGIN</b>                                                       |
| <b>apop</b>     | <b>APOP</b> . Для работы этого метода пароль должен храниться в незашифрованном виде.          |
| <b>cram-md5</b> | <b>AUTH=CRAM-MD5</b> . Для работы этого метода пароль должен храниться в незашифрованном виде. |
| <b>external</b> | <b>AUTH=EXTERNAL</b>                                                                           |

Методы аутентификации с передачей пароля открытым текстом (**USER/PASS**, **AUTH PLAIN** и **AUTH LOGIN**) включены всегда, однако если метод **plain** не указан, то **AUTH PLAIN** и **AUTH LOGIN** не будут автоматически добавляться в *pop3\_capabilities*.

#### pop3\_capabilities

|                  |                                                |
|------------------|------------------------------------------------|
| <i>Синтаксис</i> | <code>pop3_capabilities расширение ...;</code> |
| По умолчанию     | <code>pop3_capabilities TOP USER UIDL;</code>  |
| <i>Контекст</i>  | mail, server                                   |

Позволяет указать список расширений протокола POP3, выдаваемый клиенту по команде CAPA. В зависимости от значения директивы *starttls* к этому списку автоматически добавляются методы аутентификации, указанные в директиве *pop3\_auth* (расширение **SASL**), и **STLS**.

В данной директиве имеет смысл указать расширения, поддерживаемые POP3-серверами, на которые проксируются клиенты (если эти расширения относятся к командам, используемым после аутентификации, когда Angie прозрачно проксирует подключение клиента на бэкенд).

## Proxy

Модуль обеспечивает поддержку почтовых протоколов (POP3, IMAP, SMTP), позволяя серверу работать в качестве прокси между клиентами и почтовыми серверами. Он устанавливает соединения с серверами, выполняет безопасную аутентификацию с использованием открытого текста, SSL/TLS или STARTTLS, правильно маршрутизирует трафик клиентов и поддерживает гибкий выбор методов аутентификации и серверов.

## Директивы

### proxy\_buffer

|                  |                      |
|------------------|----------------------|
| <i>Синтаксис</i> | proxy_buffer размер; |
| По умолчанию     | proxy_buffer 4k 8k;  |
| <i>Контекст</i>  | mail, server         |

Задаёт размер буфера, используемого при проксировании. По умолчанию размер одного буфера равен размеру страницы. В зависимости от платформы это или 4К, или 8К.

### proxy\_pass\_error\_message

|                  |                                    |
|------------------|------------------------------------|
| <i>Синтаксис</i> | proxy_pass_error_message on   off; |
| По умолчанию     | proxy_pass_error_message off;      |
| <i>Контекст</i>  | mail, server                       |

Определяет, передавать ли клиенту сообщение об ошибке, полученное при аутентификации на бэкенде.

Обычно, если аутентификация в Angie прошла успешно, бэкенд не может вернуть ошибку. Если же он все-таки возвращает ошибку, это значит, что произошла ошибка внутри системы. В таких случаях сообщение бэкенда может содержать информацию, которую нельзя показывать клиенту. Однако для некоторых POP3-серверов ошибка в ответ на правильный пароль является штатным поведением. В этом случае директиву стоит включить.

### proxy\_protocol

|                  |                          |
|------------------|--------------------------|
| <i>Синтаксис</i> | proxy_protocol on   off; |
| По умолчанию     | proxy_protocol off;      |
| <i>Контекст</i>  | mail, server             |

Включает протокол **PROXY** для соединений с бэкендом.

## proxy\_smtp\_auth

|                  |                           |
|------------------|---------------------------|
| <i>Синтаксис</i> | proxy_smtp_auth on   off; |
| По умолчанию     | proxy_smtp_auth off;      |
| <i>Контекст</i>  | mail, server              |

Разрешает или запрещает аутентификацию пользователей на SMTP-бэкенде при помощи команды AUTH.

Если также включен *XCLIENT*, то команда XCLIENT не будет отправлять параметр LOGIN.

## proxy\_timeout

|                  |                      |
|------------------|----------------------|
| <i>Синтаксис</i> | proxy_timeout время; |
| По умолчанию     | proxy_timeout 24h;   |
| <i>Контекст</i>  | mail, server         |

Задаёт таймаут между двумя идущими подряд операциями чтения или записи на клиентском соединении или соединении с проксируемым сервером. Если по истечении этого времени данные не передавались, соединение закрывается.

## xclient

|                  |                   |
|------------------|-------------------|
| <i>Синтаксис</i> | xclient on   off; |
| По умолчанию     | xclient on;       |
| <i>Контекст</i>  | mail, server      |

Разрешает или запрещает передачу команды XCLIENT с параметрами клиента при подключении к SMTP-бэкенду.

При помощи XCLIENT МТА может писать в лог информацию о клиенте и применять различные ограничения на основе этих данных.

Если команда XCLIENT разрешена, то при подключении к бэкенду Angie посылает ему следующие команды:

- EHLO с именем сервера
- XCLIENT
- EHLO или HELO, как ее передал клиент

Если *найденное* по IP-адресу клиента имя указывает на тот же адрес, оно передается в параметре NAME команды XCLIENT. Если имя не может быть найдено, указывает на другой адрес, или не задан *resolver*, то в параметре NAME передается [UNAVAILABLE]. Если же в процессе поиска имени или адреса произошла ошибка, передается [TEMPUNAVAIL].

Если команда XCLIENT запрещена, то при подключении к бэкенду Angie передает команду EHLO с именем сервера, если клиент передал EHLO, иначе HELO с именем сервера.

## RealIP

Позволяет менять адрес и необязательный порт клиента на переданные в указанном поле заголовка адрес и порт клиента на переданные в заголовке протокола PROXY. Протокол PROXY должен быть предварительно включен при помощи установки параметра `proxy_protocol` в директиве `listen`.

### Пример конфигурации

```
listen 110 proxy_protocol;

set_real_ip_from 192.168.1.0/24;
set_real_ip_from 192.168.2.1;
set_real_ip_from 2001:0db8::/32;
```

## Директивы

### set\_real\_ip\_from

|                  |                                                     |
|------------------|-----------------------------------------------------|
| <i>Синтаксис</i> | <code>set_real_ip_from адрес   CIDR   unix;;</code> |
| По умолчанию     | —                                                   |
| <i>Контекст</i>  | mail, server                                        |

Задаёт доверенные адреса, которые передают верный адрес для замены. Если указано специальное значение `unix:`, доверенными будут считаться все UNIX-сокеты.

## SMTP

Модуль обеспечивает поддержку почтового протокола SMTP, позволяя серверу проксировать исходящий почтовый трафик между клиентами и почтовыми серверами. Он устанавливает соединения с серверами SMTP, поддерживает безопасную аутентификацию с помощью методов LOGIN или PLAIN, обеспечивает шифрование через STARTTLS и SSL/TLS и маршрутизирует клиентские запросы на основе результатов аутентификации.

## Директивы

### smtp\_auth

|                  |                                     |
|------------------|-------------------------------------|
| <i>Синтаксис</i> | <code>smtp_auth метод ...;</code>   |
| По умолчанию     | <code>smtp_auth plain login;</code> |
| <i>Контекст</i>  | mail, server                        |

Задаёт разрешенные методы SASL-аутентификации SMTP-клиентов. Поддерживаемые методы:

|                       |                                                                                                      |
|-----------------------|------------------------------------------------------------------------------------------------------|
| <code>plain</code>    | <code>AUTH PLAIN</code>                                                                              |
| <code>login</code>    | <code>AUTH LOGIN</code>                                                                              |
| <code>cram-md5</code> | <code>AUTH CRAM-MD5</code> . Для работы этого метода пароль должен храниться в незашифрованном виде. |
| <code>external</code> | <code>AUTH EXTERNAL</code>                                                                           |
| <code>none</code>     | Аутентификация не требуется                                                                          |

Методы аутентификации с передачей пароля открытым текстом (`AUTH PLAIN` и `AUTH LOGIN`) включены всегда, однако если методы `plain` и `login` не указаны, то `AUTH PLAIN` и `AUTH LOGIN` не будут автоматически добавляться в `smtp_capabilities`.

### smtp\_capabilities

|                  |                                                       |
|------------------|-------------------------------------------------------|
| <i>Синтаксис</i> | <code>smtp_capabilities</code> <i>расширение ...;</i> |
| По умолчанию     | —                                                     |
| <i>Контекст</i>  | mail, server                                          |

Позволяет указать список расширений протокола SMTP, выдаваемый клиенту в ответе на команду EHLO. В зависимости от значения директивы `starttls` к этому списку автоматически добавляются методы аутентификации, указанные в директиве `smtp_auth`, и `STARTTLS`.

В данной директиве имеет смысл указать расширения, поддерживаемые МТА, на который проксируются клиенты (если эти расширения относятся к командам, используемым после аутентификации, когда Angie прозрачно проксирует подключение клиента на бэкенд).

### smtp\_client\_buffer

|                  |                                                |
|------------------|------------------------------------------------|
| <i>Синтаксис</i> | <code>smtp_client_buffer</code> <i>размер;</i> |
| По умолчанию     | <code>smtp_client_buffer 4k 8k;</code>         |
| <i>Контекст</i>  | mail, server                                   |

Задаёт размер буфера для чтения SMTP-команд. По умолчанию размер одного буфера равен размеру страницы. В зависимости от платформы это или 4К, или 8К.

### smtp\_greeting\_delay

|                  |                                                 |
|------------------|-------------------------------------------------|
| <i>Синтаксис</i> | <code>smtp_greeting_delay</code> <i>размер;</i> |
| По умолчанию     | <code>smtp_greeting_delay 0;</code>             |
| <i>Контекст</i>  | mail, server                                    |

Позволяет задать задержку перед отправкой SMTP-приветствия, чтобы отклонить клиентов, не ожидающих приветствия до начала отправки SMTP-команд.

## SSL

Модуль обеспечивает поддержку шифрования SSL/TLS для почтовых прокси-протоколов (POP3, IMAP, SMTP), позволяя устанавливать защищённые соединения между клиентами и сервером. Он обеспечивает шифрование SSL/TLS для входящих подключений, поддерживает обновление соединений через STARTTLS, управляет сертификатами и ключами, а также контролирует настройки SSL, такие как выбор шифров и версий протоколов.

При сборке из исходного кода модуль не собирается по умолчанию; его необходимо включить с помощью параметра сборки `--with-mail_ssl_module`. В пакетах и образах из наших репозиториев модуль включен в сборку.

### Примечание

Для этого модуля нужна библиотека OpenSSL.

## Пример конфигурации

Для уменьшения загрузки процессора рекомендуется

- установить число *рабочих процессов* равным числу процессоров,
- включить *разделяемый кэш сессий*,
- выключить *встроенный кэш сессий*
- и, возможно, увеличить *время жизни* сессии (по умолчанию 5 минут):

```
worker_processes auto;

mail {
 ...

 server {
 listen 993 ssl;

 ssl_protocols TLSv1.2 TLSv1.3;
 ssl_ciphers AES128-SHA:AES256-SHA:RC4-SHA:DES-CBC3-SHA:RC4-MD5;
 ssl_certificate /usr/local/angie/conf/cert.pem;
 ssl_certificate_key /usr/local/angie/conf/cert.key;
 ssl_session_cache shared:SSL:10m;
 ssl_session_timeout 10m;

 # ...
 }
}
```



## Директивы

### ssl\_certificate

|                  |                               |
|------------------|-------------------------------|
| <i>Синтаксис</i> | ssl_certificate <i>файл</i> ; |
| По умолчанию     | —                             |
| <i>Контекст</i>  | mail, server                  |

Указывает файл с сертификатом в формате PEM для данного сервера. Если вместе с основным сертификатом нужно указать промежуточные, то они должны находиться в этом же файле в следующем порядке — сначала основной сертификат, а затем промежуточные. В этом же файле может находиться секретный ключ в формате PEM.

Директива может быть указана несколько раз для загрузки сертификатов разных типов, например RSA и ECDSA:

```
server {
 listen 993 ssl;

 ssl_certificate example.com.rsa.crt;
 ssl_certificate_key example.com.rsa.key;

 ssl_certificate example.com.ecdsa.crt;
 ssl_certificate_key example.com.ecdsa.key;

 # ...
}
```

Возможность задавать отдельные цепочки сертификатов для разных сертификатов есть только в OpenSSL 1.0.2 и выше. Для более старых версий следует указывать только одну цепочку сертификатов.

Вместо **файла** можно указать значение "data:*сертификат*", при котором сертификат загружается без использования промежуточных файлов.

Ненадлежащее использование подобного синтаксиса может быть небезопасно, например данные секретного ключа могут попасть в *лог ошибок*.

### ssl\_certificate\_key

|                  |                                   |
|------------------|-----------------------------------|
| <i>Синтаксис</i> | ssl_certificate_key <i>файл</i> ; |
| По умолчанию     | —                                 |
| <i>Контекст</i>  | mail, server                      |

Указывает файл с секретным ключом в формате PEM для данного виртуального сервера.

Вместо **файла** можно указать значение "engine:*имя*:id", которое загружает ключ с указанным *id* из OpenSSL engine с заданным именем.

Вместо **файла** также можно указать значение "data:*ключ*", при котором секретный ключ загружается без использования промежуточных файлов. При этом следует учитывать, что ненадлежащее использование подобного синтаксиса может быть небезопасно, например данные секретного ключа могут попасть в *лог ошибок*.

## ssl\_ciphers

|                  |                                            |
|------------------|--------------------------------------------|
| <i>Синтаксис</i> | <code>ssl_ciphers шифры;</code>            |
| По умолчанию     | <code>ssl_ciphers HIGH:!aNULL:!MD5;</code> |
| <i>Контекст</i>  | mail, server                               |

Описывает разрешенные шифры. Шифры задаются в формате, поддерживаемом библиотекой OpenSSL, например:

```
ssl_ciphers ALL:!aNULL:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP;
```

Список шифров зависит от установленной версии OpenSSL. Полный список можно посмотреть с помощью команды `openssl ciphers`.

### ⚠ Предупреждение

Директива `ssl_ciphers` не настраивает шифры для TLS 1.3 при использовании OpenSSL. Для настройки шифров TLS 1.3 в OpenSSL используйте директиву `ssl_conf_command`, добавленную для расширенной конфигурации SSL.

- В LibreSSL шифры TLS 1.3 можно настраивать с помощью `ssl_ciphers`.
- В BoringSSL шифры TLS 1.3 настроить невозможно.

## ssl\_client\_certificate

|                  |                                           |
|------------------|-------------------------------------------|
| <i>Синтаксис</i> | <code>ssl_client_certificate файл;</code> |
| По умолчанию     | —                                         |
| <i>Контекст</i>  | mail, server                              |

Указывает файл с доверенными сертификатами CA в формате PEM, которые используются для проверки клиентских сертификатов.

Список сертификатов будет отправляться клиентам. Если это нежелательно, можно воспользоваться директивой `ssl_trusted_certificate`.

## ssl\_conf\_command

|                  |                                             |
|------------------|---------------------------------------------|
| <i>Синтаксис</i> | <code>ssl_conf_command имя значение;</code> |
| По умолчанию     | —                                           |
| <i>Контекст</i>  | mail, server                                |

Задаёт произвольные конфигурационные команды OpenSSL.

### i Примечание

Директива поддерживается при использовании OpenSSL 1.0.2 и выше. Чтобы настроить шифры TLS 1.3 в OpenSSL, используйте команду `ciphersuites`.

На одном уровне может быть указано несколько директив `ssl_conf_command`:

```
ssl_conf_command Options PrioritizeChaCha;
ssl_conf_command Ciphersuites TLS_CHACHA20_POLY1305_SHA256;
```

Директивы наследуются с предыдущего уровня конфигурации при условии, что на данном уровне не описаны свои директивы `ssl_conf_command`.

#### Предупреждение

Изменение настроек OpenSSL напрямую может привести к неожиданному поведению.

### ssl\_crl

|                  |                            |
|------------------|----------------------------|
| <i>Синтаксис</i> | <code>ssl_crl файл;</code> |
| По умолчанию     | —                          |
| <i>Контекст</i>  | mail, server               |

Указывает файл с отозванными сертификатами (CRL) в формате PEM, используемыми для *проверки* клиентских сертификатов.

### ssl\_dhparam

|                  |                                |
|------------------|--------------------------------|
| <i>Синтаксис</i> | <code>ssl_dhparam файл;</code> |
| По умолчанию     | —                              |
| <i>Контекст</i>  | mail, server                   |

Указывает файл с параметрами для DHE-шифров.

#### Предупреждение

По умолчанию параметры не заданы, и соответственно DHE-шифры не будут использоваться.

### ssl\_ecdh\_curve

|                  |                                     |
|------------------|-------------------------------------|
| <i>Синтаксис</i> | <code>ssl_ecdh_curve кривая;</code> |
| По умолчанию     | <code>ssl_ecdh_curve auto;</code>   |
| <i>Контекст</i>  | mail, server                        |

Задаёт кривую для ECDHE-шифров.

#### Примечание

При использовании OpenSSL 1.0.2 и выше можно указывать несколько кривых, например:

```
ssl_ecdh_curve prime256v1:secp384r1;
```

Специальное значение `auto` соответствует встроенному в библиотеку OpenSSL списку кривых для OpenSSL 1.0.2 и выше, или `prime256v1` для более старых версий.

#### **Примечание**

При использовании OpenSSL 1.0.2 и выше директива задает список кривых, поддерживаемых сервером. Поэтому для работы ECDSA-сертификатов важно, чтобы список включал кривые, используемые в сертификатах.

### `ssl_password_file`

|                  |                                      |
|------------------|--------------------------------------|
| <i>Синтаксис</i> | <code>ssl_password_file файл;</code> |
| По умолчанию     | —                                    |
| <i>Контекст</i>  | mail, server                         |

Задаёт файл с паролями от *секретных ключей*, где каждый пароль указан на отдельной строке. Пароли применяются по очереди в момент загрузки ключа.

Пример:

```
mail {
 ssl_password_file /etc/keys/global.pass;
 ...

 server {
 server_name mail1.example.com;
 ssl_certificate_key /etc/keys/first.key;
 }

 server {
 server_name mail2.example.com;

 # вместо файла можно указать именованный канал
 ssl_password_file /etc/keys/fifo;
 ssl_certificate_key /etc/keys/second.key;
 }
}
```

### `ssl_prefer_server_ciphers`

|                  |                                                  |
|------------------|--------------------------------------------------|
| <i>Синтаксис</i> | <code>ssl_prefer_server_ciphers on   off;</code> |
| По умолчанию     | <code>ssl_prefer_server_ciphers off;</code>      |
| <i>Контекст</i>  | mail, server                                     |

При использовании протоколов SSLv3 и TLS устанавливает приоритет серверных шифров над клиентскими.

## ssl\_protocols

|                  |                                                                                   |
|------------------|-----------------------------------------------------------------------------------|
| <i>Синтаксис</i> | <code>ssl_protocols [SSLv2] [SSLv3] [TLSv1] [TLSv1.1] [TLSv1.2] [TLSv1.3];</code> |
| По умолчанию     | <code>ssl_protocols TLSv1.2 TLSv1.3;</code>                                       |
| <i>Контекст</i>  | mail, server                                                                      |

Изменено в версии 1.2.0: Параметр `TLSv1.3` добавлен к используемым по умолчанию.

Разрешает указанные протоколы.

### Примечание

Параметры `TLSv1.1` и `TLSv1.2` работают только при использовании OpenSSL 1.0.1 и выше.

Параметр `TLSv1.3` работает только при использовании OpenSSL 1.1.1 и выше.

## ssl\_session\_cache

|                  |                                                                                          |
|------------------|------------------------------------------------------------------------------------------|
| <i>Синтаксис</i> | <code>ssl_session_cache off   none   [builtin[:размер]] [shared:название:размер];</code> |
| По умолчанию     | <code>ssl_session_cache none;</code>                                                     |
| <i>Контекст</i>  | mail, server                                                                             |

Задаёт тип и размеры кэшей для хранения параметров сессий. Тип кэша может быть следующим:

|                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>off</b>     | жёсткое запрещение использования кэша сессий: Angie явно сообщает клиенту, что сессии не могут использоваться повторно.                                                                                                                                                                                                                                                                                                                                                          |
| <b>none</b>    | мягкое запрещение использования кэша сессий: Angie сообщает клиенту, что сессии могут использоваться повторно, но на самом деле не хранит параметры сессии в кэше.                                                                                                                                                                                                                                                                                                               |
| <b>builtin</b> | встроенный в OpenSSL кэш, используется в рамках только одного рабочего процесса. Размер кэша задается в сессиях. Если размер не задан, то он равен 20480 сессиям. Использование встроенного кэша может вести к фрагментации памяти.                                                                                                                                                                                                                                              |
| <b>shared</b>  | кэш, разделяемый между всеми рабочими процессами. Размер кэша задается в байтах, в 1 мегабайт может поместиться около 4000 сессий. У каждого разделяемого кэша должно быть произвольное название. Кэш с одинаковым названием может использоваться в нескольких серверах. Также он используется для автоматического создания, хранения и периодического обновления ключей TLS session tickets, если они не указаны явно с помощью директивы <code>ssl_session_ticket_key</code> . |

Можно использовать одновременно оба типа кэша, например:

```
ssl_session_cache builtin:1000 shared:SSL:10m;
```

однако использование только разделяемого кэша без встроенного должно быть более эффективным.

## ssl\_session\_ticket\_key

|                  |                                      |
|------------------|--------------------------------------|
| <i>Синтаксис</i> | ssl_session_ticket_key <i>файл</i> ; |
| По умолчанию     | —                                    |
| <i>Контекст</i>  | mail, server                         |

Задаёт файл с секретным ключом, применяемым при шифровании и расшифровании TLS session tickets. Директива необходима, если один и тот же ключ нужно использовать на нескольких серверах. По умолчанию используется случайно сгенерированный ключ.

Если указано несколько ключей, то только первый ключ используется для шифрования TLS session tickets. Это позволяет настроить ротацию ключей, например:

```
ssl_session_ticket_key current.key;
ssl_session_ticket_key previous.key;
```

Файл должен содержать 80 или 48 байт случайных данных и может быть создан следующей командой:

```
openssl rand 80 > ticket.key
```

В зависимости от размера файла для шифрования будет использоваться либо AES256 (для 80-байтных ключей), либо AES128 (для 48-байтных ключей).

## ssl\_session\_tickets

|                  |                               |
|------------------|-------------------------------|
| <i>Синтаксис</i> | ssl_session_tickets on   off; |
| По умолчанию     | ssl_session_tickets on;       |
| <i>Контекст</i>  | mail, server                  |

Разрешает или запрещает возобновление сессий при помощи TLS session tickets.

## ssl\_session\_timeout

|                  |                                    |
|------------------|------------------------------------|
| <i>Синтаксис</i> | ssl_session_timeout <i>время</i> ; |
| По умолчанию     | ssl_session_timeout 5m;            |
| <i>Контекст</i>  | mail, server                       |

Задаёт время, в течение которого клиент может повторно использовать параметры сессии.

## ssl\_trusted\_certificate

|                  |                                       |
|------------------|---------------------------------------|
| <i>Синтаксис</i> | ssl_trusted_certificate <i>файл</i> ; |
| По умолчанию     | —                                     |
| <i>Контекст</i>  | mail, server                          |

Задаёт файл с доверенными сертификатами СА в формате PEM, которые используются для *про-верки* клиентских сертификатов.

В отличие от *ssl\_client\_certificate*, список этих сертификатов не будет отправляться клиентам.

## ssl\_verify\_client

|                  |                                                         |
|------------------|---------------------------------------------------------|
| <i>Синтаксис</i> | ssl_verify_client on   off   optional   optional_no_ca; |
| По умолчанию     | ssl_verify_client off;                                  |
| <i>Контекст</i>  | mail, server                                            |

Разрешает проверку клиентских сертификатов. Результат проверки передается в заголовке Auth-SSL-Verify в запросе *аутентификации*. Если при проверке клиентского сертификата произошла ошибка или клиент не предоставил требуемый сертификат, соединение закрывается.

|                |                                                                                                                                                                                                                                                                                                                      |
|----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| optional       | запрашивает клиентский сертификат, и если сертификат был предоставлен, проверяет его                                                                                                                                                                                                                                 |
| optional_no_ca | запрашивает сертификат клиента, но не требует, чтобы он был подписан доверенным сертификатом СА. Это предназначено для случаев, когда фактическая проверка сертификата осуществляется внешним по отношению к Angie сервисом. Содержимое сертификата доступно в запросах, <i>посылаемых</i> на сервер аутентификации. |

## ssl\_verify\_depth

|                  |                                 |
|------------------|---------------------------------|
| <i>Синтаксис</i> | ssl_verify_depth <i>число</i> ; |
| По умолчанию     | ssl_verify_depth 1;             |
| <i>Контекст</i>  | mail, server                    |

Устанавливает глубину проверки в цепочке клиентских сертификатов.

## starttls

|                  |                                        |
|------------------|----------------------------------------|
| <i>Синтаксис</i> | <code>starttls on   off   only;</code> |
| По умолчанию     | <code>starttls off;</code>             |
| <i>Контекст</i>  | mail, server                           |

|             |                                                                          |
|-------------|--------------------------------------------------------------------------|
| <b>on</b>   | разрешить использование команд STLS для POP3 и STARTTLS для IMAP и SMTP; |
| <b>off</b>  | запретить использование команд STLS и STARTTLS;                          |
| <b>only</b> | требовать предварительного перехода на TLS.                              |

## Сторонние модули

Мы собираем пакеты для популярных сторонних nginx-совместимых модулей, разработанных за пределами нашей компании.

### Примечание

Мы не проверяем исходный код этих модулей и не отвечаем за последствия их установки; пакеты собираются на основе многочисленных запросов *исключительно* для удобства пользователей.

## Перечень модулей

| Модуль        | Версия | Описание                                             |
|---------------|--------|------------------------------------------------------|
| <i>GeoIP2</i> | 3.4    | Добавляет поиск по геоданным в базах MaxMind GeoIP2. |

## GeoIP2

Модуль GeoIP2 реализует поиск в базах MaxMind GeoIP2 по IP-адресу клиента (по умолчанию) или по значению определенной переменной. Поддерживает как IPv4, так и IPv6.

## Установка

Модуль включен в сборку.

## Загрузка модуля

Для работы с модулем необходимо загрузить его в контексте `main{}`:

```
load_module modules/nginx_http_geoip2_module.so; # для использования в блоке http{}
load_module modules/nginx_stream_geoip2_module.so; # для использования в контексте
↪stream{}
```



## Пример конфигурации

```
http {
 geoip2 /var/lib/GeoIP/GeoLite2-Country.mmdb {
 auto_reload 1h;
 $geoip2_country_code default=RU source=$http_x_forwarded_for country iso_code;
 $geoip2_country_name source=$http_x_forwarded_for country names ru;
 }

 log_format with_geoip '$server_port $remote_addr - $remote_user [$time_local] "
↪$request" '
 '$status $body_bytes_sent "$http_referer" '
 '"$http_user_agent" "$http_x_forwarded_for" "$http_host" '
 'country="$geoip2_country_code"';

 map $geoip2_country_code $denied {
 PL "1";
 QA "1";
 }

 server {
 listen 80;
 root /usr/share/angie/html;
 index index.html index.htm;
 access_log /var/log/angie/geoip_access.log with_geoip;

 if ($denied) {
 return 403;
 }
 }
}
```

## Дополнительная информация

Подробная документация и исходный код доступны по ссылке: [https://github.com/leev/nginx\\_http\\_geoip2\\_module](https://github.com/leev/nginx_http_geoip2_module)

## Настройка ACME

Модуль *ACME* в Angie обеспечивает автоматическое получение сертификатов с использованием протокола ACME. Протокол ACME предусматривает несколько способов проверки доменов (также используется термин *верификация*); модуль реализует *HTTP-проверку* и *DNS-проверку*, а также *проверку с помощью хуков* через самостоятельно реализуемый внешний сервис.

## Шаги настройки

Общие шаги для включения запроса сертификатов в конфигурации:

- **Настройте АСМЕ-клиент** в блоке `http` с помощью директивы `acme_client`, задающей уникальное имя клиента и другие параметры; можно настроить несколько клиентов АСМЕ.
- **Укажите домены, для которых запрашиваются сертификаты:** для доменных имен, перечисленных во всех директивах `server_name` всех блоков `server` с директивами `acme`, указывающими на один и тот же АСМЕ-клиент, будет выдан единый сертификат.
- **Настройте обработку запросов и вызовов АСМЕ:** это нужно для проверки владения доменом. Способ настройки зависит от способа проверки доменных имен:

| Способ                 | Требования к пользователю                                                                                                                                                                                      | Мультидомены | Домены со звездочкой |
|------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------|----------------------|
| <i>HTTP-проверка</i>   | Открыть порт 80 для входящих соединений на сервере Angie.                                                                                                                                                      | —            |                      |
| <i>DNS-проверка</i>    | Открыть порт 53 (или указанный в <code>acme_dns_port</code> ) для входящих соединений на сервере Angie.<br>Настроить NS-запись для поддомена <code>_acme-challenge.</code> , направив ее на свой сервер Angie. | —            | —                    |
| <i>Проверка хуками</i> | Реализовать внешний сервис (скрипт или приложение), который по команде Angie сможет внести изменения в DNS-зону или разместить специальный ответ на веб-сервере.                                               | —            | —                    |

- **Настройте SSL с использованием полученного сертификата и ключа:** Модуль делает сертификаты и ключи доступными в виде *встроенных переменных*, которые можно использовать в *конфигурации* для заполнения `ssl_certificate` и `ssl_certificate_key`.

Инструкции по настройке SSL см. в разделе `angie__ssl_config`.

### Совет

Процесс получения и обновления сертификатов зависит от работы многих служб и может занимать какое-то время. Запаситесь терпением, а в случае возникновения проблем или сомнений обратитесь к отладочному логу.

## Подробности реализации

Здесь ключи и сертификаты клиентов хранятся в *кодировке PEM* в соответствующих подкаталогах каталога, заданного с помощью параметра сборки `--http-acme-client-path`:

```
$ ls /var/lib/angie/acme/example/

account.key certificate.pem private.key
```

Клиенту АСМЕ требуется учетная запись на сервере СА. Для ее создания и управления ею клиент использует закрытый ключ (`account.key`); если ключа у него еще нет, ключ создается при запуске. Затем клиент использует его для регистрации учетной записи на сервере.

#### **Примечание**

Если у вас уже есть ключ учетной записи, поместите его в подкаталог клиента перед запуском для повторного использования учетной записи. Файл ключа также можно указать с помощью параметра `account_key` в `acme_client`.

Клиент ACME также использует отдельный ключ (`private.key`) для запросов на подпись сертификата (CSR); если нужно, этот ключ сертификата также создается автоматически при запуске.

При запуске клиент запрашивает сертификат, если его еще нет, подписывая и отправляя CSR для всех доменов, которыми он управляет, серверу СА. Сервер проверяет владение доменом путем *HTTP*- или *DNS-проверки* и выдает сертификат, который клиент сохраняет локально (`certificate.pem`).

Как сказано выше, сертификат будет единым для всех доменных имен, для которых используется один и тот же ACME-клиент, то есть потенциально может быть мультидоменным. Список всех имен, для которых выдан сертификат, см. в разделе *Subject Alternative Name* (SAN) полученного сертификата. Проверить его можно в командной строке, например:

```
$ openssl x509 -in certificate.pem -noout -text | grep -A5 "Subject Alternative Name"
```

Когда приближается завершение срока действия сертификата или изменяется список доменов, клиент подписывает и отправляет еще один CSR на сервер СА. Сервер снова проверяет владение и выдает новый сертификат, который клиент устанавливает локально, заменяя предыдущий.

В *конфигурации* полученный сертификат и соответствующий ключ доступны через префиксные переменные `$acme_cert_<имя>` и `$acme_cert_key_<имя>`. Их значения — содержимое соответствующих файлов, которое следует использовать с директивами `ssl_certificate` и `ssl_certificate_key`, например:

```
server {

 listen 443 ssl;

 server_name example.com www.example.com;
 acme example;

 ssl_certificate $acme_cert_example;
 ssl_certificate_key $acme_cert_key_example;
}
```

### HTTP-проверка

Проверка работает автоматически. Суть ее заключается в том, что ACME-сервер, получив запрос, запрашивает у клиента через HTTP особый файл-токен по адресу `/.well-known/acme-challenge/<TOKEN>`. Наш модуль ACME отслеживает такие запросы и самостоятельно обрабатывает их. Получив ожидаемый ответ с нужным содержимым, ACME-сервер подтверждает, что домен принадлежит клиенту.

## Пример конфигурации

Здесь ACME-клиент с именем `example` управляет сертификатами для `example.com` и `www.example.com` (учтите, что wildcard-сертификаты не поддерживаются при HTTP-проверке):

```
http {

 resolver 127.0.0.53; # требуется для директивы 'acme_client'

 acme_client example https://acme-v02.api.letsencrypt.org/directory;

 server {

 listen 80; # Может стоять в другом блоке 'server'
 # с другим списком доменов
 # или даже без него

 listen 443 ssl;

 server_name example.com www.example.com;
 acme example;

 ssl_certificate $acme_cert_example;
 ssl_certificate_key $acme_cert_key_example;
 }
}
```

Как уже отмечалось, порт 80 должен быть открыт для приема вызовов ACME по HTTP. Однако, как указывает этот пример, директива `listen` для этого порта может стоять в отдельном блоке `server`. Если существующего блока с такой директивой нет, можно ограничить новый блок одними только вызовами ACME:

```
server {

 listen 80;
 return 444; # Нет ответа, соединение закрыто
}
```

Почему это работает? Модуль перехватывает запросы к `/.well-known/acme-challenge/<TOKEN>` после чтения заголовков, но до выбора виртуального сервера и обработки директив `rewrite` и `location`. Такие перехваченные запросы обрабатываются, если значение `<TOKEN>` соответствует ожидаемому для конкретного вызова. Обращения к каталогу не будет; запрос полностью обрабатывается модулем.

## DNS-проверка

Проверка работает автоматически. Суть в том, что при получении запроса на сертификат ACME-сервер выполняет специальный DNS-запрос к поддомену `_acme-challenge`. проверяемого домена. После получения ожидаемого ответа ACME-сервер подтверждает, что домен принадлежит клиенту.

Наш модуль ACME отслеживает такие запросы и обрабатывает их автоматически, при условии, что ваши записи DNS настроены должным образом, чтобы указать сервер Angie в качестве авторитетного сервера имен для поддомена `_acme-challenge`..

Например, чтобы подтвердить домен `example.com`, используя сервер Angie с IP-адресом 93.184.215.14, DNS-конфигурация вашего домена должна включать следующие записи:

|                              |    |    |    |                 |
|------------------------------|----|----|----|-----------------|
| _acme-challenge.example.com. | 60 | IN | NS | ns.example.com. |
| ns.example.com.              | 60 | IN | A  | 93.184.215.14   |

Эта конфигурация делегирует разрешение DNS для \_acme-challenge.example.com на ns.example.com, обеспечивая доступность ns.example.com путем сопоставления с IP-адресом (93.184.215.14).

#### ⚠ Предупреждение

Распространение изменений NS-записей может занимать от нескольких минут до 48 часов в зависимости от TTL и DNS-провайдера. Рекомендуется проверить корректность настройки перед запросом сертификата.

Для проверки корректности настройки DNS можно использовать следующие команды:

```
$ dig NS _acme-challenge.example.com +short # Проверка NS-записи для поддомена _acme-
→challenge

ns.example.com.

$ dig A ns.example.com +short # Проверка A-записи для сервера имен

93.184.215.14

$ nc -zv 93.184.215.14 53 # Проверка доступности DNS-сервера на порту 53
```

Этот способ позволяет запрашивать wildcard-сертификаты, например сертификат, включающий запись \*.example.com в разделе *Subject Alternative Name* (SAN). Чтобы в явной форме запросить сертификат для поддомена, например www.example.com, следует отдельно подтвердить этот поддомен описанным выше способом.

#### ⚠ Предупреждение

Применимость данного сценария во многом зависит от возможностей, предоставляемых вашим DNS-провайдером; некоторые провайдеры не позволяют выполнять такие настройки.

### Пример конфигурации

В целом, конфигурация схожа с примером из предыдущего раздела. Нет необходимости в настройках, специфичных для HTTP; вместо этого достаточно установить challenge=dns для директивы acme\_client.

Здесь АСМЕ-клиент с именем example управляет сертификатами для example.com и \*.example.com:

```
http {

 resolver 127.0.0.53;

 acme_client example https://acme-v02.api.letsencrypt.org/directory
 challenge=dns;

 server {

 server_name example.com *.example.com;
 acme example;
```

```

 ssl_certificate $acme_cert_example;
 ssl_certificate_key $acme_cert_key_example;
 }
}

```

### Проверка с помощью хуков

В отличие от предыдущих способов, эта проверка требует дополнительных усилий. Здесь АСМЕ-сервер производит обычную *HTTP-проверку* или *DNS-проверку*, но обращается не к самому серверу Angie, а к внешнему сервису, которым сервер Angie управляет с помощью вызовов-хуков (*acme\_hook*). В свою очередь, этот сервис настраивает отдельный DNS- или HTTP-сервер, куда и направляются запросы АСМЕ-сервера.

Получив ожидаемый ответ от настроенного таким образом DNS- или HTTP-сервера, АСМЕ-сервер подтверждает, что домен принадлежит клиенту.

Angie вызывает внешний сервис в моменты, когда требуется обновление сертификатов по протоколу АСМЕ. Вызовы передаются путем проксирования запросов и данных на серверы FastCGI, SCGI и аналогичные с использованием директив *fastcgi\_pass*, *scgi\_pass* и т.д.

Сервис должен возвращать код 2xx, что можно сделать через заголовок **Status**. Любой другой код считается ошибкой, и обновление сертификата прекращается. При этом вывод от сервиса игнорируется.

### Пример конфигурации

Здесь настраивается АСМЕ-клиент **example** для подтверждения домена при помощи DNS-вызова, на что указывает параметр *challenge=dns* директивы *acme\_client*.

Блок **server** применяется ко всем поддоменам **example.com** (например, **\*.example.com**) и использует АСМЕ-клиент **example** для управления сертификатами, что указано в директиве *acme*.

Здесь также настроен именованный блок **location**, через который и производятся вызовы внешнего сервиса для DNS-верификации. Директива *acme\_hook* связывает сервер с АСМЕ-клиентом **example**. Запросы проксируются на FastCGI-сервер, работающий локально на порту 9000, с помощью директивы *fastcgi\_pass*. Директива *fastcgi\_param* передает во внешний сервис данные об АСМЕ-клиенте, хуке, типе вызова, домене, токене и ключе авторизации.

```

acme_client example https://acme-v02.api.letsencrypt.org/directory
 challenge=dns;

server {

 listen 80;

 server_name *.example.com;

 acme example;

 ssl_certificate $acme_cert_example;
 ssl_certificate_key $acme_cert_key_example;

 location @acme_hook_location {

 acme_hook example;

 fastcgi_pass localhost:9000;
 }
}

```

```

 fastcgi_param ACME_CLIENT $acme_hook_client;
 fastcgi_param ACME_HOOK $acme_hook_name;
 fastcgi_param ACME_CHALLENGE $acme_hook_challenge;
 fastcgi_param ACME_DOMAIN $acme_hook_domain;
 fastcgi_param ACME_TOKEN $acme_hook_token;
 fastcgi_param ACME_KEYAUTH $acme_hook_keyauth;

 include fastcgi.conf;
}
}

```

Пример соответствующего внешнего FastCGI-сервиса на Perl:

```

#!/usr/bin/perl

use strict; use warnings;

use FCGI;

my $socket = FCGI::OpenSocket(":9000", 5);
my $request = FCGI::Request(*STDIN, *STDOUT, *STDERR, \%ENV, $socket);

while ($request->Accept() >= 0) {
 print "\r\n";

 my $client = $ENV{ACME_CLIENT};
 my $hook = $ENV{ACME_HOOK};
 my $challenge = $ENV{ACME_CHALLENGE};
 my $domain = $ENV{ACME_DOMAIN};
 my $token = $ENV{ACME_TOKEN};
 my $keyauth = $ENV{ACME_KEYAUTH};

 if ($hook eq 'add') {

 DNS_set_TXT_record("_acme-challenge.$domain.", $keyauth);

 } elsif ($hook eq 'remove') {

 DNS_clear_TXT_record("_acme-challenge.$domain.");

 }
};

FCGI::CloseSocket($socket);

```

Здесь `DNS_set_TXT_record()` и `DNS_clear_TXT_record()` — функции, предположительно добавляющие и удаляющие TXT-записи в конфигурации некоего внешнего DNS-сервера, к которому и обратится ACME-сервер. В этих записях должны содержаться переданные сервером Angie данные, что позволит внешнему DNS-серверу успешно пройти проверку, аналогичную описанной в разделе *DNS-проверка*. Подробности реализации таких функций выходят за рамки этого руководства; так, например, передавать параметры можно и через URI запроса:

```

...

location @acme_hook_location {

 acme_hook example uri=/acme_hook/$acme_hook_name?domain=$acme_hook_domain&key=
 ↪$acme_hook_keyauth;
}

```

```
fastcgi_pass localhost:9000;

fastcgi_param REQUEST_URI $request_uri;
fastcgi_param ACME_CLIENT $acme_hook_client;
fastcgi_param ACME_CHALLENGE $acme_hook_challenge;
fastcgi_param ACME_TOKEN $acme_hook_token;

include fastcgi.conf;
}
```

Еще один пример настройки FastCGI-сервиса на PHP-FPM:

```
location @acme_hook_location {

 acme_hook example;
 root /var/www/dns;
 fastcgi_pass unix:/run/php-fpm/php-dns.sock;
 fastcgi_index hook.php;
 fastcgi_param SCRIPT_FILENAME /var/www/dns/hook.php;
 include fastcgi_params;

 fastcgi_param ACME_CLIENT $acme_hook_client;
 fastcgi_param ACME_HOOK $acme_hook_name;
 fastcgi_param ACME_CHALLENGE $acme_hook_challenge;
 fastcgi_param ACME_DOMAIN $acme_hook_domain;
 fastcgi_param ACME_TOKEN $acme_hook_token;
 fastcgi_param ACME_KEYAUTH $acme_hook_keyauth;
}
```

```
[dns]
listen = /run/php-fpm/php-dns.sock
listen.mode = 0666
user = angie
group = angie
chdir = /var/www/dns
...
```

Переданные параметры доступны в PHP через `$_SERVER['...']`.

### ACME в потоковом модуле

Потоковый модуль *ACME* позволяет автоматизировать выпуск и использование сертификатов для ТСП-трафика. Для его корректной работы необходимо сначала настроить HTTP-аналог: ACME-клиент должен быть объявлен в контексте `http`, а сам блок `stream` должен располагаться *после* блока `http` в конфигурации.



## Пример конфигурации

По умолчанию для получения сертификатов используется режим HTTP-проверки. Для него, как упоминалось в разделе *HTTP-проверка*, нужен HTTP-сервер, слушающий на порту 80:

```
HTTP-часть
http {

 resolver 127.0.0.53;

 # ACME-клиент для потоковой части
 acme_client example https://acme-v02.api.letsencrypt.org/directory;

 # Сервер для HTTP-проверки
 server {

 listen 80;
 return 444;
 }
}

Потоковая часть
stream {

 server {

 listen 12345 ssl;
 proxy_pass backend_upstream;

 ssl_certificate $acme_cert_example;
 ssl_certificate_key $acme_cert_key_example;

 server_name example.com www.example.com;
 acme example; # ссылка на ACME-клиент, определенный в HTTP-части
 }

 upstream backend_upstream {

 server 127.0.0.1:54321;
 }
}
```

Также можно использовать DNS-проверку, настроив `challenge=dns` в директиве `acme_client`; тогда сервер будет не нужен.

## Миграция с certbot

Если до перехода с nginx на Angie вы использовали `certbot` для получения и продления SSL-сертификатов от центра сертификации Let's Encrypt, выполните следующие шаги, чтобы перейти к использованию нашего модуля ACME.

Предположим, вы настроили сертификаты следующим образом:

```
$ sudo certbot --nginx -d example.com -d www.example.com
```

Автоматически созданная при этом конфигурация обычно находится в файле `/etc/nginx/sites-available/example.conf` и выглядит приблизительно так:

```
server {
 listen 80;
 server_name example.com www.example.com;
 return 301 https://$host$request_uri;
}

server {
 listen 443 ssl;
 server_name example.com www.example.com;

 root /var/www/example;
 index index.html;

 ssl_certificate /etc/letsencrypt/live/example.com/fullchain.pem;
 ssl_certificate_key /etc/letsencrypt/live/example.com/privkey.pem;
 include /etc/letsencrypt/options-ssl-nginx.conf;
 ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;
}
```

В примере выше выделены строки, которые потребуется изменить. В зависимости от ваших обстоятельств и предпочтений настройте *HTTP-проверку* или *DNS-проверку* с помощью модуля ACME.

Итоговая *конфигурация Angie* может выглядеть приблизительно так:

```
http {
 resolver 127.0.0.53;

 acme_client example https://acme-v02.api.letsencrypt.org/directory;

 server {
 listen 80;
 server_name example.com www.example.com;
 return 301 https://$host$request_uri;
 }

 server {
 listen 443 ssl;
 server_name example.com www.example.com;

 root /var/www/example;
 index index.html;

 acme example;

 ssl_certificate $acme_cert_example;
 ssl_certificate_key $acme_cert_key_example;
 }
}
```

Не забудьте перезагрузить конфигурацию после изменения:

```
$ sudo kill -HUP $(cat /run/angie.pid)
```

Убедившись, что эта конфигурация работает, вы можете удалить сертификаты `certbot`, а также отключить или целиком удалить его с сервера, если он больше нигде не используется, например:

```
$ sudo rm -rf /etc/letsencrypt

$ sudo systemctl stop certbot.timer
$ sudo systemctl disable certbot.timer
$ # -- уuu --
$ sudo rm /etc/cron.d/certbot

$ sudo apt remove certbot
$ # -- уuu --
$ sudo dnf remove certbot
```

## ГЛАВА 4

### Глобальная балансировка

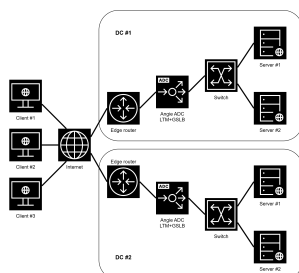
Глобальная балансировка, также GSLB (Global Server Load Balancing) или GTM (Global Traffic Management) – это сервис, который позволяет распределять запросы клиентов между несколькими дата-центрами на основе их доступности, производительности и географического положения.

Для работы глобальной балансировки настраивается делегирование зон на GSLB-серверы Angie ADC. Локальный DNS-сервер будет отправлять клиентские запросы для этих зон на один из GSLB-серверов. Сервер GSLB решает, какой дата-центр лучше обслужит запрос, выбирает сервер и передает его IP-адрес DNS-серверу. DNS-сервер отправляет этот IP-адрес клиенту, после чего клиент направляет свой запрос на новый сервер по полученному адресу.

GSLB балансирует запросы методом **round-robin** и **weighted round-robin**. Поддерживается балансировка на основе IP-адреса клиента (**clientip**). Далее будут добавлены и другие методы балансировки. В фоновом режиме проводятся проверки работоспособности апстрим-серверов, заданные в конфигурации локального балансировщика нагрузки.

Возможности GSLB в Angie ADC:

- Гибкая настройка балансировки: поддержка нескольких правил для одной зоны, поддержка одинаковых или разных конфигураций GSLB для дата-центров.
- Масштабируемость: простота добавления серверов в группы и новых правил в зоны.
- Отказоустойчивость: непрерывная доступность приложений даже при сбоях в одном из дата-центров, возможность настройки TTL.



В этом разделе:

- *Методы балансировки*
- *Настройка GSLB*
- *Параметры конфигурации*

Рис. 1: Рис. 1: Пример развертывания Angie ADC с GSLB в двух дата-центрах

## 4.1 Методы балансировки

### 4.1.1 Алгоритмы балансировки

| Алгоритм                        | Описание                                                                                                      |
|---------------------------------|---------------------------------------------------------------------------------------------------------------|
| <code>default</code>            | Запросы распределяются на первый сервер, отвечающий критериям, заданным в правиле. Используется по умолчанию. |
| <code>round_r</code>            | Запросы распределяются по серверам последовательно.                                                           |
| Взвешенный <code>round_r</code> | Запросы распределяются по серверам с учетом весов серверов ( <code>weighted</code> ).                         |

Поддерживается дополнительная настройка балансировки с учетом IP-адреса клиента (`clientip`). Далее будут поддерживаться и другие методы балансировки. Подробное описание параметров конфигурации GSLB см. в статье

*Параметры конфигурации.*

### 4.1.2 Проверки работоспособности

Модуль GSLB получает данные о состоянии апстрим-серверов от локального балансировщика на грузки Angie ADC. Предварительно необходимо настроить проверки работоспособности на локальном балансировщике.

#### Примечание

На данный момент используются только данные о доступности серверов (`{'IP-address': {'state': 'up'}}`). В следующих релизах будут учитываться и другие параметры *проверок работоспособности*.

Серверы временно удаляются из пула балансировки, если не прошли проверку. Также учитываются данные по времени ответа сервера, полученные во время проверок работоспособности. Если проверки работоспособности для сервера не заданы в конфигурации GSLB, сервер будет всегда считаться рабочими.

Рекомендуется включать проверки работоспособности для всех серверов, участвующих в балансировке, чтобы запросы перенаправлялись только на рабочие серверы. Исключение может составлять резервный сервер, используемый для ответа в случае, когда все остальные серверы не прошли проверки.

### 4.1.3 Примеры

#### 1. Балансировка на первый рабочий сервер

```
options:
 ttl: 1

zones:
 default.example.org:
 rules:
 - rule: default

rules:
 default:
 type: default
 members:
 - server: lb1-dc1-health
 - server: lb1-dc2-health

servers:
 lb1-dc1-health:
 addresses:
 - 172.22.11.1
 healthpeers: "http://172.22.11.1/api/http/upstreams/srv1/peers"
 healthinterval: 1s
 healthtimeout: 1s

 lb1-dc2-health:
 addresses:
 - 172.22.21.1
 healthpeers: "http://172.22.21.1/api/http/upstreams/srv1/peers"
 healthinterval: 1s
 healthtimeout: 1s
```

В этом примере:

- Зона `default.example.com` использует правило `default`, определенное в блоке `rules`.
- Для серверов заданы проверки работоспособности.
- Запросы будут перенаправляться на первый сервер, прошедший проверки (если проверки работоспособности не будут заданы, то запросы будут всегда перенаправлены на первый сервер в списке).

#### 2. Балансировка `round-robin` с проверками работоспособности

```
options:
 ttl: 1

zones:
 www.example.org:
 rules:
 - rule: rr
 members:
 - group: www

 test.example.com:
 rules:
```

```

- rule: rr

rules:
 rr:
 type: round_robin
 members:
 - server: www1
 - server: www2

groups:
 www:
 members:
 - server: www3
 - server: www4

servers:
 www1:
 addresses:
 - 172.22.0.10
 healthpeers: "http://172.22.0.10/api/http/upstreams/examplecom/peers"
 healthinterval: 10s
 healthtimeout: 2s

 www2:
 addresses:
 - 172.22.1.10
 healthpeers: "http://172.22.1.10/api/http/upstreams/examplecom/peers"
 healthinterval: 15s
 healthtimeout: 3s

 www3:
 addresses:
 - 172.22.0.10
 healthpeers: "http://172.22.0.10/api/http/upstreams/exampleorg/peers"
 healthinterval: 10s
 healthtimeout: 2s

 www4:
 addresses:
 - 172.22.1.10
 healthpeers: "http://172.22.1.10/api/http/upstreams/exampleorg/peers"
 healthinterval: 15s
 healthtimeout: 3s

```

В этом примере:

- Зоны `www.example.org` и `test.example.com` используют правило `rr`, определенное в блоке `rules`.
- Для зоны `www.example.org` переопределена группа серверов, которая будет использоваться в правиле `rr`.
- Для всех серверов заданы проверки работоспособности.
- Запросы будут распределяться последовательно на указанные серверы с учетом проверок работоспособности.

### 3. Балансировка round-robin с учетом весов серверов

```
options:
 ttl: 1

zones:
 wrd.example.org:
 rules:
 - rule: wrd

rules:
 wrd:
 type: round_robin
 weighted: true
 members:
 - server: www1
 - server: www2

servers:
 www1:
 addresses:
 - 172.22.0.10
 weight: 1

 www2:
 addresses:
 - 172.22.0.11
 weight: 2
```

В этом примере:

- Зона `wrd.example.org` использует правило `wrd-servers` (взвешенный round-robin), определенное в блоке `rules`. Если вес для сервера не указан, то при проверке конфигурации будет выдана ошибка.
- Серверы получают нагрузку в соответствии со своими весами (`www2` будет использоваться примерно в два раза чаще, чем `www1`).

### 4. Балансировка round-robin с проверками работоспособности и бэкап-сервером

```
options:
 ttl: 1

zones:
 multi.example.org:
 rules:
 - rule: rr-health
 - rule: fallback

rules:
 rr-health:
 type: round_robin
 members:
 - server: lb1-dc1-health
 - server: lb1-dc2-health

 fallback:
 members:
```



```
- server: fallback

servers:
 lb1-dc1-health:
 addresses:
 - 172.22.11.1
 healthpeers: "http://172.22.11.1/api/http/upstreams/srv1/peers"
 healthinterval: 1s
 healthtimeout: 1s

 lb1-dc2-health:
 addresses:
 - 172.22.21.1
 healthpeers: "http://172.22.21.1/api/http/upstreams/srv1/peers"
 healthinterval: 1s
 healthtimeout: 1s

 fallback:
 addresses:
 - 172.22.31.1
 - 172.22.31.2
```

В этом примере:

- Зона `multi.example.org` использует правила `rr-health` и `fallback`, определенные в блоке `rules`.
- Для серверов `lb1-dc1-health` и `lb1-dc2-health` заданы проверки работоспособности. Запросы будут распределяться последовательно на указанные серверы при прохождении проверок работоспособности.
- При непрохождении проверок серверами `lb1-dc1-health` и `lb1-dc2-health` запросы будут распределяться на резервный сервер `fallback`, у которого прописано два адреса. При этом оба прописанных адреса будут выдаваться в ответе одновременно.

## 5. Балансировка по IP-адресу клиента

```
options:
 ttl: 1

zones:
 clientip.example.org:
 rules:
 - rule: localhost
 matchall:
 - clientip: 127.0.0.0/8
 - rule: dc1
 matchany:
 - clientip: 172.17.0.0/16
 - clientip: 172.18.0.0/16
 - rule: dc2
 matchany:
 - clientip: 172.19.0.0/16
 - rule: rr

rules:
 dc1:
 members:
```

```

- server: www1

dc2:
 members:
 - server: www2

rr:
 type: round_robin
 members:
 - server: www1
 - server: www2

localhost:
 members:
 - server: localhost

servers:
 www1:
 addresses:
 - 172.22.0.10

 www2:
 addresses:
 - 172.22.1.10

 localhost:
 addresses:
 - 127.0.0.1

```

В этом примере:

- Зона `clientip.example.com` использует правила `localhost`, `dc1`, `dc2` и `rr`, определенные в блоке `rules`.
- Запросы клиентов из сети `127.0.0.0/8` распределяются на сервер `127.0.0.1` по правилу `localhost`.
- Запросы клиентов из сетей `172.17.0.0/16` и `172.18.0.0/16` распределяются на сервер `www1` по правилу `dc1`.
- Запросы клиентов из сети `172.19.0.0/16` распределяются на сервер `www2` по правилу `dc2`.
- Все остальные клиентские запросы распределяются по правилу `rr` последовательно на `www1` или `www2`.

## 4.2 Настройка GSLB

GSLB-сервис и локальный балансировщик нагрузки настраиваются на одном устройстве Angie ADC. Перед настройкой GSLB необходимо настроить локальную балансировку нагрузки и проверки работоспособности апстрим-серверов в каждом дата-центре.

Настройка GSLB состоит из следующих этапов:

1. Добавление зон, для которых требуется выполнять глобальную балансировку, в файл `Corefile`.

Например:

```
www.example.org {
 gslb
}
```

2. Настройка правил балансировки для этих зон в файле `gslbd.yaml`.

См. [примеры правил](#) и [описание параметров конфигурации](#).

3. Настройка делегирования заданных зон на GSLB-серверы Angie ADC на первичном DNS-сервере.

Пример для `www.example.org`:

```
adc-dc1 IN A 172.22.0.10 # IP-адрес первого GSLB-сервера Angie ADC
adc-dc2 IN A 172.22.1.10 # IP-адрес второго GSLB-сервера Angie ADC
www IN NS adc-dc1.example.org # делегирование зоны www.example.org на первый
→GSLB-сервер
www IN NS adc-dc2.example.org # делегирование зоны www.example.org на второй
→GSLB-сервер
```

## 4.2.1 Настройка конфигурации GSLB

Вы можете настроить конфигурацию GSLB в консоли Angie ADC. В конфигурацию входят следующие файлы:

- `/etc/angie-adc-gslb/Corefile` – задает домены (зоны), для которых будет настраиваться сервис GSLB;
- `/etc/angie-adc-gslb/gslbd.yaml` – задает конфигурацию балансировщика GSLB для указанных доменов (зон) в `yaml`-формате.

Конфигурация файла `gslbd.yaml` состоит из следующих блоков:

- *Options* – определяет общие параметры балансировки для всех зон.
- *Zones* – задает список доменов и определяет конфигурацию глобальной балансировки для этих доменов. Для каждого домена указывается доменное имя и список правил балансировки. Правила применяются по очереди до первого успешного ответа. Для правил можно дополнительно задать и серверы и группы серверов, к которым будут применяться эти правила. Если серверы для правила не заданы, то будут взяты серверы, указанные по умолчанию в соответствующем правиле.
- *Rules* – задает правила балансировки. Одни и те же правила можно использовать в нескольких зонах.
- *Servers* – задает список серверов, участвующих в балансировке. Поддерживаются IPv4-адреса. Одни и те же серверы можно использовать в нескольких зонах, группах и правилах.
- *Groups* – позволяет объединять несколько серверов или групп серверов в одну группу для удобства переиспользования в разных зонах и правилах.

Примеры конфигурации можно посмотреть в статье [Методы балансировки](#).

Чтобы открыть конфигурацию GSLB для просмотра и редактирования, выполните следующие действия:

1. Откройте консоль Angie ADC.
2. В правом верхнем углу выберите **Настройки** → **Глобальная балансировка**.

Откроется конфигурационный файл `Corefile`. В этом файле хранятся базовые настройки `coredns`, на основе которого построен сервис GSLB.

3. Укажите в `Corefile` домены, для которых будет настраиваться сервис GSLB, и нажмите Сохранить.
4. В раскрывающемся списке в верхней части окна выберите файл `gslbd.yaml`.  
Откроется файл с конфигурацией GSLB для указанных доменов.
5. Настройте в `gslbd.yaml` конфигурацию GSLB, добавив соответствующие доменам зоны и правила, и нажмите Сохранить. Подробное описание параметров см. в статье [Параметры конфигурации](#).  
Изменения будут применены сразу.

## 4.3 Параметры конфигурации

### 4.3.1 groups

Позволяет объединять несколько серверов или групп серверов в одну группу для удобства переиспользования в разных зонах и правилах. Для группы можно задать произвольное имя.

Пример:

```
groups:
 www:
 members:
 - server: www1
 - server: www2
 www3:
 members:
 - server: www4
 - server: www5
 - group: www
```

| Поле    | Описание                                                                               | Тип         | Обязательно |
|---------|----------------------------------------------------------------------------------------|-------------|-------------|
| имя     | Имя группы (произвольное)                                                              | string      | Да          |
| members | Задаёт список серверов или групп серверов, входящих в группу                           | members [ ] | Да          |
| server  | Имя сервера, входящего в группу (сами серверы задаются в блоке <code>servers</code> ). | string      | Нет         |
| group   | Имя группы серверов, входящих в группу                                                 | string      | Нет         |

### 4.3.2 zones

Задаёт список доменов и определяет конфигурацию глобальной балансировки для этих доменов. Для каждого домена указывается доменное имя и список правил балансировки. Правила применяются по очереди до первого успешного ответа. Для правил можно дополнительно задать и серверы и группы серверов, к которым будут применяться эти правила. Если серверы для правила не заданы, то будут взяты серверы, указанные по умолчанию в соответствующем правиле.

Пример:

```
zones:
 www.example.org:
 rules:
 - rule: wrr-servers
 members:
```

```
- group: www
- server: www11

clientip.example.org:
rules:
- rule: dc1
 matchany:
 - clientip: 172.17.0.0/16
 - clientip: 172.18.0.0/16
- rule: dc2
 matchany:
 - clientip: 172.19.0.0/16
- rule: rr
```

| Поле     | Описание                                                                                                                                                                                                                                    | Тип             | Обязательно |
|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|-------------|
| домен    | Имя домена, для которого выполняется глобальная балансировка. В будущих релизах будут поддержаны wildcard-домены.                                                                                                                           | string          | Да          |
| rules    | Задаёт список правил для домена. Для каждого домена требуется указать хотя бы одно правило.                                                                                                                                                 | rules<br>[ ]    | Да          |
| rule     | Указывает имя используемого правила балансировки (само правило задаётся в блоке <i>rules</i> ).                                                                                                                                             | string          | Да          |
| members  | Задаёт список серверов или групп серверов, для которых будет применяться правило. Если серверы для правила не указаны, то будут взяты серверы по умолчанию из соответствующего правила. Если серверы указаны, то они будут иметь приоритет. | members<br>[ ]  | Нет         |
| server   | Имя сервера, для которого будет применяться правило (сами серверы задаются в блоке <i>servers</i> ).                                                                                                                                        | string          | Нет         |
| group    | Имя группы серверов, для которой будет применяться правило (сами группы серверов задаются в блоке <i>groups</i> ).                                                                                                                          | string          | Нет         |
| matchany | Задаёт список условий, при выполнении любого из которых, запросы клиентов будут балансироваться по указанному правилу. Добавлено в версии 0.6.0.                                                                                            | matchany<br>[ ] | Нет         |
| matchall | Задаёт список условий, при выполнении которых, запросы клиентов будут балансироваться по указанному правилу. Добавлено в версии 0.6.0.                                                                                                      | matchall<br>[ ] | Нет         |
| clientip | Задаёт сеть как условие для балансировки по IP-адресу – если IP-адрес клиента входит в указанную сеть, то условие выполняется. Используется в параметрах <i>matchany</i> и <i>matchall</i> . Добавлено в версии 0.6.0.                      | CIDR            | Нет         |

### 4.3.3 options

Определяет общие параметры балансировки для всех зон.

Пример:

```
options:
 ttl: 1
```

| Поле             | Описание                                                                                                                                                                                                                                                        | Тип   | Обязательно |
|------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|-------------|
| <code>ttl</code> | Задаёт значение TTL (Time to Live) для DNS-записей в секундах — как часто должны обновляться записи; это позволяет оценить доступность серверов. По умолчанию установлено значение 1 (клиенты и кэширующие DNS-серверы должны обновлять записи каждую секунду). | число | Да          |

#### 4.3.4 rules

Определяет правила балансировки. Одни и те же правила можно использовать в нескольких зонах.

Пример:

```
rules:
 wrd-servers:
 type: round_robin
 weighted: true
 members:
 - server: www1
 - server: www2
```

| Поле                  | Описание                                                                                                                                                                                                                                                                                                                                                                                       | Тип            | Обязательно |
|-----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|-------------|
| <code>имя</code>      | Имя правила (произвольное).                                                                                                                                                                                                                                                                                                                                                                    | string         | Да          |
| <code>type</code>     | Задаёт тип балансировки правила. Если тип балансировки не указан, запрос будет направлен на первый сервер, от которого получен ответ.<br>Возможные значения: <ul style="list-style-type: none"> <li><code>round_robin</code> — запросы распределяются по серверам последовательно;</li> <li><code>default</code> — запрос направляется на первый сервер, от которого получен ответ.</li> </ul> | string         | Нет         |
| <code>weighted</code> | Определяет, будет ли учитываться вес сервера при балансировке <code>round_robin</code> . Если вес для сервера не указан, то при проверке конфигурации с <code>weighted</code> будет выдана ошибка.<br>Добавлено в версии 0.4.0.                                                                                                                                                                | boolean        | Нет         |
| <code>members</code>  | Задаёт список серверов или групп серверов, к которым по умолчанию применяется правило.                                                                                                                                                                                                                                                                                                         | members<br>[ ] | Да          |
| <code>server</code>   | Имя сервера, для которого будет применяться правило (сами серверы задаются в блоке <code>servers</code> ).                                                                                                                                                                                                                                                                                     | string         | Нет         |
| <code>group</code>    | Имя группы серверов, для которой будет применяться правило. (сами группы серверов задаются в блоке <code>groups</code> ).                                                                                                                                                                                                                                                                      | string         | Нет         |

### 4.3.5 servers

Задаёт список серверов, участвующих в балансировке. Поддерживаются IPv4-адреса. Одни и те же серверы можно использовать в нескольких зонах, группах и правилах.

Пример:

```
servers:
 www1:
 addresses:
 - 172.22.0.10
 weight: 2
 healthpeers: "http://172.22.0.10/api/http/upstreams/examplecom/peers"
 healthinterval: 10s
 healthtimeout: 2s

 www2:
 addresses:
 - 172.22.1.10
 weight: 1
 healthpeers: "http://172.22.1.10/api/http/upstreams/examplecom/peers"
 healthinterval: 15s
 healthtimeout: 3s
```

| Поле                  | Описание                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | Тип           | Обязательно                                         |
|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------|-----------------------------------------------------|
| <b>имя</b>            | Имя сервера (произвольное).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | <b>string</b> | Да                                                  |
| <b>addresses</b>      | Задаёт список адресов сервера, на которые будут направляться запросы при балансировке. Если указано несколько адресов, запросы будут направляться на все адреса одновременно.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | <b>string</b> | Да                                                  |
| <b>weight</b>         | Задаёт вес сервера.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | <b>число</b>  | Да, если в правиле задан параметр <b>weighted</b> . |
| <b>healthpath</b>     | <p>Задаёт путь до REST API, по которому модуль GSLB будет получать информацию о состоянии апстримов от локального балансировщика нагрузки на этом сервере:</p> <ul style="list-style-type: none"> <li>для http-балансировки: <code>http://&lt;server_address&gt;/api/http/upstreams/&lt;server_name&gt;/peers</code></li> <li>для stream-балансировки: <code>http://&lt;server_address&gt;/api/stream/upstreams/&lt;server_name&gt;/peers</code></li> </ul> <div>  <b>Примечание</b><br/> Проверки работоспособности предварительно необходимо настроить на локальном балансировщике Angie ADC. </div> | <b>string</b> | Нет                                                 |
| <b>healthinterval</b> | Задаёт интервал между проверками. При указании интервала необходимо указывать единицу измерения <b>s</b> (секунды) или <b>ms</b> (миллисекунды), например: <b>5s</b> , <b>0.5s</b> , <b>200ms</b> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | <b>string</b> | Да, если указан параметр <b>healthpeers</b> .       |
| <b>healthtimeout</b>  | Задаёт таймаут на каждую проверку. При указании таймаута необходимо указывать единицу измерения <b>s</b> (секунды) или <b>ms</b> (миллисекунды), например: <b>5s</b> , <b>0.5s</b> , <b>200ms</b> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | <b>string</b> | Да, если указан параметр <b>healthpeers</b> .       |



## ГЛАВА 5

### IP-маршрутизация

Angie ADC поддерживает ручную настройку режимов высокой доступности на устройстве Angie ADC по протоколам BGP и OSPF (динамическая маршрутизация) и с VRRP (статическая маршрутизация) в вариантах Active-Standby и Active-Active. Также поддерживается использование протокола BFD для уменьшения времени реакции на события и механизма RHI для оценки состояния апстримов.

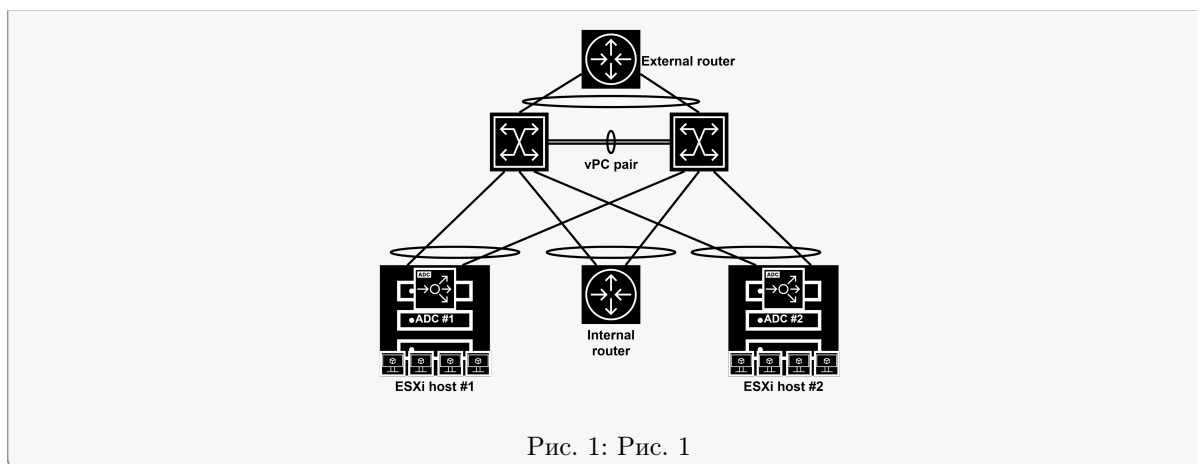
В этом разделе:

- *BGP-маршрутизация*
- *OSPF-маршрутизация*
- *VRRP-маршрутизация*
- *Использование протокола BFD для уменьшения времени реакции*
- *Настройка RHI*

#### **Примечание**

Рекомендуется резервировать все компоненты инфраструктуры. Отказ сопутствующей инфраструктуры (например выход из строя вышестоящего маршрутизатора или сбой серверов) выходит за рамки решения высокой доступности Angie ADC.

Пример отказоустойчивой инфраструктуры:



## 5.1 BGP-маршрутизация

В этом разделе описана настройка BGP-маршрутизации для обеспечения высокой доступности в режиме резервирования (Active-Standby) и в режиме распределения нагрузки (Active-Active). Для настройки используется *интерфейс командной строки (CLI) Angie ADC*.

В этом разделе:

- Режим резервирования (Active-Standby)
- Режим распределения нагрузки (Active-Active)

### 5.1.1 Резервирование с помощью протокола BGP (Active-Standby)

#### Введение

В этом разделе описана настройка высокой доступности с использованием протокола BGP в режиме резервирования (Active-Standby). Для настройки используется *интерфейс командной строки (CLI) Angie ADC*.

В качестве протокола динамической маршрутизации используется BGPv4 для адресного семейства IPv4 unicast. Отказоустойчивость в рамках IPv6 unicast настраивается аналогично.

В рассматриваемом примере Angie ADC #1 выполняет функции активной системы балансировки трафика, а Angie ADC #2 остается в резерве.

Пример типового участка сети с системами балансировки Angie ADC (ADC #1 и ADC #2) представлен на схеме:



Рис. 1

Добавим IP-адресацию:

- В сегменте сети, куда подключаются External router, ADC #1 и ADC #2, будет использоваться подсеть 192.168.0.0/24.

- В сегменте сети, куда подключаются Internal router, ADC #1 и ADC #2, будет использоваться подсеть 192.168.1.0/24.
- Четвертый октет адреса для каждого устройства (в рамках соответствующей подсети) показан на схеме ниже.

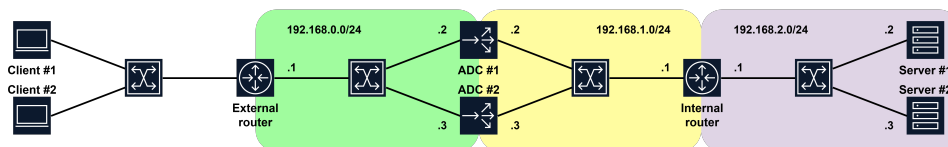


Рис. 2

Адресация клиентов не приводится, так как в данном случае мы будем считать, что их адреса формально не определены, то есть они могут подключаться из любой сети, включая сеть интернет. Для этих целей External router будет анонсировать маршрут по умолчанию по BGP для ADC #1 и ADC #2. В данном документе будем использовать только одну автономную систему BGP №65001.

#### **Примечание**

Автономная система BGP — логическое объединение устройств в сети для маршрутизации через протокол BGP.

### **BGP-сессии**

Устанавливаемые BGP-сессии представлены на схеме:

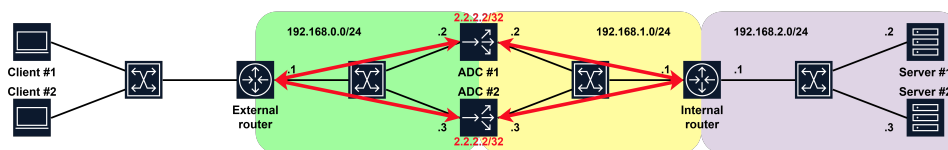


Рис. 3

Красным цветом на схеме отмечены IP-адреса виртуальных серверов. Это те адреса, к которым подключаются клиенты снаружи. Системы балансировки должны анонсировать маршруты до этих виртуальных серверов в сторону External router, причем маршрут через ADC #1 должен быть более предпочтительным. Добиться этого можно разными способами, манипулируя атрибутами пути в BGP.

Так как все BGP-взаимодействие производится строго в рамках одной автономной системы, мы будем использовать атрибут Local preference (чем больше, тем лучше) для выбора наиболее оптимального пути. Трафик от серверов в сторону клиента должен проходить через ту же систему балансировки, что и трафик от клиентов к серверам. Добиться этого можно также путем манипулирования значением атрибута Local preference для префикса 0.0.0.0/0.

## Обработка типовых сбоев

Ниже описаны сценарии работы Angie ADC при сбоях.

### Полный отказ ADC #1

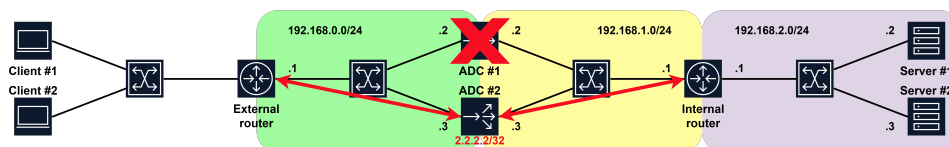


Рис. 4

При полном отказе или отключении от сети ADC #1 разрываются обе BGP-сессии с ADC #1. Через ADC #1 не проходят никакие объявления маршрутов. Остается связь по BGP только с ADC #2. Переключение на резервную систему ADC #2 производится автоматически. Никакие дополнительные настройки не требуются.

### Отказ внешнего интерфейса ADC #1

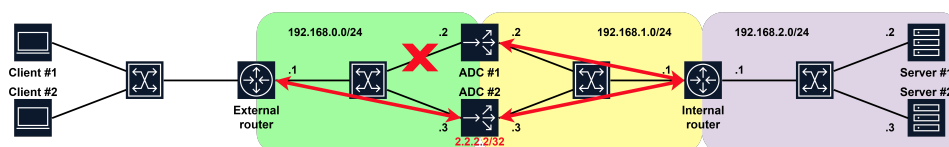


Рис. 5

В этом случае разрывается BGP-сессия между ADC #1 и External router. ADC #1 теперь не может объявить маршрут до адреса виртуального сервера в сторону External router. Несмотря на наличие BGP-сессии между ADC #1 и Internal router, ADC #1 не отправляет маршрут по умолчанию в сторону Internal router, так как сама не получает его от External router из-за сбоя линка. Таким образом, Internal router видит маршрут по умолчанию только через ADC #2, а External router получает маршрут до адреса виртуального сервера только через полностью работающий ADC #2.

### Отказ линка между ADC #1 и Internal router

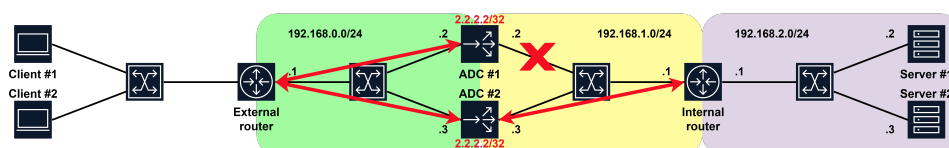


Рис. 6

Если BGP-сессия между External router и ADC #1 сохраняется, то ADC #1 будет продолжать анонсировать маршрут до адреса виртуального сервера, хотя реальные серверы для ADC #1 уже недоступны. Таким образом, мы получаем классический black hole для трафика. Чтобы избежать этого, анонсирование маршрутов до виртуальных серверов должно быть условным, то есть ADC #1 анонсирует маршрут до адреса виртуального сервера, только если в BGP-таблице присутствуют префиксы, соответствующие подсетям реальных серверов.

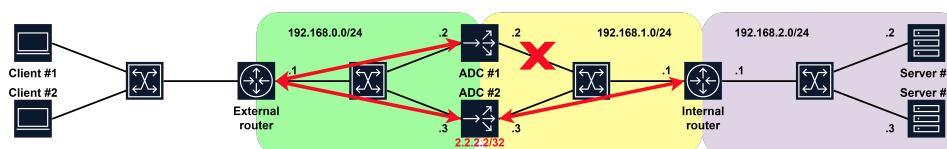


Рис. 7

## Настройка маршрутизации

Приведем ключевые настройки маршрутизации для External router, Internal router, ADC #1 и ADC #2.

### Ключевые настройки маршрутизации для External router

```
hostname external
interface GigabitEthernet0/0
 ip address 192.168.0.1 255.255.255.0
!Интерфейс в сторону систем балансировки
router bgp 65001
 neighbor 192.168.0.2 remote-as 65001
!Сессия с ADC#1
 neighbor 192.168.0.2 default-originate
!Анонсирование маршрута по умолчанию
 neighbor 192.168.0.3 remote-as 65001
!Сессия с ADC#2
 neighbor 192.168.0.3 default-originate
!Анонсирование маршрута по умолчанию
end
```

### Ключевые настройки маршрутизации для Internal router

```
hostname internal
interface GigabitEthernet0/0
 ip address 192.168.1.1 255.255.255.0
!Интерфейс в сторону систем балансировки
interface GigabitEthernet1/0
 ip address 192.168.2.1 255.255.255.0
!Интерфейс в сторону подсети с реальными серверами
router bgp 65001
 network 192.168.2.0 mask 255.255.255.0
!Добавление информации о подсети с реальными серверами в BGP
 neighbor 192.168.1.2 remote-as 65001
!Сессия с ADC#1
 neighbor 192.168.1.3 remote-as 65001
!Сессия с ADC#2
end
```

## Ключевые настройки маршрутизации Angie ADC #1

```
hostname adc1
!Имя устройства
ip prefix-list vip seq 5 permit 2.2.2.2/32
!Префикс-лист, описывающий адреса виртуальных серверов
ip prefix-list servers seq 5 permit 192.168.2.0/24
!Префикс-лист, описывающий адреса реальных серверов.
interface ens37
 ip address 192.168.0.2/24
!Интерфейс в сторону External router
exit
interface ens38
 ip address 192.168.1.2/24
!Интерфейс в сторону Internal router
exit
interface lo
 ip address 2.2.2.2/32
!Loopback-интерфейс для адресов виртуальных серверов.
exit
router bgp 65001
 neighbor 192.168.0.1 remote-as 65001
!Сессия с External router
 neighbor 192.168.1.1 remote-as 65001
!Сессия с Internal router
address-family ipv4 unicast
 network 2.2.2.2/32
!Добавление префикса для адреса виртуального сервера в BGP
 neighbor 192.168.0.1 route-reflector-client
!Разрешаем передачу префиксов между двумя iBGP-клиентами
 neighbor 192.168.0.1 route-map fromclients in
!Манипуляции над маршрутами, полученными от External router, меняем local preference
 neighbor 192.168.0.1 route-map 2clients out
!Манипуляции над маршрутами, отправляемыми в сторону External router, меняем local
↪preference
 neighbor 192.168.0.1 advertise-map 2clients exist-map servers
!Условная отправка префиксов: если префикс есть в route-map servers, то будет отправка
 neighbor 192.168.1.1 route-reflector-client
!Разрешаем передачу префиксов между двумя iBGP-клиентами
 neighbor 192.168.1.1 next-hop-self force
!Подменяем адрес атрибута next-hop на собственный даже для отраженных префиксов
exit-address-family
exit
route-map 2clients permit 10
 match ip address prefix-list vip
 set local-preference 150
!Для префиксов, соответствующих адресам виртуальных серверов, увеличиваем Local
↪preference
exit
route-map 2clients deny 20
!Блокируем отправку всех остальных префиксов
exit
route-map fromclients permit 10
 set local-preference 150
!Увеличиваем Local preference для маршрутов, получаемых от External router
exit
route-map servers permit 10
```

```
match ip address prefix-list servers
!Условие для условной отправки: если в BGP-таблице есть маршрут,
!соответствующий префикс-листу servers, то условие считается выполненным
exit
end
```

## Ключевые настройки маршрутизации Angie ADC #2

```
hostname adc2
!Имя устройства
ip prefix-list vip seq 5 permit 2.2.2.2/32
!Префикс-лист, описывающий адреса виртуальных серверов
ip prefix-list servers seq 5 permit 192.168.2.0/24
!Префикс-лист, описывающий адреса реальных серверов.
interface ens37
 ip address 192.168.0.3/24
!Интерфейс в сторону External router
exit
interface ens38
 ip address 192.168.1.3/24
!Интерфейс в сторону Internal router
exit
interface lo
 ip address 2.2.2.2/32
!Loopback-интерфейс для адресов виртуальных серверов.
exit
router bgp 65001
 neighbor 192.168.0.1 remote-as 65001
!Сессия с External router
 neighbor 192.168.1.1 remote-as 65001
!Сессия с Internal router
address-family ipv4 unicast
 network 2.2.2.2/32
!Добавление префикса для адреса виртуального сервера в BGP
 neighbor 192.168.0.1 route-reflector-client
!Разрешаем передачу префиксов между двумя iBGP-клиентами
 neighbor 192.168.0.1 route-map fromclients in
!Манипуляции над маршрутами, полученными от External router, меняем local preference
 neighbor 192.168.0.1 route-map 2clients out
!Манипуляции над маршрутами, отправляемыми в сторону External router, меняем local
↪preference
 neighbor 192.168.0.1 advertise-map 2clients exist-map servers
!Условная отправка префиксов: если префикс есть в route-map servers, то будет отправка
 neighbor 192.168.1.1 route-reflector-client
!Разрешаем передачу префиксов между двумя iBGP-клиентами
 neighbor 192.168.1.1 next-hop-self force
!Подменяем адрес атрибута next-hop на собственный даже для отраженных префиксов
 exit-address-family
exit
route-map 2clients permit 10
 match ip address prefix-list vip
 set local-preference 50
!Для префиксов, соответствующих адресам виртуальных серверов, уменьшаем Local
↪preference
exit
route-map 2clients deny 20
```

```
!Блокируем отправку всех остальных префиксов
exit
route-map fromclients permit 10
 set local-preference 50
!Уменьшаем Local preference для маршрутов, получаемых от External router
exit
route-map servers permit 10
 match ip address prefix-list servers
!Условие для условной отправки: если в BGP-таблице есть маршрут,
!соответствующий префикс-листу servers, то условие считается выполненным
exit
end
```

## Проверка работы

Выполним проверку работы системы маршрутизации, когда ADC #1 и ADC #2 находятся в полностью работоспособном состоянии.

### Примечание

В листингах ниже отсутствуют префиксы, не относящиеся к обсуждаемым вопросам отказоустойчивости.

Маршрутизатор External router видит маршрут к адресу виртуального сервера через обе системы балансировки, однако маршрут через ADC #1 является более предпочтительным:

```
external#sho ip ro
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
 D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
 N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
 E1 - OSPF external type 1, E2 - OSPF external type 2
 i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
 ia - IS-IS inter area, * - candidate default, U - per-user static route
 o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
 + - replicated route, % - next hop override

Gateway of last resort is not set

 2.0.0.0/32 is subnetted, 1 subnets
B 2.2.2.2 [200/0] via 192.168.0.2, 02:06:11
 192.168.0.0/24 is variably subnetted, 2 subnets, 2 masks
C 192.168.0.0/24 is directly connected, GigabitEthernet0/0
L 192.168.0.1/32 is directly connected, GigabitEthernet0/0
external#sho ip bg
BGP table version is 9, local router ID is 11.11.11.11
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
 r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
 x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found
```

| Network        | Next Hop    | Metric | LocPrf | Weight | Path |
|----------------|-------------|--------|--------|--------|------|
| 0.0.0.0        | 0.0.0.0     |        |        | 0      | i    |
| *>i 2.2.2.2/32 | 192.168.0.2 | 0      | 150    | 0      | i    |
| * i            | 192.168.0.3 | 0      | 50     | 0      | i    |



Маршрутизатор Internal router видит маршрут по умолчанию через обе системы балансировки, однако маршрут через ADC #1 является более предпочтительным:

```
internal#sho ip ro
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
 D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
 N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
 E1 - OSPF external type 1, E2 - OSPF external type 2
 i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
 ia - IS-IS inter area, * - candidate default, U - per-user static route
 o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
 + - replicated route, % - next hop override

Gateway of last resort is 192.168.1.2 to network 0.0.0.0

B* 0.0.0.0/0 [200/0] via 192.168.1.2, 02:12:37
 192.168.1.0/24 is variably subnetted, 2 subnets, 2 masks
C 192.168.1.0/24 is directly connected, GigabitEthernet0/0
L 192.168.1.1/32 is directly connected, GigabitEthernet0/0
 192.168.2.0/24 is variably subnetted, 2 subnets, 2 masks
C 192.168.2.0/24 is directly connected, GigabitEthernet1/0
L 192.168.2.1/32 is directly connected, GigabitEthernet1/0
internal#sho ip bgp
BGP table version is 12, local router ID is 33.33.33.33
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
 r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
 x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

 Network Next Hop Metric LocPrf Weight Path
*>i 0.0.0.0 192.168.1.2 0 150 0 i
* i 192.168.1.1/32 192.168.1.3 0 50 0 i
*> 192.168.2.0/24 0.0.0.0 0 32768 i
```

Отключим теперь на ADC #1 интерфейс в сторону Internal router и проверим маршрутизацию:

```
adc1# sho int bri
Interface Status VRF Addresses

ens37 up default 192.168.0.2/24
ens38 down default 192.168.1.2/24
lo up default 2.2.2.2/32
adc1# conf t
adc1(config)# int ens38
adc1(config-if)# shut
adc1# sho int bri
Interface Status VRF Addresses

ens37 up default 192.168.0.2/24
ens38 down default 192.168.1.2/24
lo up default 2.2.2.2/32
```

Изучим теперь таблицу маршрутизации и BGP-таблицу на маршрутизаторе External router. Несмотря на наличие BGP-сессии до каждой из систем балансировки, только по одной сессии прилетает маршрут на адрес виртуального сервера – от ADC #2:

```
external#sho ip bg su
BGP router identifier 11.11.11.11, local AS number 65001
```

```

BGP table version is 8, main routing table version 8
2 network entries using 296 bytes of memory
2 path entries using 128 bytes of memory
2/1 BGP path/bestpath attribute entries using 272 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
BGP using 696 total bytes of memory
BGP activity 3/1 prefixes, 4/2 paths, scan interval 60 secs
Neighbor V AS MsgRcvd MsgSent TblVer InQ OutQ Up/Down State/
->PfxRcd
192.168.0.2 4 1 156 150 8 0 0 00:07:29 0
192.168.0.3 4 1 157 150 8 0 0 00:07:29 1
external#sho ip bg
BGP table version is 8, local router ID is 11.11.11.11
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
 r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
 x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

 Network Next Hop Metric LocPrf Weight Path
 0.0.0.0 0.0.0.0 0 50 0 i
*>i 2.2.2.2/32 192.168.0.3 0 50 0 i
external#sho ip ro
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
 D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
 N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
 E1 - OSPF external type 1, E2 - OSPF external type 2
 i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
 ia - IS-IS inter area, * - candidate default, U - per-user static route
 o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
 + - replicated route, % - next hop override
Gateway of last resort is not set
2.0.0.0/32 is subnetted, 1 subnets
B 2.2.2.2 [200/0] via 192.168.0.3, 00:04:36
 192.168.0.0/24 is variably subnetted, 2 subnets, 2 masks
C 192.168.0.0/24 is directly connected, GigabitEthernet0/0
L 192.168.0.1/32 is directly connected, GigabitEthernet0/0

```

И теперь маршрутная информация со стороны Internal router:

```

internal#sho ip bgp
BGP table version is 8, local router ID is 33.33.33.33
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
 r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
 x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

 Network Next Hop Metric LocPrf Weight Path
*>i 0.0.0.0 192.168.1.3 0 50 0 i
*> 192.168.2.0/24 0.0.0.0 0 32768 i
internal#sho ip ro
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
 D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
 N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
 E1 - OSPF external type 1, E2 - OSPF external type 2
 i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2

```

```

 ia - IS-IS inter area, * - candidate default, U - per-user static route
 o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
 + - replicated route, % - next hop override
Gateway of last resort is 192.168.1.3 to network 0.0.0.0
B* 0.0.0.0/0 [200/0] via 192.168.1.3, 00:00:49
C 192.168.1.0/24 is directly connected, GigabitEthernet0/0
L 192.168.1.1/32 is directly connected, GigabitEthernet0/0
C 192.168.2.0/24 is directly connected, GigabitEthernet1/0
L 192.168.2.1/32 is directly connected, GigabitEthernet1/0

```

## Применение с IP transparency

Все описанное выше относится к режиму работы Angie ADC с включенной опцией IP transparency. При включенной опции IP Transparency Angie ADC сохраняет исходный адрес отправителя в IP-пакетах, позволяя серверам на сетевом уровне различать клиентские подключения.

Очень часто администраторы систем балансировки отключают эту опцию, в этом случае Angie ADC будет выполнять не только dNAT, но sNAT над трафиком, приходящим от клиентов. Если система балансировки Angie ADC выполняет sNAT, то в этом случае маршрутизация несколько упрощается, так как в этом случае нет необходимости анонсировать на Internal router маршрут по умолчанию. Если ADC #1 и ADC #2 выполняют sNAT в разные адреса, то в этом случае нет необходимости заботиться о том, с какими значениями Local preference (и иными атрибутами пути) маршруты до соответствующих адресов попадают на Internal router.

## 5.1.2 Распределение нагрузки с помощью протокола BGP (Active-Active)

### Введение

В этом разделе описана настройка высокой доступности с использованием протокола BGP в режиме распределения нагрузки (Active-Active). Для настройки используется *командная строка Angie ADC CLI*.

В качестве протокола динамической маршрутизации используется BGPv4 для адресного семейства IPv4 unicast. Отказоустойчивость в рамках IPv6 unicast настраивается аналогично.

В рассматриваемом примере устройства Angie ADC #1 и Angie ADC #2 выполняют функции активной системы балансировки трафика одновременно. Для динамического исключения одного из Angie ADC из работы в случае сбоя (технические проблемы, потери связности и так далее) можно использовать *механизм RHI (Route Health Injection)*.

Пример типового участка сети с системами балансировки Angie ADC (ADC #1 и ADC #2) представлен на схеме:



Рис. 1

Добавим IP-адресацию:

- В сегменте сети, куда подключаются External router, ADC #1 и ADC #2, будет использоваться подсеть 192.168.0.0/24.
- В сегменте сети, куда подключаются Internal router, ADC #1 и ADC #2, будет использоваться подсеть 192.168.1.0/24.

- Четвертый октет адреса для каждого устройства (в рамках соответствующей подсети) показан на схеме ниже.

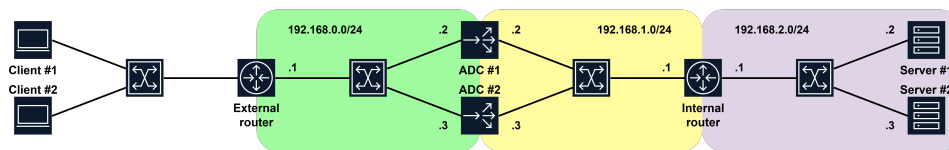


Рис. 2

Адресация клиентов не приводится, так как в данном случае мы будем считать, что их адреса формально не определены, то есть они могут подключаться из любой сети, включая сеть интернет. Для этих целей External router будет анонсировать маршрут по умолчанию по BGP для ADC #1 и ADC #2.

## BGP-сессии

Устанавливаемые BGP-сессии представлены на схеме:

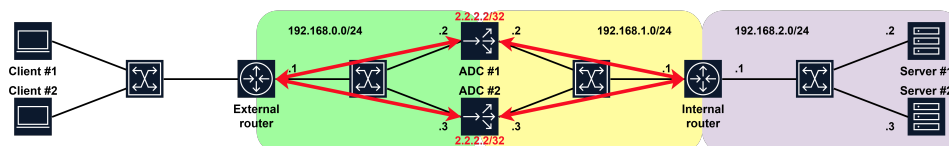


Рис. 3

Красным цветом на схеме отмечены IP-адреса виртуальных серверов. Это те адреса, к которым подключаются клиенты снаружи. ADC #1 и ADC #2 должны анонсировать маршруты до этих виртуальных серверов в сторону External router, причем маршруты через ADC #1 и ADC #2 должны иметь схожие атрибуты пути, чтобы маршрутизатор External router мог выполнить распределение трафика между ними.

Трафик внутри пользовательской сессии от серверов должен возвращаться строго через ту же систему балансировки, через которую он проходил до этого. Добиться этого можно с помощью механизма SNAT, когда адрес клиента подменяется на адрес интерфейса системы балансировки.

## Обработка типовых сбоев

Ниже описаны сценарии работы Angie ADC при сбоях.

### Полный отказ ADC #1

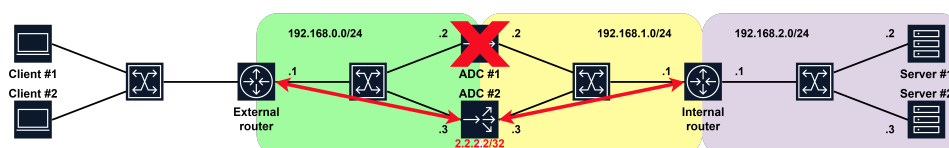


Рис. 4

При полном отказе или отключении от сети ADC #1 разрываются обе BGP-сессии с ADC #1. Через ADC #1 не проходят никакие объявления маршрутов. Остается связь по BGP только с ADC #2. Отключение ADC #1 происходит автоматически. Никакие дополнительные действия не требуются.

## Отказ внешнего интерфейса ADC #1

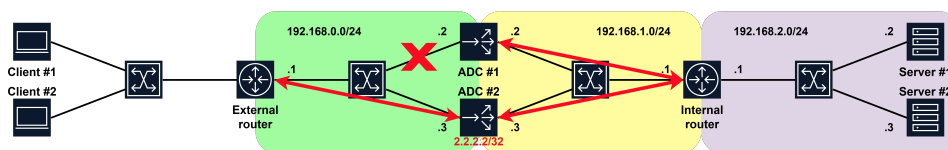


Рис. 5

В этом случае разрывается BGP-сессия между ADC #1 и External router. ADC #1 теперь не может объявить маршрут до адреса виртуального сервера в сторону External router, а пользовательский трафик будет распределяться только на систему балансировки ADC #2. Так как клиентские сессии устанавливаются снаружи, трафик всех вновь устанавливаемых сессий будет проходить только через ADC #2.

## Отказ линка между ADC #1 и Internal router

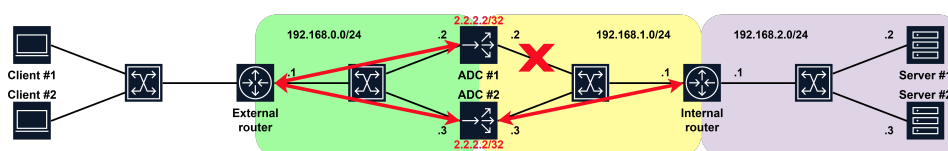


Рис. 6

Если BGP-сессия между External router и ADC #1 сохраняется, то ADC #1 будет продолжать анонсировать маршрут до адреса виртуального сервера, хотя реальные серверы для ADC #1 уже недостижимы. Таким образом, мы получаем классический black hole для трафика. Чтобы избежать этого, анонсирование маршрутов до виртуальных серверов должно быть условным. То есть ADC #1 должен отозвать маршрут до адреса виртуального сервера при падении тестов RHI:

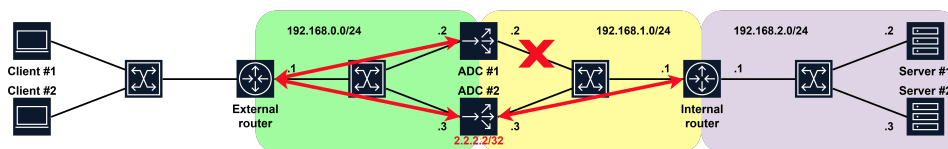


Рис. 7

## Настройка маршрутизации

При настройке BGP-маршрутизации необходимо указывать собственный номер автономной системы.

### Примечание

Автономная система BGP — логическое объединение устройств в сети для маршрутизации через протокол BGP.

Ниже приведены примеры трех распространенных топологий настройки маршрутизации во внешнем сегменте сети. Настройка маршрутизации во внутреннем сегменте выполняется аналогично.

#### **Примечание**

В список обязательных команд для некоторых платформ необходимо включить команду задания ограничения на максимальное количество путей, между которыми производится ECMP-балансировка.

Пример такой команды для маршрутизаторов Cisco на базе IOS и IOS-XE:

```
platform loadbalance max-paths 2
```

### **Топология #1**

В топологии #1 все три устройства (External router, ADC #1 и ADC #2) принадлежат одной автономной системе:

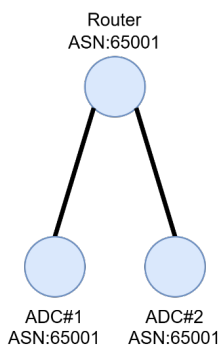


Рис. 8

Обычно на ADC #1 не нужны маршруты, объявляемые со стороны ADC #2. Если для конкретной сети вам нужно, чтобы ADC#1 знал маршруты, которые объявляет ADC #2 (или наоборот), то тогда на маршрутизаторе необходимо создать RR-кластер (Route Reflector cluster). Это позволит обойти ограничение на передачу префиксов, накладываемое правилом BGP split-horizon.

#### **Примечание**

Правило BGP split-horizon запрещает отправлять полученные от iBGP-соседа маршруты в сторону другого iBGP-соседа, защищая таким образом от заикливания маршрутов.

Создание RR-кластера поможет также в ситуации, когда перед маршрутизатором External router расположен другой маршрутизатор, находящийся в той же самой автономной системе.

### **Ключевые настройки маршрутизации для External router**

На примере маршрутизатора Cisco Systems на базе IOS или IOS-XE:

```
!Имя устройства
hostname external
interface GigabitEthernet0/0
!Интерфейс в сторону систем балансировки
ip address 192.168.0.1 255.255.255.0
router bgp 65001
!Сессия с ADC#1
neighbor 192.168.0.2 remote-as 65001
!Сессия с ADC#2
neighbor 192.168.0.3 remote-as 65001
```

```

address-family ipv4
!Включение соседа
neighbor 192.168.0.2 activate
!Включение соседа
neighbor 192.168.0.3 activate
!Анонсирование маршрута по умолчанию
neighbor 192.168.0.2 default-originate
!Анонсирование маршрута по умолчанию
neighbor 192.168.0.3 default-originate
!Добавление соседа в RR-кластер
neighbor 192.168.0.2 route-reflector-client
!Добавление соседа в RR-кластер
neighbor 192.168.0.3 route-reflector-client
!Указание максимального количества iBGP путей, между которыми будет выполняться
↪балансировка
maximum-paths ibgp 2
end

```

### Ключевые настройки маршрутизации Angie ADC #1

```

!Имя устройства
hostname adc1
!Интерфейс в сторону External router
interface ens33
ip address 192.168.0.2/24
exit
!Loopback-интерфейс для адресов виртуальных серверов
interface lo
ip address 2.2.2.2/32
ip address 3.3.3.3/32
exit
router bgp 65001
!Сессия с External router
neighbor 192.168.0.1 remote-as 65001
address-family ipv4 unicast
!Добавление префикса для адреса виртуального сервера в BGP
network 2.2.2.2/32
network 3.3.3.3/32
exit-address-family
exit
end

```

### Ключевые настройки маршрутизации Angie ADC #2

```

!Имя устройства
hostname adc2
!Интерфейс в сторону External router
interface ens33
ip address 192.168.0.3/24
exit
!Loopback-интерфейс для адресов виртуальных серверов
interface lo
ip address 2.2.2.2/32
ip address 4.4.4.4/32

```

```
exit
router bgp 65001
!Сессия с External router
neighbor 192.168.0.1 remote-as 65001
address-family ipv4 unicast
!Добавление префикса для адреса виртуального сервера в BGP
network 2.2.2.2/32
network 4.4.4.4/32
exit-address-family
exit
end
```

## Проверка работы

```
external#sho ip bgp
BGP table version is 4, local router ID is 192.168.0.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found
Network Next Hop Metric LocPrf Weight Path
0.0.0.0 0.0.0.0 0 i
*mi 2.2.2.2/32 192.168.0.2 0 100 0 i
*>i 192.168.0.3 0 100 0 i
external#sho ip ro
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2
i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
ia - IS-IS inter area, * - candidate default, U - per-user static route
o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
+ - replicated route, % - next hop override
2.0.0.0/32 is subnetted, 1 subnets
B 2.2.2.2 [200/0] via 192.168.0.2, 00:12:23
[200/0] via 192.168.0.3, 00:12:23
192.168.0.0/24 is variably subnetted, 2 subnets, 2 masks
C 192.168.0.0/24 is directly connected, GigabitEthernet0/0
L 192.168.0.1/32 is directly connected, GigabitEthernet0/0
```

Выводы просмотревых команд сокращены для краткости. Для тестирования работы RR-кластера мы завели на ADC #1 адрес 3.3.3.3/32, а на ADC #2 — 4.4.4.4/32.

Ниже представлены BGP-таблица и таблица маршрутизации от ADC #2:

```
adc2# sho ip bgp
BGP table version is 8, local router ID is 4.4.4.4, vrf id 0
Default local pref 100, local AS 65001
Status codes: s suppressed, d damped, h history, u unsorted, * valid, > best, =_
↔multipath,
i internal, r RIB-failure, S Stale, R Removed
Nexthop codes: @NNN nexthop's vrf id, < announce-nh-self
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found
Network Next Hop Metric LocPrf Weight Path
```



```
*>i 0.0.0.0/0 192.168.0.1 0 100 0 i
*> 2.2.2.2/32 0.0.0.0(adc2) 0 32768 i
* i 192.168.0.2 0 100 0 i
*>i 3.3.3.3/32 192.168.0.2 0 100 0 i
*> 4.4.4.4/32 0.0.0.0(adc2) 0 32768 i
Displayed 4 routes and 5 total paths
adc2# sho ip bgp 3.3.3.3
BGP routing table entry for 3.3.3.3/32, version 7
Paths: (1 available, best #1, table default)
Not advertised to any peer
Local
192.168.0.2 from 192.168.0.2 (3.3.3.3)
Origin IGP, metric 0, localpref 100, valid, internal, bestpath-from-AS Local, best_
→(First path received)
Originator: 3.3.3.3, Cluster list: 192.168.0.1
Last update: Mon Apr 21 19:47:21 2025
adc2# sho ip ro
Codes: K - kernel route, C - connected, L - local, S - static,
R - RIP, O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,
T - Table, v - VNC, V - VNC-Direct, F - PBR,
f - OpenFabric, t - Table-Direct,
> - selected route, * - FIB route, q - queued, r - rejected, b - backup
t - trapped, o - offload failure
B 0.0.0.0/0 [200/0] via 192.168.0.1, ens33, weight 1, 00:03:35
L * 2.2.2.2/32 is directly connected, lo, 00:40:12
C>* 2.2.2.2/32 is directly connected, lo, 00:40:12
B>* 3.3.3.3/32 [200/0] via 192.168.0.2, ens33, weight 1, 00:03:35
L * 4.4.4.4/32 is directly connected, lo, 00:40:12
C>* 4.4.4.4/32 is directly connected, lo, 00:40:12
C>* 192.168.0.0/24 is directly connected, ens33, 00:40:13
L>* 192.168.0.3/32 is directly connected, ens33, 00:40:13
```

## Топология #2

В топологии #2 все системы балансировки (Angie ADC #1 и Angie ADC #2) принадлежат одной автономной системе, а маршрутизатор External router – другой:

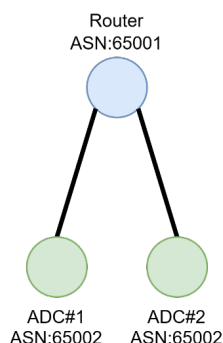


Рис. 9

Обычно на ADC #1 не нужны маршруты, объявляемые со стороны ADC #2. Если для конкретной сети вам нужно, чтобы ADC #1 знал маршруты, которые объявляет ADC #2 (или наоборот), то тогда на системе балансировки нагрузки надо разрешить прием маршрутов, содержащих собственный номер автономной системы в атрибуте AS-PATH.

## Ключевые настройки маршрутизации для External router

На примере маршрутизатора Cisco Systems на базе IOS или IOS-XE:

```
!Имя устройства
hostname external
!Интерфейс в сторону систем балансировки
interface GigabitEthernet0/0
ip address 192.168.0.1 255.255.255.0
router bgp 65001
!Сессия с ADC#1
neighbor 192.168.0.2 remote-as 65002
!Сессия с ADC#2
neighbor 192.168.0.3 remote-as 65002
address-family ipv4
!Включение соседа
neighbor 192.168.0.2 activate
!Включение соседа
neighbor 192.168.0.3 activate
!Анонсирование маршрута по умолчанию
neighbor 192.168.0.2 default-originate
!Анонсирование маршрута по умолчанию
neighbor 192.168.0.3 default-originate
!Указание максимального количества eBGP путей, между которыми будет выполняться↵
↵балансировка
maximum-paths 2
end
```

## Ключевые настройки маршрутизации Angie ADC #1

```
!Имя устройства
hostname adc1
!Интерфейс в сторону External router
interface ens33
ip address 192.168.0.2/24
exit
!Loopback-интерфейс для адресов виртуальных серверов
interface lo
ip address 2.2.2.2/32
ip address 3.3.3.3/32
exit
router bgp 65002
!Сессия с External router
neighbor 192.168.0.1 remote-as 65001
address-family ipv4 unicast
!Добавление префикса для адреса виртуального сервера в BGP
network 2.2.2.2/32
!Прием маршрутов, AS-PATH которых содержит собственный ASN
!Опция origin указывает на то, что приниматься будут только те маршруты,
!которые порождены в автономной системе с собственным ASN
neighbor 192.168.0.1 allowas-in origin
exit-address-family
exit
end
```

## Ключевые настройки маршрутизации Angie ADC #2

```
!Имя устройства
hostname adc2
!Интерфейс в сторону External router
interface ens33
ip address 192.168.0.3/24
exit
!Loopback-интерфейс для адресов виртуальных серверов
interface lo
ip address 2.2.2.2/32
ip address 4.4.4.4/32
exit
router bgp 65002
!Сессия с External router
neighbor 192.168.0.1 remote-as 65001
address-family ipv4 unicast
!Добавление префикса для адреса виртуального сервера в BGP
network 2.2.2.2/32
!Прием маршрутов, AS-PATH которых содержит собственный ASN.
!Опция origin указывает на то, что приниматься будут только те маршруты,
!которые порождены в автономной системе с собственным ASN
neighbor 192.168.0.1 allowas-in origin
exit-address-family
exit
end
```

## Проверка работы

```
external#sho ip bgp
BGP table version is 11, local router ID is 192.168.0.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found
Network Next Hop Metric LocPrf Weight Path
0.0.0.0 0.0.0.0 0 i
*> 2.2.2.2/32 192.168.0.2 0 0 65002 i
*m 192.168.0.3 0 0 65002 i
external#sho ip ro
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2
i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
ia - IS-IS inter area, * - candidate default, U - per-user static route
o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
+ - replicated route, % - next hop override
Gateway of last resort is not set
2.0.0.0/32 is subnetted, 1 subnets
B 2.2.2.2 [20/0] via 192.168.0.2, 00:13:18
[20/0] via 192.168.0.3, 00:13:18
192.168.0.0/24 is variably subnetted, 2 subnets, 2 masks
C 192.168.0.0/24 is directly connected, GigabitEthernet0/0
L 192.168.0.1/32 is directly connected, GigabitEthernet0/0
```

Выводы просмотревших команд сокращены для краткости.

Для тестирования команды `allowas-in` мы завели на ADC #1 адрес 3.3.3.3/32, а на ADC #2 — 4.4.4.4/32.

Ниже представлены BGP-таблица и таблица маршрутизации от ADC #2:

```
adc2# sho ip bgp
BGP table version is 4, local router ID is 2.2.2.2, vrf id 0
Default local pref 100, local AS 65002
Status codes: s suppressed, d damped, h history, u unsorted, * valid, > best, =_
↔multipath,
i internal, r RIB-failure, S Stale, R Removed
Nexthop codes: @NNN nexthop's vrf id, < announce-nh-self
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found
Network Next Hop Metric LocPrf Weight Path
*> 0.0.0.0/0 192.168.0.1 0 65001 i
*> 2.2.2.2/32 0.0.0.0(adc2) 0 32768 i
* 192.168.0.2 0 65001 65002 i
*> 3.3.3.3/32 192.168.0.2 0 65001 65002 i
*> 4.4.4.4/32 0.0.0.0(adc2) 0 32768 i
* 192.168.0.1 0 65001 65002 i
Displayed 4 routes and 6 total paths
adc2# sho ip ro
Codes: K - kernel route, C - connected, L - local, S - static,
R - RIP, O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,
T - Table, v - VNC, V - VNC-Direct, F - PBR,
f - OpenFabric, t - Table-Direct,
> - selected route, * - FIB route, q - queued, r - rejected, b - backup
t - trapped, o - offload failure
B 0.0.0.0/0 [20/0] via 192.168.0.1, ens33, weight 1, 00:07:26
L * 2.2.2.2/32 is directly connected, lo, 00:12:14
C>* 2.2.2.2/32 is directly connected, lo, 00:12:14
B>* 3.3.3.3/32 [20/0] via 192.168.0.2, ens33, weight 1, 00:01:02
L * 4.4.4.4/32 is directly connected, lo, 00:06:43
C>* 4.4.4.4/32 is directly connected, lo, 00:06:43
C>* 192.168.0.0/24 is directly connected, ens33, 00:12:15
L>* 192.168.0.3/32 is directly connected, ens33, 00:12:15
```

### Топология #3

Все три устройства (External router, ADC #1 и ADC #2) принадлежат разным автономным системам:

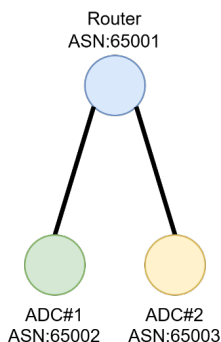


Рис. 10

Обычно маршрутизаторы Cisco не балансируют трафик до подсети, известной через соседей из разных автономных систем. Чтобы обойти данное ограничение, необходимо изменить поведение по умолчанию маршрутизатора BGP с помощью следующей команды:

```
bgp bestpath as-path multipath-relax
```

### Ключевые настройки маршрутизации для External router

На примере маршрутизатора Cisco Systems на базе IOS или IOS-XE:

```
!Имя устройства
hostname external
!Интерфейс в сторону систем балансировки
interface GigabitEthernet0/0
ip address 192.168.0.1 255.255.255.0
router bgp 65001
!Команда, разрешающая выполнять балансировку для префиксов, полученных из разных ASN
bgp bestpath as-path multipath-relax
!Сессия с ADC#1
neighbor 192.168.0.2 remote-as 65002
!Сессия с ADC#2
neighbor 192.168.0.3 remote-as 65003
address-family ipv4
!Включение соседа
neighbor 192.168.0.2 activate
!Включение соседа
neighbor 192.168.0.3 activate
!Анонсирование маршрута по умолчанию
neighbor 192.168.0.2 default-originate
!Анонсирование маршрута по умолчанию
neighbor 192.168.0.3 default-originate
!Указание максимального количества eBGP путей, между которыми будет выполняться ↵
↵балансировка
maximum-paths 2
end
```

### Ключевые настройки маршрутизации Angie ADC #1

```
!Имя устройства
hostname adc1
!Интерфейс в сторону External router
interface ens33
ip address 192.168.0.2/24
exit
!Loopback-интерфейс для адресов виртуальных серверов
interface lo
ip address 2.2.2.2/32
exit
router bgp 65002
!Сессия с External router
neighbor 192.168.0.1 remote-as 65001
address-family ipv4 unicast
!Добавление префикса для адреса виртуального сервера в BGP
network 2.2.2.2/32
```

```
exit-address-family
exit
end
```

## Ключевые настройки маршрутизации Angie ADC #2

```
!Имя устройства
hostname adc2
!Интерфейс в сторону External router
interface ens33
ip address 192.168.0.3/24
exit
!Loopback-интерфейс для адресов виртуальных серверов
interface lo
ip address 2.2.2.2/32
exit
router bgp 65003
!Сессия с External router
neighbor 192.168.0.1 remote-as 65001
address-family ipv4 unicast
!Добавление префикса для адреса виртуального сервера в BGP
network 2.2.2.2/32
exit-address-family
exit
end
```

## Проверка работы

```
external#sho ip bgp
BGP table version is 11, local router ID is 192.168.0.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found
Network Next Hop Metric LocPrf Weight Path
0.0.0.0 0.0.0.0 0 0 i
*> 2.2.2.2/32 192.168.0.2 0 0 65002 i
*m 192.168.0.3 0 0 65003 i
external#sho ip ro
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2
i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
ia - IS-IS inter area, * - candidate default, U - per-user static route
o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
+ - replicated route, % - next hop override
Gateway of last resort is not set
2.0.0.0/32 is subnetted, 1 subnets
B 2.2.2.2 [20/0] via 192.168.0.2, 00:10:22
[20/0] via 192.168.0.3, 00:16:07
192.168.0.0/24 is variably subnetted, 2 subnets, 2 masks
```

C 192.168.0.0/24 is directly connected, GigabitEthernet0/0  
L 192.168.0.1/32 is directly connected, GigabitEthernet0/0

Выводы просмотревых команд сокращены для краткости.

### Применение с IP transparency

Все описанное выше относится к режиму работы Angie ADC с выключенной опцией IP transparency. В этом случае Angie ADC будет выполнять проксирование клиентских запросов с подменой адресов источника и получателя. Если Angie ADC подменяет адрес клиента, то при маршрутизации нет необходимости анонсировать на Internal router маршрут по умолчанию. Если ADC #1 и ADC #2 используют разные адреса для подмены адреса клиента, то нет необходимости заботиться о том, с какими значениями Local preference (и иными атрибутами пути) маршруты до соответствующих адресов попадают на Internal router.

При включении опции IP Transparency Angie ADC будет сохранять исходный адрес отправителя в IP-пакетах, позволяя серверам на сетевом уровне различать клиентские подключения.

### Заключение

Система балансировки нагрузки Angie ADC может работать в любой из перечисленных топологий. Каждый вариант имеет свои особенности настройки и работы, о чем следует помнить при проектировании и настройке информационной системы.

## 5.2 OSPF-маршрутизация

В этом разделе описана настройка OSPF-маршрутизации для обеспечения высокой доступности в режиме резервирования (Active-Standby) и в режиме распределения нагрузки (Active-Active). Для настройки используется *интерфейс командной строки (CLI) Angie ADC*.

В этом разделе:

- *Режим резервирования (Active-Standby)*
- *Режим распределения нагрузки (Active-Active)*

### 5.2.1 Резервирование с помощью протокола OSPF (Active-Standby)

#### Введение

В этом разделе описана настройка высокой доступности с использованием протокола OSPF в режиме резервирования (Active-Standby). Для настройки OSPF используется *интерфейс командной строки (CLI) Angie ADC*.

В качестве протокола динамической маршрутизации используется OSPFv2 для адресного семейства IPv4 unicast. Отказоустойчивость в рамках IPv6 unicast настраивается аналогично.

Для обеспечения отказоустойчивости нужны две системы балансировки Angie ADC — основная и резервная. Резервная система Angie ADC будет принимать нагрузку при отказе основной.

Пример типового участка сети с системами балансировки Angie ADC (ADC #1 и ADC #2):



Рис. 1

Добавим *IP-адресацию*:

- В сегменте сети, куда подключаются External router, ADC #1 и ADC #2, будет использоваться подсеть 192.168.0.0/24.
- В сегменте сети, куда подключаются Internal router, ADC #1 и ADC #2, будет использоваться подсеть 192.168.1.0/24.
- Четвертый октет адреса для каждого устройства (в рамках соответствующей подсети) показан на схеме ниже.

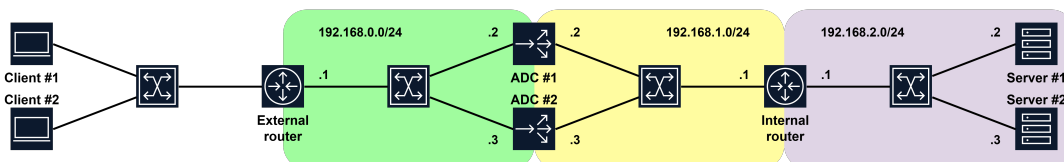


Рис. 2

*Адресация клиентов* не приводится, так как в данном случае мы будем считать, что их адреса формально не определены, то есть, они могут подключаться из любой сети, включая сеть интернет. Для этих целей External router будет анонсировать маршрут по умолчанию по OSPF для ADC #1 и ADC #2. Мы будем использовать OSPF-домен, включающий в себя четыре устройства: External router, Internal router, ADC #1 и ADC #2.

Красным текстом на схеме ниже отмечены *IP-адреса виртуальных серверов*. Это те адреса, к которым подключаются клиенты снаружи. Системы балансировки Angie ADC должны анонсировать маршруты до этих виртуальных серверов в сторону External router, причем маршрут через ADC #1 должен быть более предпочтительным. Добиться этого можно разными способами, например, манипулируя значением метрики внешнего маршрута в OSPF. Чем больше значение метрики, тем менее предпочтительный маршрут. В данном документе мы будем вручную задавать косты для интерфейсов системы балансировки.

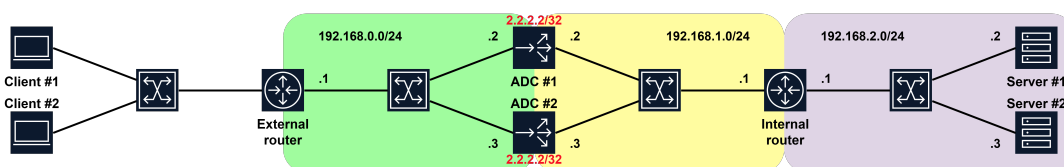


Рис. 3

Трафик от серверов в сторону клиента должен проходить через ту же систему балансировки, что и трафик от клиентов к серверам. Добиться этого можно также путем манипулирования значением метрики для префикса 0.0.0.0/0. Манипуляции с маршрутом по умолчанию нужны при включении опции IP transparency. В противном случае трафик и так будет возвращаться через ту же систему балансировки, так как поведение ADC напоминает операцию sNAT.



## Обработка типовых сбоев

Ниже описаны сценарии работы Angie ADC при сбоях.

### Полный отказ ADC #1

В этом случае оба OSPF-соседства с ADC #1 разрываются, никакие объявления маршрутов через ADC #1 не пролетают. Остается связь по OSPF только с ADC #2. То есть, переключение на резервную систему производится полностью автоматически, никакие дополнительные настройки не требуются.

Красными стрелками обозначены рабочие OSPF-соседства:

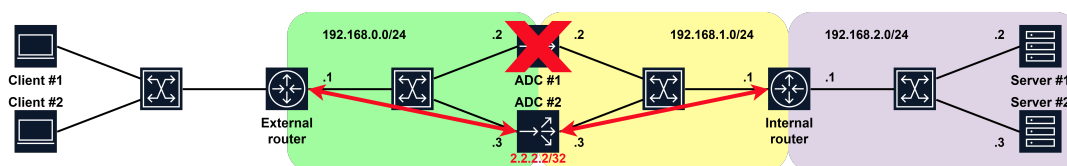


Рис. 4

### Отказ внешнего интерфейса ADC #1

Предлагаемая схема не предполагает возможности отслеживания частичных отказов и реагирования на них. Пример такого отказа представлен на схеме ниже.

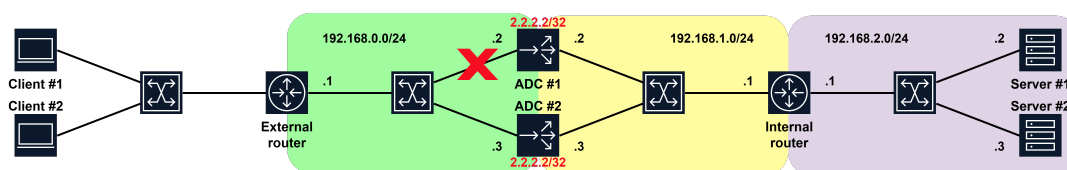


Рис. 5

В случае обозначенного на схеме отказа линка разрывается OSPF-сессия между ADC #1 и External router. ADC #1 теперь не может объявить маршрут до адреса виртуального сервера в сторону External router напрямую. Internal router рассчитывает маршрут по умолчанию только через ADC #2, а External router считает маршрут до адреса виртуального сервера только через полностью работающий ADC #2.

### Отказ линка между ADC #1 и Internal router

Отказ линка между ADC #1 и Internal router более сложная ситуация: если OSPF-соседство между External router и ADC #1 сохраняется, то ADC #1 будет продолжать анонсировать маршрут до адреса виртуального сервера, хотя реальные сервера для системы балансировки ADC #1 уже недоступны.

Таким образом, мы получаем классический пример black hole для трафика:

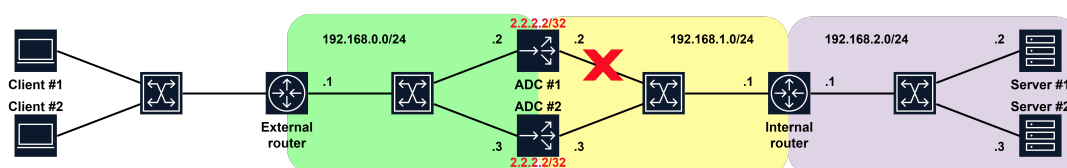


Рис. 6

Чтобы избежать этой проблемы, необходимо, чтобы ситуация отказа только одного из линков была невозможной, т.е. нужно обеспечить отказоустойчивое подключение к сети для хоста виртуализации, на котором работает ADC.

Рекомендованная схема подключения:

- Объединить в одну LAG-группу все физические интерфейсы хоста виртуализации и использовать этот виртуальный LAG-интерфейс в качестве транка, то есть передавать по нему агрегированными несколько виртуальных сетей (один VLAN для external сети, второй VLAN для internal сети).
- Для резервирования коммутаторов можно использовать технологию vPC или стекирование, пример представлен на схеме ниже.

Из соображений простоты на схеме не отображено резервирование внешнего и внутреннего маршрутизаторов.

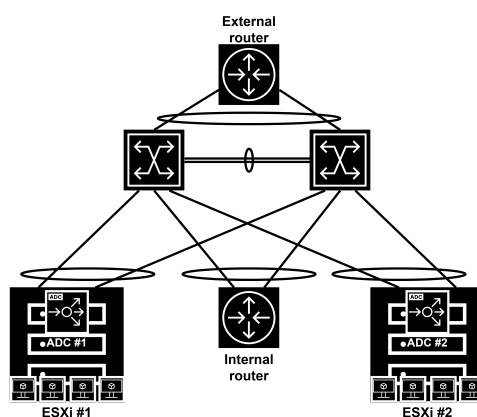


Рис. 7

## Настройки маршрутизации

Приведем ключевые настройки маршрутизации для External router, Internal router, ADC #1 и ADC #2.

### Ключевые настройки маршрутизации для External router

```
hostname external
!Имя устройства
interface GigabitEthernet0/0
ip address 192.168.0.1 255.255.255.0
!Интерфейс в сторону систем балансировки
router ospf 1
network 192.168.0.0 0.0.0.255 area 0
!Включение OSPF на интерфейсе в сторону систем балансировки
default-information originate always
!Анонсирование маршрута по умолчанию
end
```

## Ключевые настройки маршрутизации для Internal router

```
hostname internal
!Имя устройства
interface GigabitEthernet0/0
 ip address 192.168.1.1 255.255.255.0
!Интерфейс в сторону систем балансировки
interface GigabitEthernet1/0
 ip address 192.168.2.1 255.255.255.0
!Интерфейс в сторону подсети с реальными серверами
router ospf 1
 redistribute connected subnets
!добавление в OSPF информации о подсети с серверами
 network 192.168.1.0 0.0.0.255 area 0
!Включение OSPF на интерфейсе в сторону систем балансировки
end
```

## Ключевые настройки маршрутизации системы балансировки ADC #1

```
hostname adc1
!Имя устройства
interface ens33
!Интерфейс в сторону маршрутизатора external router
 description external
 ip address 192.168.0.2/24
 ip ospf 1 area 0
!Включение OSPF на интерфейсе
 ip ospf cost 10
!Назначение стоимости интерфейса в OSPF вручную
exit
interface ens37
!Интерфейс в сторону маршрутизатора internal router
 description internal
 ip address 192.168.1.2/24
 ip ospf 1 area 0
!Включение OSPF на интерфейсе
exit
interface lo
!Loopback-интерфейс для назначения адресов виртуальных серверов
 ip address 2.2.2.2/32
!Адрес тестового виртуального сервера
exit
ip prefix-list c2o seq 5 permit 2.2.2.2/32
!Префикс-лист, который используется для отбора префиксов для добавления в OSPF
route-map c2o permit 10
 match ip address prefix-list c2o
!Route-map с помощью которой производится отбор префиксов для добавления в OSPF
exit
router ospf 1
!Настройки процесса маршрутизации OSPF
 ospf router-id 2.2.2.2
!Необходимо вручную задать идентификатор маршрутизатора для OSPF
!Если этого не сделать, то автоматика может выбрать одинаковые для ADC#1 и ADC#2
 redistribute connected metric 10 route-map c2o
!Добавление информации о подключенных сетях в OSPF
!Метрика 10, чтобы стать более предпочтительным next-hop по сравнению с ADC#2
```

```
exit
end
```

## Ключевые настройки маршрутизации системы балансировки ADC #2

```
hostname adc2
!Имя устройства
interface ens33
!Интерфейс в сторону маршрутизатора external router
description external
ip address 192.168.0.3/24
ip ospf 1 area 0
!Включение OSPF на интерфейсе
ip ospf cost 100
!Назначение стоимости интерфейса в OSPF вручную
exit
interface ens37
!Интерфейс в сторону маршрутизатора internal router
description internal
ip address 192.168.1.3/24
ip ospf 1 area 0
!Включение OSPF на интерфейсе
exit
interface lo
!Loopback-интерфейс для назначения адресов виртуальных серверов
ip address 2.2.2.2/32
!Адрес тестового виртуального сервера
exit
ip prefix-list c2o seq 5 permit 2.2.2.2/32
!Префикс-лист, который используется для отбора префиксов для добавления в OSPF
route-map c2o permit 10
match ip address prefix-list c2o
!Route-map с помощью которой производится отбор префиксов для добавления в OSPF
exit
router ospf 1
!Настройки процесса маршрутизации OSPF
ospf router-id 22.22.22.22
!Необходимо вручную задать идентификатор маршрутизатора для OSPF
!Если этого не сделать, то автоматика может выбрать одинаковые для ADC#1 и ADC#2
redistribute connected metric 20 route-map c2o
!Добавление информации о подключенных сетях в OSPF
!Метрика 20, чтобы стать менее предпочтительным next-hop по сравнению с ADC#1
exit
end
```

## Проверка работы

Выполним проверку работы системы маршрутизации, когда ADC #1 и ADC #2 находятся в полностью работоспособном состоянии.

### Примечание

В листингах ниже отсутствуют префиксы, не относящиеся к обсуждаемым вопросам отказоустойчивости.

Маршрутизатор External router видит маршрут к адресу виртуального сервера через обе системы балансировки, однако маршрут через ADC #1 является более предпочтительным.

```
external#sho ip ro os
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
 D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
 N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
 E1 - OSPF external type 1, E2 - OSPF external type 2
 i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
 ia - IS-IS inter area, * - candidate default, U - per-user static route
 o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
 + - replicated route, % - next hop override
Gateway of last resort is not set
 2.0.0.0/32 is subnetted, 1 subnets
O E2 2.2.2.2 [110/10] via 192.168.0.2, 00:41:28, GigabitEthernet0/0
external#sho ip os da ex 2.2.2.2
 OSPF Router with ID (192.168.0.1) (Process ID 1)
 Type-5 AS External Link States
Routing Bit Set on this LSA in topology Base with MTID 0

LS age: 898
Options: (No TOS-capability, No DC, Upward)
LS Type: AS External Link
Link State ID: 2.2.2.2 (External Network Number)
Advertising Router: 2.2.2.2
LS Seq Number: 80000003
Checksum: 0xAA0C
Length: 36
Network Mask: /32
 Metric Type: 2 (Larger than any link state path)
 MTID: 0
 Metric: 10
 Forward Address: 0.0.0.0
 External Route Tag: 0

LS age: 885
Options: (No TOS-capability, No DC, Upward)
LS Type: AS External Link
Link State ID: 2.2.2.2 (External Network Number)
Advertising Router: 22.22.22.22
LS Seq Number: 80000003
Checksum: 0xB4A7
Length: 36
Network Mask: /32
 Metric Type: 2 (Larger than any link state path)
 MTID: 0
 Metric: 20
```

```
Forward Address: 0.0.0.0
External Route Tag: 0
```

Маршрутизатор Internal router видит маршрут по умолчанию (путь через ADC #1 является более предпочтительным).

```
internal#sho ip ro os
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
 D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
 N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
 E1 - OSPF external type 1, E2 - OSPF external type 2
 i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
 ia - IS-IS inter area, * - candidate default, U - per-user static route
 o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
 + - replicated route, % - next hop override
Gateway of last resort is 192.168.1.2 to network 0.0.0.0
O*E2 0.0.0.0/0 [110/1] via 192.168.1.2, 00:43:21, GigabitEthernet0/0
internal#sho ip os da ext 0.0.0.0
 OSPF Router with ID (192.168.2.1) (Process ID 1)
 Type-5 AS External Link States
Routing Bit Set on this LSA in topology Base with MTID 0
LS age: 992
Options: (No TOS-capability, DC, Upward)
LS Type: AS External Link
Link State ID: 0.0.0.0 (External Network Number)
Advertising Router: 192.168.0.1
LS Seq Number: 80000003
Checksum: 0x2521
Length: 36
Network Mask: /0
 Metric Type: 2 (Larger than any link state path)
 MTID: 0
 Metric: 1
 Forward Address: 0.0.0.0
 External Route Tag: 1
```

Отключим теперь ADC #1 полностью и проверим маршрутизацию.

Маршрутизатор Internal router:

```
internal#sho ip ro os
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
 D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
 N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
 E1 - OSPF external type 1, E2 - OSPF external type 2
 i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
 ia - IS-IS inter area, * - candidate default, U - per-user static route
 o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
 + - replicated route, % - next hop override
Gateway of last resort is 192.168.1.3 to network 0.0.0.0
O*E2 0.0.0.0/0 [110/1] via 192.168.1.3, 00:01:08, GigabitEthernet0/0
```

Маршрутизатор External router:

```
external#sho ip ro os
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
 D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
 N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
 E1 - OSPF external type 1, E2 - OSPF external type 2
```

```
i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
ia - IS-IS inter area, * - candidate default, U - per-user static route
o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
+ - replicated route, % - next hop override
Gateway of last resort is not set
2.0.0.0/32 is subnetted, 1 subnets
0 E2 2.2.2.2 [110/10] via 192.168.0.3, 00:00:10, GigabitEthernet0/0
```

После возвращения в строй системы балансировки ADC #1 маршрутизация через это устройство восстанавливается.

## Закключение

Все описанное выше относится к режиму работы системы балансировки трафика с включенной опцией IP transparency. При опции IP Transparency Angie ADC сохраняет исходный адрес отправителя в IP-пакетах, позволяя серверам на сетевом уровне различать клиентские подключения. Очень часто администраторы систем балансировки отключают эту опцию, в этом случае Angie ADC будет выполнять не только dNAT, но sNAT над трафиком, приходящим от клиентов. Если система балансировки выполняет sNAT, то в этом случае маршрутизация несколько упрощается, так как в этом случае нет необходимости анонсировать в сторону Internal router маршрут по умолчанию. Если ADC #1 и ADC #2 выполняют sNAT в разные адреса, то в этом случае нет необходимости заботиться о том, с какими значениями метрики маршруты до соответствующих адресов попадают на маршрутизатор Internal router, так как анонсы будут уникальными.

## 5.2.2 Распределение нагрузки с помощью протокола OSPF (Active-Active)

### Введение

В этом разделе описана настройка высокой доступности с использованием протокола OSPF в режиме распределения нагрузки (Active-Active). Для настройки OSPF используется *интерфейс командной строки (CLI) Angie ADC*.

В качестве протокола динамической маршрутизации используется OSPFv2 для адресного семейства IPv4 unicast. Отказоустойчивость в рамках IPv6 unicast настраивается аналогично.

Для обеспечения распределения нагрузки между ADC нужны две системы балансировки Angie, работающие параллельно и одновременно. Пример типового участка сети с системами балансировки Angie ADC (ADC #1 и ADC #2):

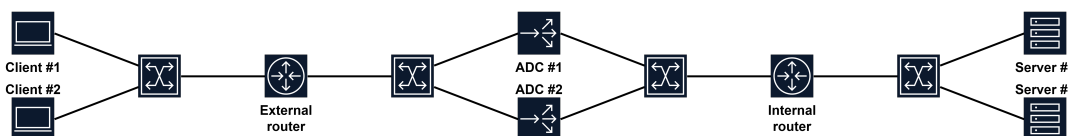


Рис. 1

Добавим IP-адресацию:

- В сегменте сети, куда подключаются External router, ADC #1 и ADC #2, будет использоваться подсеть 192.168.0.0/24.
- В сегменте сети, куда подключаются Internal router, ADC #1 и ADC #2, будет использоваться подсеть 192.168.1.0/24.
- Четвертый октет адреса для каждого устройства (в рамках соответствующей подсети) показан на схеме ниже.

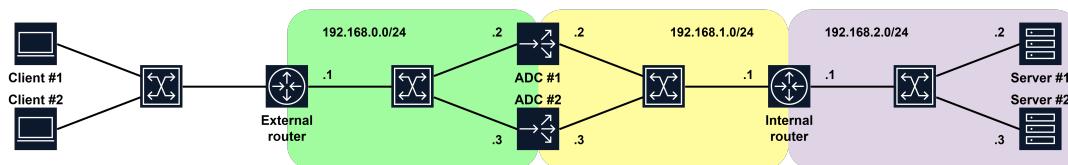


Рис. 2

Адресация клиентов не приводится, так как в данном случае мы будем считать, что их адреса формально не определены, то есть, они могут подключаться из любой сети, включая сеть интернет. Для этих целей External router будет анонсировать маршрут по умолчанию по OSPF для ADC #1 и ADC #2. Мы будем использовать OSPF-домен, включающий в себя четыре устройства: External router, Internal router, ADC #1 и ADC #2.

Красным текстом на схеме ниже отмечены IP-адреса виртуальных серверов. Это те адреса, к которым подключаются клиенты снаружи. Чтобы маршрутизатор External router мог выполнить распределение трафика между системами балансировки Angie ADC (см. ECMP – Equal-Cost Multi-Path), они должны одинаково анонсировать маршруты до виртуальных серверов в сторону External router (должны совпадать типы маршрутов и косты).

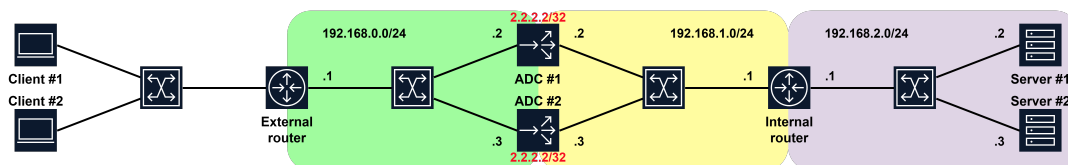


Рис. 3

Трафик внутри пользовательской сессии от серверов должен возвращаться строго через ту же систему балансировки, через которую он проходил до этого. Добиться этого можно с помощью механизма SNAT, когда адрес клиента подменяется на адрес интерфейса системы балансировки.

## Обработка типовых сбоев

Ниже описаны сценарии работы Angie ADC при сбоях.

### Полный отказ ADC #1

В этом случае оба OSPF-соседства с ADC #1 разрываются, никакие объявления маршрутов через ADC #1 не пролетают. Остается связь по OSPF только с ADC #2. То есть, переключение на резервную систему производится полностью автоматически, никакие дополнительные настройки не требуются.

Красными стрелками обозначены рабочие OSPF-соседства:

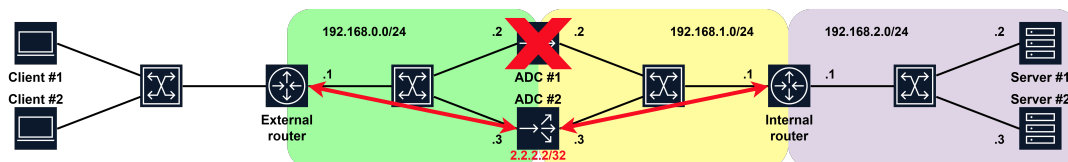


Рис. 4



## Отказ внешнего интерфейса ADC #1

Предлагаемая схема не предполагает возможности отслеживания частичных отказов и реагирования на них. Пример такого отказа представлен на схеме ниже.

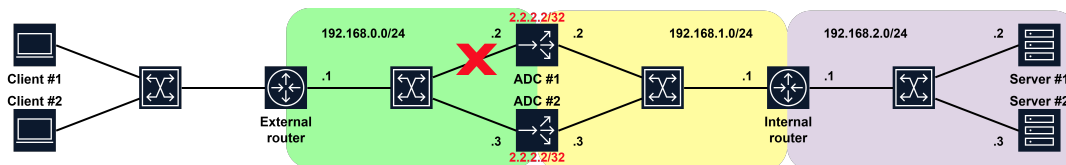


Рис. 5

В случае обозначенного на схеме отказа линка разрывается OSPF-сессия между ADC #1 и External router. ADC #1 теперь не может объявить маршрут до адреса виртуального сервера в сторону External router напрямую. Internal router получает маршруты до адресов, которые используются в SNAT от обеих систем балансировки, однако трафик от клиентов, прошедший через ADC #2, будет возвращаться тем же путем (так же через ADC #2), так как ADC #1 и ADC #2 используют уникальные адреса для SNAT.

## Отказ линка между ADC #1 и Internal router

Отказ линка между ADC #1 и Internal router более сложная ситуация: если OSPF-соседство между External router и ADC #1 сохраняется, то ADC #1 будет продолжать анонсировать маршрут до адреса виртуального сервера, хотя реальные сервера для системы балансировки ADC #1 уже недоступны.

Таким образом, мы получаем классический пример black hole для трафика:

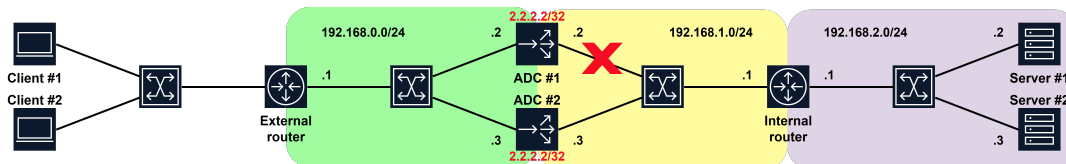


Рис. 6

Чтобы избежать этой проблемы, необходимо, чтобы ситуация отказа только одного из линков была невозможной, т.е. нужно обеспечить отказоустойчивое подключение к сети для хоста виртуализации, на котором работает ADC.

Вторым способом решить указанную проблему является *технология RHI*, позволяющая отозвать маршрут при недоступности апстрим-серверов.

Рекомендованная схема подключения:

- Объединить в одну LAG-группу все физические интерфейсы хоста виртуализации и использовать этот виртуальный LAG-интерфейс в качестве транка, то есть передавать по нему агрегированными несколько виртуальных сетей (один VLAN для внешней (external) сети, второй VLAN для внутренней (internal) сети).
- Для резервирования коммутаторов можно использовать такие технологии как vPC, VSS или стекирование, пример представлен на схеме ниже.

Из соображений простоты на схеме не отображено резервирование внешнего и внутреннего маршрутизаторов.

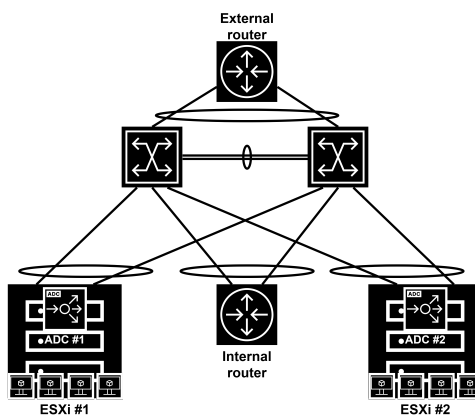


Рис. 7

#### Важно

Маршрутизатор, выполняющий ECMP-балансировку, и обе системы Angie ADC могут располагаться в одной OSPF-зоне или в разных. Если при этом маршрутизатор является OSPF ABR (Area Border Router), а ADC #1 и ADC #2 находятся в разных OSPF-зонах (видны через разные интерфейсы ABR), то ECMP-балансировка может не поддерживаться некоторыми вендорами сетевого оборудования.

Такой дизайн сети не рекомендуется:

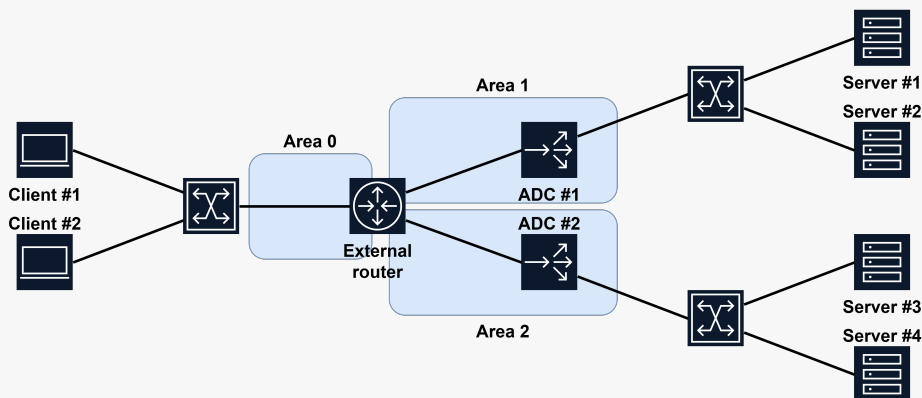


Рис. 8

## Настройки маршрутизации

Приведем ключевые настройки маршрутизации для External router, Internal router, ADC #1 и ADC #2.

### Ключевые настройки маршрутизации для External router

```
!Имя устройства
hostname external
!Интерфейс в сторону систем балансировки
interface GigabitEthernet0/0
ip address 192.168.0.1 255.255.255.0
router ospf 1
!Включение OSPF на интерфейсе в сторону систем балансировки
network 192.168.0.0 0.0.0.255 area 0
!Анонсирование маршрута по умолчанию
default-information originate always
end
```

### Ключевые настройки маршрутизации для Internal router

```
!Имя устройства
hostname internal
!Интерфейс в сторону систем балансировки
interface GigabitEthernet0/0
ip address 192.168.1.1 255.255.255.0
!Интерфейс в сторону подсети с реальными серверами
interface GigabitEthernet1/0
ip address 192.168.2.1 255.255.255.0
router ospf 1
!Добавление в OSPF информации о подсети с серверами
redistribute connected subnets
!Включение OSPF на интерфейсе в сторону систем балансировки
network 192.168.1.0 0.0.0.255 area 0
end
```

### Ключевые настройки маршрутизации системы балансировки ADC #1

```
!Имя устройства
hostname adc1
!Интерфейс в сторону маршрутизатора external router
interface ens33
description external
ip address 192.168.0.2/24
!Включение OSPF на интерфейсе
ip ospf 1 area 0
exit
!Интерфейс в сторону маршрутизатора internal router
interface ens37
description internal
ip address 192.168.1.2/24
!Включение OSPF на интерфейсе
ip ospf 1 area 0
exit
!Loopback-интерфейс для назначения адресов виртуальных серверов
interface lo
!Адрес тестового виртуального сервера
ip address 2.2.2.2/32
exit
```

```
!Префикс-лист, который используется для отбора префиксов для добавления в OSPF
ip prefix-list c2o seq 5 permit 2.2.2.2/32
!Route-map, с помощью которой производится отбор префиксов для добавления в OSPF
route-map c2o permit 10
match ip address prefix-list c2o
exit
!Настройки процесса маршрутизации OSPF
router ospf 1
!Необходимо вручную задать идентификатор маршрутизатора для OSPF
!Иначе автоматика может выбрать одинаковые идентификаторы для ADC#1 и ADC#2
ospf router-id 2.2.2.2
!Добавление информации о подключенных сетях в OSPF
redistribute connected route-map c2o
exit
end
```

### Ключевые настройки маршрутизации системы балансировки ADC #2

```
!Имя устройства
hostname adc2
!Интерфейс в сторону маршрутизатора external router
interface ens33
description external
ip address 192.168.0.3/24
!Включение OSPF на интерфейсе
ip ospf 1 area 0
exit
!Интерфейс в сторону маршрутизатора internal router
interface ens37
description internal
ip address 192.168.1.3/24
!Включение OSPF на интерфейсе
ip ospf 1 area 0
exit
!Loopback-интерфейс для назначения адресов виртуальных серверов
interface lo
!Адрес тестового виртуального сервера
ip address 2.2.2.2/32
exit
!Префикс-лист, который используется для отбора префиксов для добавления в OSPF
ip prefix-list c2o seq 5 permit 2.2.2.2/32
!Route-map, с помощью которой производится отбор префиксов для добавления в OSPF
route-map c2o permit 10
match ip address prefix-list c2o
exit
!Настройки процесса маршрутизации OSPF
router ospf 1
!Необходимо вручную задать идентификатор маршрутизатора для OSPF
!Иначе автоматика может выбрать одинаковые идентификаторы для ADC#1 и ADC#2
ospf router-id 22.22.22.22
!Добавление информации о подключенных сетях в OSPF
redistribute connected route-map c2o
exit
end
```

## Проверка работы

Выполним проверку работы системы маршрутизации, когда ADC #1 и ADC #2 находятся в полностью работоспособном состоянии.

### Примечание

В листингах ниже отсутствуют префиксы, не относящиеся к обсуждаемым вопросам отказоустойчивости.

Маршрутизатор External router видит маршрут к адресу виртуального сервера через обе системы балансировки.

```
external#sho ip ro os
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2
i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
ia - IS-IS inter area, * - candidate default, U - per-user static route
o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
+ - replicated route, % - next hop override
Gateway of last resort is not set
2.0.0.0/32 is subnetted, 1 subnets
O E2 2.2.2.2 [110/20] via 192.168.0.2, 00:41:28, GigabitEthernet0/0
via 192.168.0.3, 00:41:15, GigabitEthernet0/0
external#sho ip os da ex 2.2.2.2
OSPF Router with ID (192.168.0.1) (Process ID 1)
Type-5 AS External Link States
Routing Bit Set on this LSA in topology Base with MTID 0

LS age: 898
Options: (No TOS-capability, No DC, Upward)
LS Type: AS External Link
Link State ID: 2.2.2.2 (External Network Number)
Advertising Router: 2.2.2.2
LS Seq Number: 80000003
Checksum: 0xAA0C
Length: 36
Network Mask: /32
Metric Type: 2 (Larger than any link state path)
MTID: 0
Metric: 20
Forward Address: 0.0.0.0
External Route Tag: 0

LS age: 885
Options: (No TOS-capability, No DC, Upward)
LS Type: AS External Link
Link State ID: 2.2.2.2 (External Network Number)
Advertising Router: 22.22.22.22
LS Seq Number: 80000003
Checksum: 0xB4A7
Length: 36
Network Mask: /32
Metric Type: 2 (Larger than any link state path)
MTID: 0
```

```
Metric: 20
Forward Address: 0.0.0.0
External Route Tag: 0
```

По умолчанию Angie ADC будет устанавливать соединение с апстрим-сервером с адреса того интерфейса, через который этот сервер достижим, в соответствии с таблицей маршрутизации (192.168.1.2 для ADC #1 и 192.168.1.3 для ADC #2). Оба адреса принадлежат подсети 192.168.1.0/24, которая для маршрутизатора Internal router является непосредственно подключенной (Directly Connected). Анонсировать другие адреса не нужно. Если используется SNAT pool, то каждая из систем балансировки объявит в сеть дополнительные адреса, соответствующие пулу.

Отключим теперь ADC #1 полностью и проверим маршрутизацию.

Маршрутизатор External router:

```
external#sho ip ro os
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2
i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
ia - IS-IS inter area, * - candidate default, U - per-user static route
o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
+ - replicated route, % - next hop override
Gateway of last resort is not set
2.0.0.0/32 is subnetted, 1 subnets
O E2 2.2.2.2 [110/20] via 192.168.0.3, 01:40:10, GigabitEthernet0/0
```

После возвращения в строй ADC #1 маршрутизация через это устройство восстанавливается.

### Применение с IP transparency

Все описанное выше относится к режиму работы системы балансировки трафика с отключенной опцией IP transparency. При включении опции IP Transparency Angie ADC сохраняет исходный адрес отправителя в IP-пакетах, позволяя серверам на сетевом уровне различать клиентские подключения. Однако в этом случае маршрутизатор Internal router будет возвращать трафик через неправильную систему балансировки. Исходный адрес клиента может быть передан серверу либо с помощью заголовка X-FORWARDED-FOR, либо с помощью специального проху protocol.

## 5.3 VRRP-маршрутизация

VRRP (Virtual Router Redundancy Protocol) — это протокол для виртуального резервирования устройств. VRRP позволяет переключиться на резервный Angie ADC, который автоматически принимает на себя IP-адрес и роль мастера, если основное устройство Angie ADC отказало. Таким образом обеспечивается отказоустойчивость для подключенных узлов в сети при статической маршрутизации. Поддерживаемые версии: VRRPv2 (для IPv4) и VRRPv3 (для IPv4 и IPv6).

#### Примечание

Настройка VRRP осуществляется через *командную строку (CLI)* на порту 2022.

В этом разделе:

- *Режим резервирования (Active-Standby)*
- *Режим распределения нагрузки (Active-Active)*

- Команды VRRP
- Создание и удаление VRRP-групп

### 5.3.1 Предварительная настройка для VMware ESXi

Если Angie ADC развернут на VMware ESXi, то для работы VRRP необходимо дополнительно настроить сетевые политики безопасности ESXi:

1. Разрешите передачу трафика с портов, участвующих в VRRP. Для этого:
  - На VMware ESXi до версии 6.7 создайте отдельную группу (Port Group) для портов, участвующих в VRRP, и включите для этой группы **Promiscuous mode**.
  - На VMware ESXi версии 6.7+ включите **MAC learning** (MAC-обучение) для группы портов, участвующих в VRRP (подходит для распределенных коммутаторов vDS).
2. Разрешите **MAC Address Changes** и **Forged Transmits** для передачи трафика при изменении MAC-адреса.

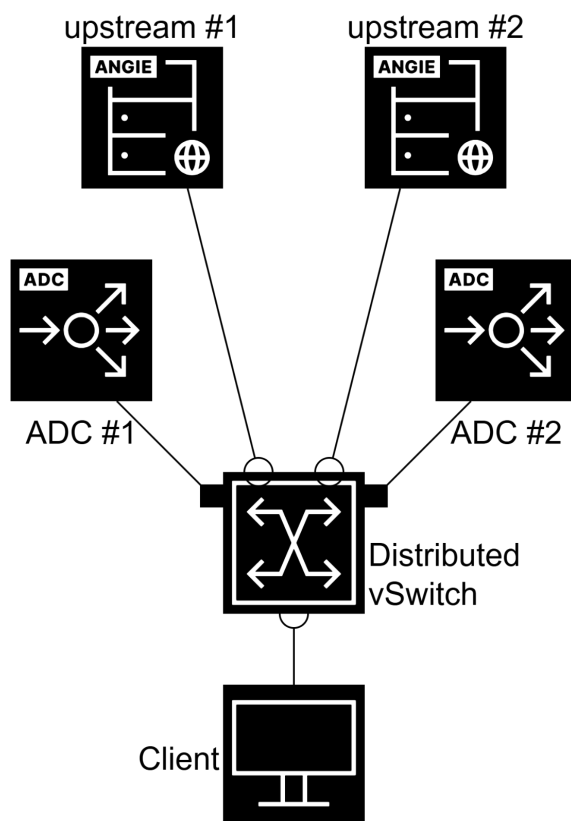


Рис. 2: Рис. 1: ESXi до версии 6.7 — устройства Angie ADC подключены через отдельные порт-группы, на которых включен режим Promiscuous.

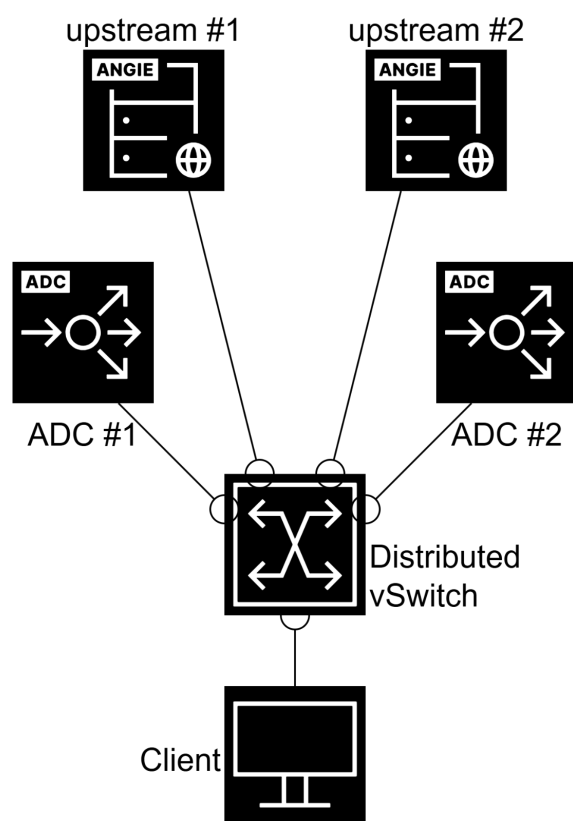


Рис. 3: Рис. 2: ESXi версии 6.7+ — устройства Angie ADC подключены через обычные порт-группы, при этом включена политика, разрешающая подмену MAC-адресов в Distributed Virtual Port Group (DVPG.macManagementPolicy=yes).



## Режим резервирования (Active-Standby)

В этом разделе описана настройка высокой доступности с использованием протокола VRRP в режиме резервирования (Active-Standby). Для настройки используется *интерфейс командной строки (CLI) Angie ADC*.

Также можно посмотреть следующие статьи:

- *Настройка режима распределения нагрузки (Active-Active) с помощью протокола VRRP;*
- *Особенности создания и удаления VRRP-групп;*
- *Команды VRRP.*

## Введение

Для обеспечения отказоустойчивости нужны две системы балансировки Angie ADC — основная и резервная. Резервная система Angie ADC будет принимать нагрузку при отказе основной. Автоматическое переключение между двумя системами балансировки Angie ADC можно реализовать с помощью протокола VRRP. Для настройки VRRP используется *командная строка Angie ADC CLI*. Пример настройки приведен ниже.

### Примечание

Резервирование других сетевых компонентов (маршрутизаторы, коммутаторы, межсетевые экраны) выходит за рамки настоящей статьи.

Пример типового участка сети с системами балансировки Angie ADC (ADC #1 и ADC #2):

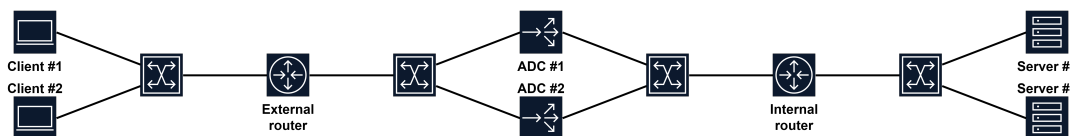


Рис. 1

Добавим IP-адресацию:

- В сегменте сети, куда подключаются External router, ADC #1 и ADC #2, будет использоваться подсеть 192.168.0.0/24.
- В сегменте сети, куда подключаются Internal router, ADC #1 и ADC #2, будет использоваться подсеть 192.168.1.0/24.
- Четвертый октет адреса для каждого устройства (в рамках соответствующей подсети) показан на схеме ниже.

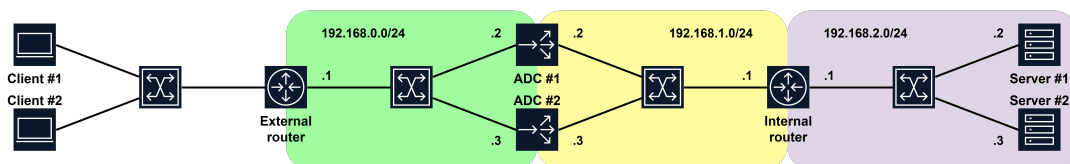


Рис. 2

Адресация клиентов не приводится, так как в данном случае мы будем считать, что их адреса формально не определены, то есть, они могут подключаться из любой сети, включая сеть интернет.

Для этого на ADC #1 и ADC #2 необходимо указать адрес правого интерфейса маршрутизатора External router (192.168.0.1) в качестве шлюза по умолчанию.

Виртуальный адрес, который соответствует сервису, может быть размещен на loopback-интерфейсе каждой системы балансировки, либо это может быть виртуальный адрес, используемый протоколом VRRP. Первый способ будет более гибким.

При размещении адреса сервиса на loopback-интерфейсе (на схеме ниже это адрес 2.2.2.2) необходимо добавить статический маршрут на этот адрес на маршрутизаторе External router. Причем в качестве next-hop необходимо использовать виртуальный адрес VRRP, за счет чего достигается отказоустойчивость решения.

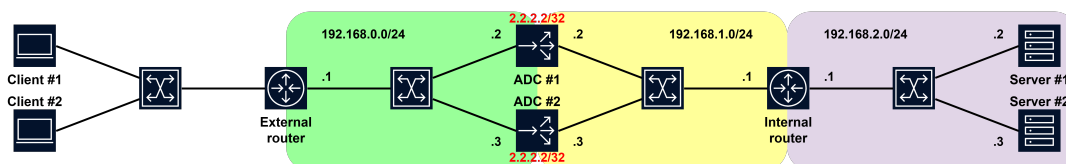


Рис. 3

При использовании виртуального адреса VRRP в качестве адреса сервиса (на схеме ниже это 192.168.0.10) настройка соответствующих статических маршрутов на маршрутизаторе External router не требуется. Виртуальный адрес VRRP находится в одной подсети с интерфейсами маршрутизатора с точки зрения маршрутизатора External router.

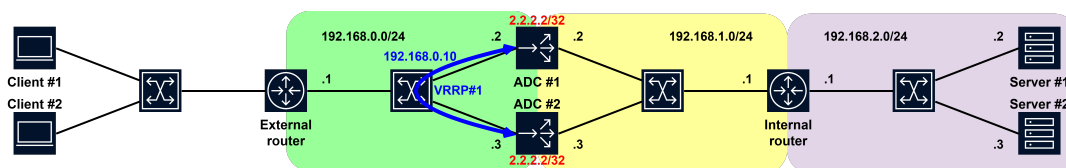


Рис. 4

Также допускается использование VRRP и при работе системы балансировки Angie ADC в «одно-руком» режиме.

## Обработка типовых сбоев

Ниже описаны сценарии работы Angie ADC при сбоях.

### Полный отказ ADC #1

При полном отказе или отключении от сети ADC #1 происходят пере выборы VRRP-мастера. Переключение на резервную систему ADC #2 производится автоматически. Никакие дополнительные настройки не требуются.

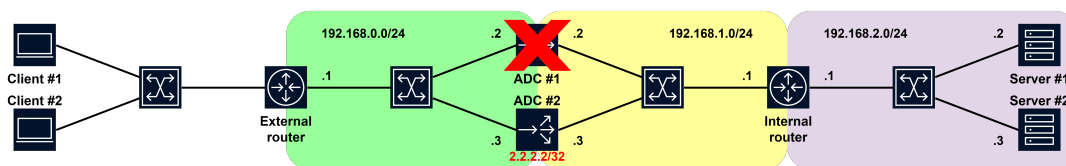


Рис. 5

## Отказ внешнего интерфейса ADC #1

Предлагаемая схема не предполагает возможности отслеживания частичных отказов и реагирования на них (пример ниже).

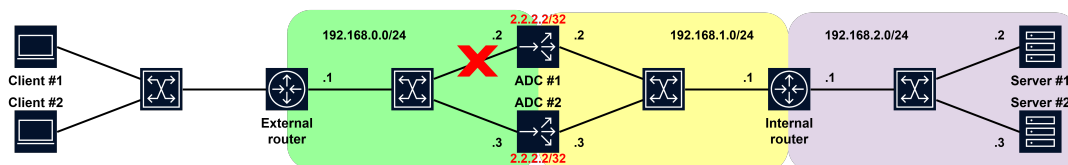


Рис. 6

В случае частичного отказа, как на схеме выше, ADC #1 перестает рассылать сообщения VRRP. ADC #2 начинает считать себя мастером, что заставляет виртуальную машину отвечать на ARP-запросы об адресе 192.168.0.10, а также обрабатывать пользовательский трафик. Обратный трафик (от серверов в сторону клиентов) будет проходить через ADC #2, так как ADC #2 устанавливает сессии до серверов с адреса своего внутреннего интерфейса.

### Примечание

IP transparency на данном этапе обеспечить невозможно. Для работы IP transparency требуется наличие маршрута по умолчанию на маршрутизаторе Internal router. Если прописать маршрут по умолчанию через ADC #1, то отказ внешнего интерфейса ADC #1 приведет к невозможности доставки трафика и его отбрасыванию (black hole). Решить эту проблему можно за счет использования протокола VRRP как на внешнем, так и на внутреннем интерфейсах. Причем состояние групп должно быть согласованным, то есть если для внешней сети мастером является система балансировки ADC #2, то и для внутренней сети ADC #2 также должна являться мастером. Добиться этого можно с помощью опции VRRP tracking, которая будет выпущена в ближайших релизах Angie ADC.

## Отказ линка между ADC #1 и Internal router

Отказ линка между ADC #1 и маршрутизатором Internal router более сложная ситуация: если ADC #1 сохраняет за собой роль мастера, то трафик от клиентов до серверов будет все еще передаваться через ADC #1, хотя реальные сервера для системы балансировки ADC #1 уже недостижимы. Таким образом, мы получаем классический пример black hole:

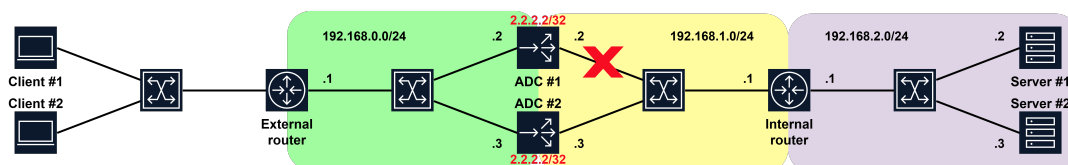


Рис. 7

Чтобы избежать этой проблемы, необходимо, чтобы ситуация отказа только одного из линков была невозможной, т.е. нужно обеспечить отказоустойчивое подключение к сети для хоста виртуализации, на котором работает ADC.

Рекомендованная схема подключения:

- Объединить в одну LAG-группу все физические интерфейсы хоста виртуализации и использовать этот виртуальный LAG-интерфейс в качестве транка, то есть передавать по нему те

гированными несколько виртуальных сетей (один VLAN для external сети, второй VLAN для internal сети).

- Для резервирования коммутаторов можно использовать технологию vPC или стекирование, пример представлен на схеме ниже.

Из соображений простоты на схеме не отображено резервирование внешнего и внутреннего маршрутизаторов.

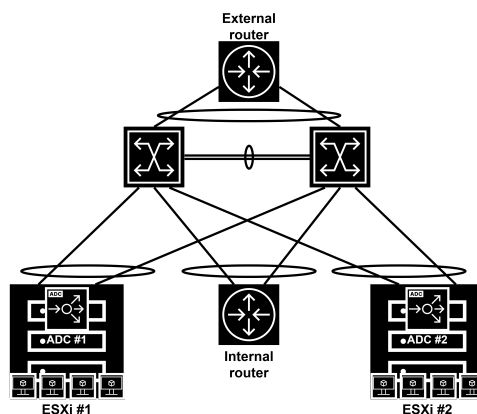


Рис. 8

#### Примечание

В ближайших релизах Angie ADC появится опция VRRP tracking, которая позволит автоматически переводить систему балансировки ADC #2 в режим VRRP backup при возникновении обсуждаемого отказа.

## Настройка VRRP

### Базовая настройка

Для настройки VRRP необходимо указать виртуальный IP-адрес, все остальные настройки опциональны:

```
#! Задать виртуальный IP-адрес 192.168.0.10/24 для первой VRRP группы внешнего
↪интерфейсе (ens33).
#! Задаётся для ADC #1 и для ADC #2.
vrrp 1 int ens33 ip 192.168.0.10/24
```

Указанная выше настройка одинакова для ADC #1 и ADC #2.

#### Примечание

Полный список поддерживаемых команд VRRP можно посмотреть в *справочнике по командам VRRP*.

## Изменение приоритета

Если требуется, чтобы система балансировки ADC #1 выигрывала выборы и принимала на себя роль мастера, то ее приоритет необходимо сделать выше, чем приоритет у ADC #2. Значение приоритета по умолчанию – 100. Пример настройки ADC #1 представлен ниже:

```
#! Задается для ADC #1
#vrrp 1 int ens33 priority 110
```

Аналогичная настройка делается для ADC #2, но с меньшим значением приоритета:

```
#! Задается для ADC #2
#vrrp 1 int ens33 priority 90
```

## Использование preempt

Опция VRRP `preempt` включена по умолчанию. Эта опция заставляет включившуюся систему балансировки инициировать перевыборы мастера, если ее собственный приоритет выше приоритета текущего мастера.

Включить `preempt` можно с помощью следующей команды:

```
vrrp 1 int ens33 preempt
```

Выключение `preempt` реализовано через `no`:

```
no vrrp 1 int ens33 preempt
```

## Прописывание маршрута

Для указания статического маршрута, например на внешнем маршрутизаторе с указанием виртуального адреса VRRP в качестве next-hop, используется стандартная команда:

```
External(config)#ip route 2.2.2.2 255.255.255 192.168.0.10
```

## Проверка работы

Убедимся, что ADC #1 является мастером:

```
show vrrp int ens33 1 sum
Interface VRID Priority IPv4 IPv6 State (v4) State (v6)

ens33 1 110 1 0 Master Backup
```

Убедимся, что ADC #2 не обладает ролью мастера:

```
show vrrp int ens33 1 sum
Interface VRID Priority IPv4 IPv6 State (v4) State (v6)

ens33 1 90 1 0 Backup Backup
```

Отключим теперь систему балансировки ADC #1 и убедимся, что ADC #2 приняла на себя роль мастера:

```
show vrrp int ens33 1 sum
Interface VRID Priority IPv4 IPv6 State (v4) State (v6)

ens33 1 90 1 0 Master Backup
```

#### Примечание

В момент переезда роли Master на вторую систему балансировки ADC возможны кратковременные перерывы обработки трафика, вызванные задержками обнаружения падения активной ноды. Характерное время переключения ~3 секунды. Ускорить процедуру переключения можно за счет редактирования параметра `advertisement-interval` (см. [справочник по командам VRRP](#)). В будущем планируется использование протокола BFD для ускорения процедуры обнаружения отказов.

Заново включим систему балансировки ADC #1. Поведение устройства зависит от состояния опции VRRP `preempt`. Если опция `preempt` активирована (значение по умолчанию), то ADC #1 инициирует перевыборы и заберет на себя роль мастера. Если опция `preempt` отключена, то ADC #1 не будет пытаться отобрать роль мастера у системы балансировки ADC #2.

Проверить текущее состояние опции `preempt` можно с помощью следующей команды:

```
show vrrp int ens33 1
Virtual Router ID 1
Protocol Version 3
Autoconfigured No
Shutdown No
Interface ens33
VRRP interface (v4) vrrp4-ens33-1
VRRP interface (v6) None
Primary IP (v4) 192.168.0.3
Primary IP (v6) ::
Virtual MAC (v4) 00:00:5e:00:01:01
Virtual MAC (v6) 00:00:5e:00:02:01
Status (v4) Master
Status (v6) Initialize
Priority 110
Effective Priority (v4) 110
Effective Priority (v6) 110
Preempt Mode No
Accept Mode Yes
Checksum with IPv4 Pseudoheader Yes
Advertisement Interval 1000 ms
Master Advertisement Interval (v4) Rx 1000 ms (stale)
Master Advertisement Interval (v6) Rx 0 ms (stale)
Advertisements Tx (v4) 1163
Advertisements Tx (v6) 0
Advertisements Rx (v4) 0
Advertisements Rx (v6) 0
Gratuitous ARP Tx (v4) 1
Neigh. Adverts Tx (v6) 0
State transitions (v4) 2
State transitions (v6) 0
Skew Time (v4) 600 ms
Skew Time (v6) 0 ms
Master Down Interval (v4) 3600 ms
Master Down Interval (v6) 0 ms
IPv4 Addresses 1
```

```
..... 192.168.0.10
IPv6 Addresses 0
```

Вне зависимости от того, какая система балансировки активна в данный момент, на маршрутизаторах в ARP-записи об адресе 192.168.0.10 будет один и тот же MAC-адрес: 00-00-5e-00-01-01 (при использовании первой VRRP-группы).

```
External#sho ip arp
Protocol Address Age (min) Hardware Addr Type Interface
Internet 192.168.0.10 - 0000.5e00.0101 ARPA GigabitEthernet0/0
```

Это означает, что при отказе одной из систем балансировки и аварийном переключении на вторую, единственное изменение, которое должно произойти в сети – переобучение коммутаторов, то есть потребуется актуализировать таблицы коммутации для адреса 00-00-5e-00-01-01.

## Заключение

С помощью протокола VRRP можно построить отказоустойчивое решение по балансировке нагрузки на сервера там, где использование протоколов динамической маршрутизации невозможно по той или иной причине.

Ряд дополнительных опций, расширяющих возможности протокола VRRP, находятся в данный момент в стадии разработки и будут доступны в ближайших релизах Angie ADC.

## Режим распределения нагрузки (Active-Active)

В этом разделе описана настройка высокой доступности с использованием протокола VRRP в режиме распределения нагрузки (Active-Active). Для настройки используется *интерфейс командной строки (CLI) Angie ADC*.

Также можно посмотреть следующие статьи:

- *Настройка режима резервирования (Active-Standby) с помощью протокола VRRP;*
- *Особенности создания и удаления VRRP-групп;*
- *Команды VRRP.*

## Введение

Реализация полноценной схемы распределения нагрузки (Active-Active) не предусмотрена протоколом VRRP, но это ограничение можно обойти. Для этого рассмотрим два сценария подключения Angie ADC:

- между клиентами и системами балансировки установлен хотя бы один маршрутизатор;
- между клиентами и системами балансировки нет маршрутизатора.

## Подключение Angie ADC с маршрутизатором

Пример сети с маршрутизатором между клиентами и системами балансировки Angie ADC (ADC #1 и ADC #2):

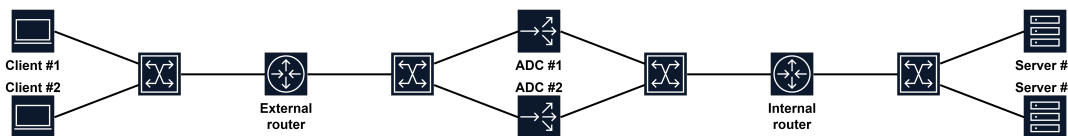


Рис. 1

Добавим IP-адресацию:

- В сегменте сети, куда подключаются External router, ADC #1 и ADC #2, будет использоваться подсеть 192.168.0.0/24.
- В сегменте сети, куда подключаются Internal router, ADC #1 и ADC #2, будет использоваться подсеть 192.168.1.0/24.
- Четвертый октет адреса для каждого устройства (в рамках соответствующей подсети) показан на схеме ниже.

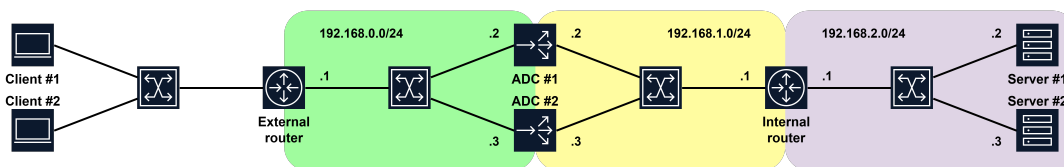


Рис. 2

Адресация клиентов не приводится, так как в данном случае мы будем считать, что их адреса формально не определены, то есть, они могут подключаться из любой сети, включая сеть интернет. Для этого на ADC #1 и ADC #2 необходимо указать адрес правого интерфейса маршрутизатора External router (192.168.0.1) в качестве шлюза по умолчанию.

Виртуальный адрес, который соответствует сервису, может быть размещен на loopback-интерфейсе каждой системы балансировки, либо это может быть виртуальный адрес, используемый протоколом VRRP.

- Первый способ (рис. 3 – сервис размещен на loopback-интерфейсах и IP-адреса этих интерфейсов совпадают) будет более гибким, так как позволяет использовать только один IP-адрес для сервиса.
- Во втором случае (используется виртуальный адрес протокола VRRP) необходимо сообщить клиенту, что сервис работает на двух адресах (например, с помощью DNS).
- В третьем случае, если сервис размещен на loopback-интерфейсах, но IP-адреса этих интерфейсов не совпадают, также потребуется сообщить клиенту, что сервис работает на двух адресах.

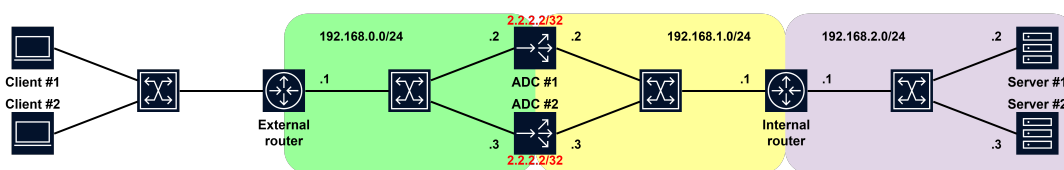


Рис. 3



При размещении адреса сервиса на loopback-интерфейсе (на схеме ниже это адрес 2.2.2.2) необходимо добавить статический маршрут на этот адрес на маршрутизаторе External router. Причем в качестве next-hop необходимо использовать виртуальный адрес VRRP – 192.168.0.10, за счет чего достигается отказоустойчивость решения.

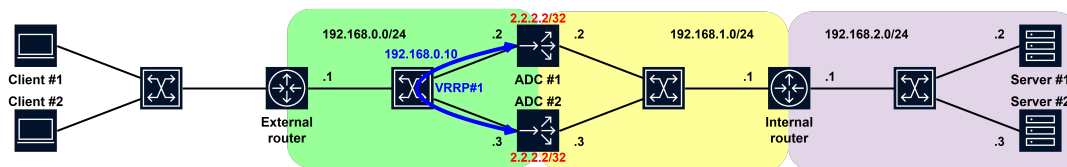


Рис. 4

### Настройка режима Active-Active

Чтобы перейти от режима работы Active-Standby к режиму Active-Active, необходимо на том же интерфейсе систем балансировки настроить вторую VRRP группу таким образом, чтобы мастером в ней являлась другая система Angie ADC (достигается за счет опции **priority**). Для второй группы должен быть настроен виртуальный IP-адрес, отличающийся от адреса первой группы.

На схеме ниже вторая группа имеет виртуальный IP-адрес 192.168.0.11:

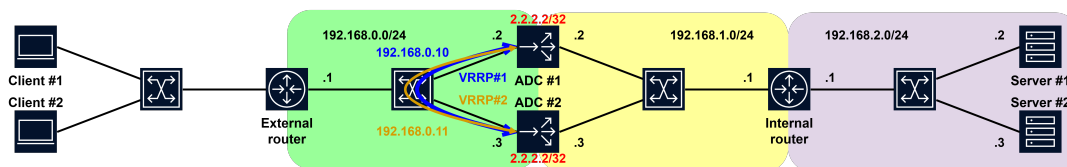


Рис. 5

На стороне маршрутизатора External router должно быть прописано два статических маршрута до префикса 2.2.2.2/32:

- один с next-hop 192.168.0.10;
- второй – 192.168.0.11.

Маршрутизатор External router должен выполнять ECMP-балансировку трафика (часть сессий отправляется в сторону ADC #1, а оставшиеся – в сторону ADC #2).

Ниже приведен пример настройки маршрутизатора Cisco:

```
External(config)#ip route 2.2.2.2 255.255.255.255 192.168.0.10
External(config)#ip route 2.2.2.2 255.255.255.255 192.168.0.11
External(config)#exi
External#sho ip ro 2.2.2.2
Routing entry for 2.2.2.2/32
Known via "static", distance 1, metric 0
Routing Descriptor Blocks:
192.168.0.11
Route metric is 0, traffic share count is 1
* 192.168.0.10
Route metric is 0, traffic share count is 1
External#
```

У VRRP-группы есть виртуальный MAC-адрес (00:00:5e:00:01:n, где n – номер группы VRRP, номер виртуального маршрутизатора, в шестнадцатеричном виде). По MAC-адресу коммутатор определяет, для какой из систем балансировки предназначен трафик. Первая группа будет иметь виртуальный MAC-адрес 00:00:5e:00:01:01, вторая – 00:00:5e:00:01:02.

Убедиться в этом можно, просмотрев ARP-таблицу на маршрутизаторе External router:

```
External#sho ip arp
Protocol Address Age (min) Hardware Addr Type Interface
Internet 192.168.0.1 - ca01.0950.0008 ARPA GigabitEthernet0/0
Internet 192.168.0.10 0 0000.5e00.0101 ARPA GigabitEthernet0/0
Internet 192.168.0.11 0 0000.5e00.0102 ARPA GigabitEthernet0/0
External#
```

ARP-таблица маршрутизатора External router не меняется при отказе одной из систем балансировки нагрузки Angie ADC. Изменения будут на стороне коммутатора. Ниже приводится часть таблицы коммутации, когда обе системы балансировки «живы» и выполняют свои функции:

```
switch#sho mac address-table dynamic vlan 51
Mac Address Table

Vlan Mac Address Type Ports

51 0000.5e00.0101 DYNAMIC Gi0/7
51 0000.5e00.0102 DYNAMIC Gi0/8
```

Система балансировки ADC #1 подключена к интерфейсу Gi0/7 коммутатора, а система балансировки ADC #2 – к интерфейсу Gi0/8.

При отказе одной из систем балансировки оставшаяся будет иметь обе VRRP-группы в статусе Master. С точки зрения коммутатора это будет выглядеть так, словно оба виртуальных MAC-адреса доступны через один интерфейс.

```
switch#sho mac address-table dynamic vlan 51
Mac Address Table

Vlan Mac Address Type Ports

51 0000.5e00.0101 DYNAMIC Gi0/7
51 0000.5e00.0102 DYNAMIC Gi0/7
```

## Подключение Angie ADC без маршрутизатора

Пример схемы сети представлен на рисунке ниже.

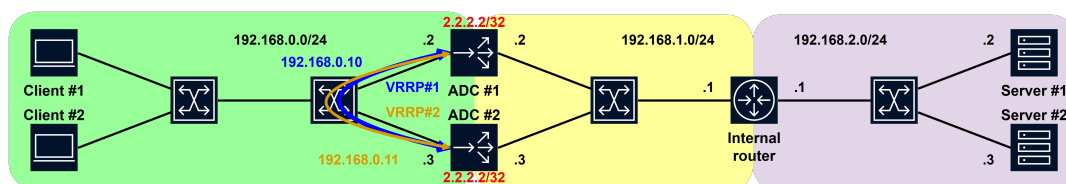


Рис. 6

В данном случае клиенты должны находиться в той же подсети, что и внешние интерфейсы систем балансировки Angie ADC (192.168.0.0/24).

Если клиенты Client #1 и Client #2 не имеют выхода в интернет, то тогда на них будет достаточно настроить один из адресов 192.168.0.10 и 192.168.0.11 в качестве шлюза по умолчанию. Тогда половина клиентов будет использовать адрес 192.168.0.10 в качестве шлюза по умолчанию, а вторая половина – 192.168.0.11.

Если для выхода в интернет используется какое-то отдельное устройство, то на клиентах придется прописать статический маршрут до префикса 2.2.2.2/32. На половине клиентов в качестве адреса

следующего перехода (next-hop) необходимо настроить виртуальный адрес первой группы – 192.168.0.10, а на оставшихся клиентах – адрес второй группы, 192.168.0.11.

На схеме ниже желтой и красной стрелками указаны направления на шлюзы по умолчанию.

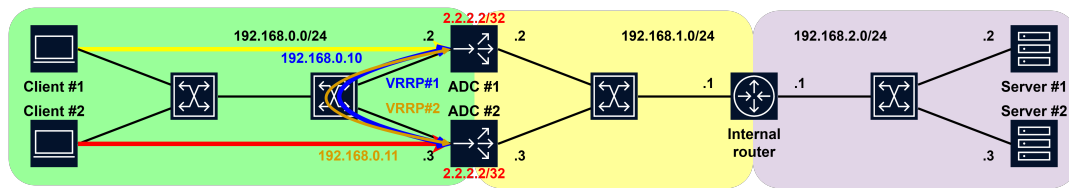


Рис. 7

Использовать ECMP-подход здесь нецелесообразно, так как некоторые клиентские операционные системы могут не поддерживать такую балансировку.

## Настройка VRRP

Настройка VRRP на системах балансировки не зависит от того, какая схема подключения используется, с маршрутизатором или без.

Для настройки VRRP необходимо:

- на каждой системе балансировки создать две группы;
- для каждой из групп указать виртуальный IP-адрес и приоритет.

Все остальные настройки опциональны.

### Примечание

Полный список поддерживаемых команд VRRP можно посмотреть в [справочнике по командам VRRP](#).

Пример настройки VRRP для ADC #1:

```
#! Задать виртуальный IP-адрес 192.168.0.10/24 для первой VRRP группы внешнего
↪интерфейсе (ens33).
vrrp 1 int ens33 ip 192.168.0.10/24
#! Задать приоритет 110 для первой VRRP группы внешнего интерфейсе (ens33).
vrrp 1 int ens33 priority 110
#! Задать виртуальный IP-адрес 192.168.0.11/24 для второй VRRP группы внешнего
↪интерфейсе (ens33).
vrrp 2 int ens33 ip 192.168.0.11/24
#! Задать приоритет 90 для второй VRRP группы внешнего интерфейсе (ens33).
vrrp 2 int ens33 priority 90
```

Настройка VRRP для ADC #2 симметричная:

```
#! Задать виртуальный IP-адрес 192.168.0.10/24 для первой VRRP группы внешнего
↪интерфейсе (ens33).
vrrp 1 int ens33 ip 192.168.0.10/24
#! Задать приоритет 90 для первой VRRP группы внешнего интерфейсе (ens33).
vrrp 1 int ens33 priority 90
#! Задать виртуальный IP-адрес 192.168.0.11/24 для второй VRRP группы внешнего
↪интерфейсе (ens33).
vrrp 2 int ens33 ip 192.168.0.11/24
```

```
#! Задать приоритет 110 для второй VRRP группы внешнего интерфейсе (ens33).
vrrp 2 int ens33 priority 110
```

## Проверка работы

Убедимся, что в стабильном и полностью рабочем состоянии ADC #1 является мастером для первой группы VRRP, а для второй группы мастером является система ADC #2.

Вывод с системы балансировки ADC #1:

```
show vrrp int ens33 sum
Interface VRID Priority IPv4 IPv6 State (v4) State (v6)

ens33 1 110 1 0 Master Backup
ens33 2 90 1 0 Backup Backup
```

Вывод с системы балансировки ADC #2:

```
show vrrp int ens33 sum
Interface VRID Priority IPv4 IPv6 State (v4) State (v6)

ens33 1 90 1 0 Backup Backup
ens33 2 110 1 0 Master Backup
```

При отказе, например, системы балансировки ADC #2, вывод информации о группах VRRP с системы балансировки ADC #1 будет выглядеть так:

```
show vrrp int ens33 sum
Interface VRID Priority IPv4 IPv6 State (v4) State (v6)

ens33 1 110 1 0 Master Backup
ens33 2 90 1 0 Master Backup
##
```

## Заключение

С помощью протокола VRRP можно построить отказоустойчивое решение по балансировке нагрузки на сервера там, где использование протоколов динамической маршрутизации невозможно по той или иной причине. Стоит отдельно отметить, что поддерживается работа систем балансировки как в режиме Active-Standby, так и в режиме Active-Active.

Ряд дополнительных опций, расширяющих возможности протокола VRRP, находятся сейчас в разработке и будут доступны в ближайших релизах Angie ADC.

## Команды VRRP

Здесь описываются команды VRRP, поддерживаемые в Angie ADC, и особенности их работы. Настройка VRRP осуществляется через *командную строку*.

Также вы можете посмотреть следующие статьи:

- *Настройка режима резервирования (Active-Standby) с помощью протокола VRRP;*
- *Настройка режима распределения нагрузки (Active-Active) с помощью протокола VRRP;*
- *Особенности создания и удаления VRRP-групп.*

Обобщенный пример команды VRRP:

```
vrrp n int ens33 command
```

Параметры:

- **n** - номер VRRP-группы (так же VRID — Virtual Router ID). Поддерживаются номера от 1 до 255.
- **ens33** - имя физического интерфейса. Если требуется создать VRRP-группы для другого физического интерфейса, то в команде необходимо указать имя этого интерфейса. VRRP можно независимо использовать для нескольких интерфейсов одновременно.
- **command** - команда VRRP (список поддерживаемых команд см. ниже).

## Команды настройки

| Команда                                                            | Описание                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | Значение по умолчанию                                     |
|--------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------|
| <code>vrrp n int &lt;interface&gt; ip &lt;ip-address&gt;</code>    | Команда добавляет IPv4-адрес для Angie ADC. Добавление IP-адреса автоматически активирует его. Задается в формате <code>&lt;ip_address&gt;/&lt;subnet_mask&gt;</code> .<br><br><b>Примечание</b><br>Маска подсети IP-адреса должна соответствовать маске физического интерфейса.                                                                                                                                                                                                                                          | n/a                                                       |
| <code>vrrp n int &lt;interface&gt; version v</code>                | Команда задает версию протокола VRRP (v равно 2 или 3). Если версия протокола не задана, по умолчанию используется VRRPv3.<br><br><b>Примечание</b><br>Версии протокола VRRP на двух Angie ADC, работающих вместе, должны совпадать, иначе оба Angie ADC VA будут считать себя мастером.                                                                                                                                                                                                                                  | 3                                                         |
| <code>vrrp n int &lt;interface&gt; advertisement-interval i</code> | Команда устанавливает значение интервала отправки анонсов для VRRP-группы n. Значение задается в миллисекундах, но должно быть кратно 10, так как VRRP использует единицы измерения в сотых долях секунды (i от 10 до 40950 в мс).                                                                                                                                                                                                                                                                                        | 1000 мс (1 с)                                             |
| <code>vrrp n int &lt;interface&gt; preempt</code>                  | Команда включает режим <b>preempt</b> , разрешающий запуск выборов нового мастера. При включенном режиме <b>preempt</b> резервный Angie ADC может автоматически стать мастером, если его приоритет выше, чем у текущего Angie ADC с ролью <b>master</b> .                                                                                                                                                                                                                                                                 | Enabled (режим preempt включен)                           |
| <code>vrrp n int &lt;interface&gt; priority p</code>               | Команда устанавливает приоритет p для устройств в VRRP-группе n на интерфейсе <interface> (p от 1 до 254). Значение приоритета 255 зарезервировано для текущего устройства в роли <b>master</b> , чей адрес интерфейса совпадает с виртуальным адресом группы. Значение 255 нельзя задать вручную. Устройство с наивысшим приоритетом становится <b>master</b> . Если у нескольких устройств в VRRP-группе одинаковый приоритет и режим preempt включен, мастером становится устройство с наибольшим основным IP-адресом. | 100                                                       |
| <code>vrrp n int &lt;interface&gt; shutdown</code>                 | Команда выключает VRRP-группу n на интерфейсе <interface>. Используйте эту команду, если нужно завершить работу VRRP-группы без ее удаления. Чтобы обратно включить VRRP для этой группы, необходимо отменить текущую команду с помощью <b>no</b> (см. следующую таблицу).                                                                                                                                                                                                                                                | Disabled (режим shutdown не активирован, группа включена) |
| <code>vrrp n int &lt;interface&gt; checksum-with-ipv4</code>       | Команда включает использование псевдо-заголовка IPv4 при расчете контрольной суммы VRRPv3 для VRRP-группы n. Эта команда не влияет на VRRPv2 и IPv6.                                                                                                                                                                                                                                                                                                                                                                      | Enabled (использование псевдо-заголовков IPv4 включено)   |

## Команды удаления и отмены

| Команда                                                                                                                                                   | Описание                                                                                                                                                               |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>no vrrp n int<br/>&lt;interface&gt; ip<br/>&lt;ip-address&gt;</code>                                                                                | Команда удаляет IP-адрес, настроенный для VRRP-группы <b>n</b> .                                                                                                       |
| <code>no vrrp n int<br/>&lt;interface&gt;</code>                                                                                                          | Команда удаляет все пользовательские настройки, относящиеся к VRRP-группе <b>n</b> для интерфейса <code>&lt;interface&gt;</code> . Выставляются значения по умолчанию. |
| <div>  <b>Примечание</b> </div> <div>Сама группа следом удаляется.</div> |                                                                                                                                                                        |
| <code>no vrrp n int<br/>&lt;interface&gt; priority p</code>                                                                                               | Команда удаляет пользовательскую настройку приоритета для VRRP-группы <b>n</b> .                                                                                       |
| <code>no vrrp n int<br/>&lt;interface&gt; version v</code>                                                                                                | Команда удаляет версию протокола VRRP, настроенную для VRRP-группы <b>n</b> . Выставляется значение по умолчанию.                                                      |
| <code>no vrrp n int<br/>&lt;interface&gt;<br/>advertisement-interval i</code>                                                                             | Команда удаляет значение интервала отправки анонсов, настроенное для VRRP-группы <b>n</b> . Выставляется значение по умолчанию.                                        |
| <code>no vrrp n int<br/>&lt;interface&gt; shutdown</code>                                                                                                 | Команда включает ранее выключенную VRRP-группу <b>n</b> на интерфейсе <code>&lt;interface&gt;</code> .                                                                 |
| <code>no vrrp n int<br/>&lt;interface&gt; preempt</code>                                                                                                  | Команда выключает режим <b>preempt</b> для группы <b>n</b> , выборы не будут запускаться даже при более высоком приоритете группы <b>n</b> .                           |
| <code>no vrrp n int<br/>&lt;interface&gt;<br/>checksum-with-ipv4-pseudo</code>                                                                            | Команда выключает использование псевдо-заголовка IPv4 при расчете контрольной суммы VRRPv3 для VRRP-группы <b>n</b> .                                                  |

## Дефолтные команды для настройки и отмены

| Команда                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | Описание                                                                                                                                                                                                                                                                 | Значение по умолчанию                                            |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------|
| <code>vrrp default advertisement-inte</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | Команда устанавливает значение интервала отправки анонсов для всех новых VRRP-групп при их создании. Значение задается в миллисекундах, но должно быть кратно 10, так как VRRP использует единицы измерения в сотых долях секунды (i от 10 до 40950 в мс).               | 1000 мс (1 с)                                                    |
| <code>vrrp default shutdown</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | Команда устанавливает значение "выключено" для всех новых VRRP-групп при их создании, то есть переводит их в выключенное состояние сразу после загрузки или применения конфигурации. Режим shutdown можно выключить глобально ( <code>no vrrp default shutdown</code> ). | Enabled (режим default shutdown включен, новые группы выключены) |
| <div>  <b>Примечание</b> </div> <p>После перезагрузки Angie ADC при включенном режиме <code>vrrp default shutdown</code> все VRRP-группы переводятся в выключенное состояние, в том числе те, которые были включены вручную. Чтобы заново включить конкретную группу после перезагрузки, введите последовательно следующие две команды:</p> <pre>vrrp n int &lt;interface&gt; shutdown</pre> <pre>no vrrp n int &lt;interface&gt; shutdown</pre> |                                                                                                                                                                                                                                                                          |                                                                  |
| <code>vrrp default checksum-with-ipv4</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | Команда включает использование псевдо-заголовка IPv4 при расчете контрольной суммы VRRPv3 для всех новых VRRP-групп при их создании. Эта команда не влияет на VRRPv2 и IPv6.                                                                                             | Enabled (дефолтное использование включено)                       |
| <code>no vrrp default advertisement-inte</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | Команда удаляет пользовательское значение интервала отправки анонсов, настроенное для всех новых групп VRRP. Выставляется значение по умолчанию 1000 мс (1 сек.).                                                                                                        | n/a                                                              |
| <code>no vrrp default shutdown</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | Команда отключает настройку, по которой все новые VRRP-группы создаются выключенными. Новые VRRP-группы будут создаваться сразу включенными.                                                                                                                             | n/a                                                              |
| <code>no vrrp default checksum-with-ipv4</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | Команда отключает настройку, по которой для всех новых VRRP-групп при расчете контрольной суммы используется псевдо-заголовок IPv4.                                                                                                                                      | n/a                                                              |



## Команды для просмотра информации

| Команда                                                                                                 | Описание                                                                                                                                |
|---------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| <code>show vrrp</code>                                                                                  | Команда показывает информацию о состоянии всех VRRP-групп для всех интерфейсов.                                                         |
| <code>show vrrp int &lt;interface&gt;</code>                                                            | Команда показывает информацию о всех VRRP-группах для интерфейса <code>&lt;interface&gt;</code> .                                       |
| <code>show vrrp int &lt;interface&gt; n</code><br>или<br><code>show vrrp n int &lt;interface&gt;</code> | Команда показывает информацию о VRRP-группе <code>n</code> для интерфейса <code>&lt;interface&gt;</code> ( <code>n</code> от 1 до 255). |

## Создание и удаление VRRP-групп

Здесь описываются особенности создания и удаления VRRP-групп. Настройка VRRP осуществляется через *интерфейс командной строки (CLI) Angie ADC*.

Также можно посмотреть следующие статьи:

- *Настройка режима резервирования (Active-Standby) с помощью протокола VRRP;*
- *Настройка режима распределения нагрузки (Active-Active) с помощью протокола VRRP;*
- *Справочник по VRRP-командам.*

VRRP-группа — это логическая сущность, привязанная к физическому интерфейсу или его подынтерфейсу. Создание и удаление VRRP-групп вручную в Angie ADC не требуется. VRRP-группы создаются автоматически при введении команд, изменяющих конфигурацию по умолчанию, и удаляются также автоматически, если для группы не остается параметров, значения которых отличаются от значений по умолчанию. Это позволяет сохранять минимальную конфигурацию VRRP по умолчанию.

Вы можете просмотреть список всех VRRP-групп с помощью команды `show vrrp`.

Ниже приведены примеры ситуаций, в которых VRRP-группа будет создана или удалена.

### Создание VRRP-группы

Новая VRRP-группа создается автоматически, если изменяется значение по умолчанию любого из ее параметров.

Например, если для VRRP-группы 10, привязанной к интерфейсу `ens33`, указать IP-адрес (по умолчанию не указан), то указанная группа будет создана автоматически:

```
vrrp 10 int ens33 ip 192.168.0.1/24
```

Если для VRRP-группы 11, привязанной к интерфейсу `ens33`, указать значение параметра `preempt` равным `disabled` (по умолчанию `enabled`), то указанная группа также будет создана автоматически:

```
no vrrp 11 int ens33 preempt
```

## Удаление VRRP-группы

Рассмотрим пример. Для интерфейса **ens33** создано две VRRP-группы, с параметром **preempt** со значением **disabled** (выключено):

```
no vrrp 10 int ens33 preempt
```

```
no vrrp 11 int ens33 preempt
```

Группа удаляется в следующих случаях:

- Администратор ввел команду, возвращающую все параметры группы к значениям по умолчанию, в результате чего группа удаляется. Пример удаления группы 10:

```
no vrrp 10 int ens33
```

- Администратор вернул параметр группы к значению по умолчанию, при этом другие параметры группы уже имеют значения по умолчанию. Группа будет автоматически удалена, так как ее настройка больше не содержит ни одного параметра, значение которого отличается от значения по умолчанию.

Пример удаления группы 11 (параметру **preempt** присваивается значение по умолчанию **enabled**):

```
vrrp 11 int ens33 preempt
```

## 5.4 Использование протокола BFD для уменьшения времени реакции

### 5.4.1 Введение

Стандартные таймеры протоколов маршрутизации слишком велики и не подходят для современных сетей. Уменьшить время реакции протокола на какое-либо событие (непрямое падение, *indirect failure*) можно двумя способами:

- за счет уменьшения значения таймеров (может привести к увеличению нагрузки);
- с помощью специализированного протокола — BFD (предпочтительный способ).

BFD (Bidirectional Forwarding Detection) — это легковесный протокол, используемый для быстрого обнаружения отказов сетевых каналов между двумя устройствами. Он работает независимо и комплементарно к протоколам маршрутизации и помогает быстро определить, что связь между узлами нарушена. Если в инфраструктуре важна быстрая реакция на сбой, BFD значительно уменьшает время сходимости сети, так как позволяет получить субсекундное время реакции.

Как работает BFD?

1. Установление сессии: два сетевых устройства обмениваются контрольными пакетами BFD для установления связи.
2. Мониторинг: оба узла отправляют друг другу небольшие пакеты с фиксированным интервалом.
3. Обнаружение отказа: если один из узлов перестает получать пакеты от соседа в течение заданного времени, соединение объявляется "down".
4. Реакция: протоколы маршрутизации отслеживают состояние BFD-сессий и разрывают свои соединения, что позволяет переключиться на резервный канал менее, чем за секунду после сбоя.

**i Примечание**

Для настройки используется *командная строка Angie ADC CLI*.

## 5.4.2 Настройка с OSPF

Для протокола OSPF поддержка BFD активируется на интерфейсе с помощью команды `ip ospf bfd`.

Пример конфигурации интерфейса:

```
interface ens33
ip address 192.168.0.2/24
ip ospf 1 area 0
ip ospf bfd
ip ospf bfd profile test
```

Если параметры работы протокола BFD на каком-либо интерфейсе должны отличаться от стандартных, то новые параметры можно указать путем привязки профиля.

Протокол OSPF осуществляет динамическое обнаружение соседей на интерфейсе, а затем протокол BFD пытается установить с ними соседство:

```
angie-va# sho bfd peers
BFD Peers:
peer 192.168.0.3 vrf default interface ens33
ID: 192599120
Remote ID: 626157519
Active mode
Status: up
Uptime: 4 second(s)
Diagnostics: ok
Remote diagnostics: ok
Peer Type: dynamic
RTT min/avg/max: 0/0/0 usec
Local timers:
Detect-multiplier: 3
Receive interval: 300ms
Transmission interval: 300ms
Echo receive interval: 50ms
Echo transmission interval: disabled
Remote timers:
Detect-multiplier: 3
Receive interval: 300ms
Transmission interval: 300ms
Echo receive interval: 50ms
```

Протокол BFD может установить сессию с каким-либо соседом через определенный интерфейс и одновременно с этим не установить сессию с другим соседом. Это ожидаемое поведение для соседей, не обладающих поддержкой BFD. OSPF-соединение при этом не разрывается.

Если протокол BFD смог установить сессию с каким-либо из OSPF-соседей, то доступность этого соседа будет определяться с помощью протокола BFD.

Если протокол BFD не смог установить сессию с каким-либо из OSPF-соседей, то доступность этого соседа будет определяться с помощью встроенных механизмов OSPF (протокол Hello). При этом команда `sho bfd peers` будет отображать такого соседа в статусе `down`:

```
angie-v# sho bfd peers
BFD Peers:
peer 192.168.0.3 vrf default interface ens33
ID: 192599120
Remote ID: 0
Active mode
Status: down
Downtime: 4 minute(s), 0 second(s)
Diagnostics: ok
Remote diagnostics: ok
Peer Type: dynamic
RTT min/avg/max: 0/0/0 usec
Local timers:
Detect-multiplier: 3
Receive interval: 300ms
Transmission interval: 300ms
Echo receive interval: 50ms
Echo transmission interval: disabled
Remote timers:
Detect-multiplier: 3
Receive interval: 1000ms
Transmission interval: 1000ms
Echo receive interval: disabled
```

### 5.4.3 Настройка с BGP

По умолчанию при указании BGP-соседа BFD-сессия между маршрутизаторами не устанавливается.

Пример такой конфигурации:

```
angie-v# sho run
! Часть конфига, не относящаяся к BGP/BFD, удалена для краткости.
router bgp 3
no bgp ebgp-requires-policy
neighbor 192.168.0.2 remote-as 2
exit
!
angie-v# sho ip bg su
IPv4 Unicast Summary:
BGP router identifier 3.3.3.3, local AS number 3 VRF default vrf-id 0
BGP table version 0
RIB entries 0, using 0 bytes of memory
Peers 1, using 24 KiB of memory
Neighbor V AS MsgRcvd MsgSent TblVer InQ OutQ Up/Down State/
 PfxRcd PfxSnt Desc
192.168.0.2 4 2 8 9 0 0 0 00:05:35
 0 0 N/A
! BGP сессия успешно установлена, хотя никаких BFD-соседей у системы нет.
Total number of neighbors 1

angie-v# sho bfd pe
BFD Peers:
angie-v#
```

Чтобы активировать установление BFD-сессии с BGP-соседом, установите в секции настройки маршрутизатора BGP опцию `bfd` для соседа:

```
router bgp 2
no bgp ebgp-requires-policy
neighbor 192.168.0.3 remote-as 3
neighbor 192.168.0.3 bfd
exit
```

Чтобы BFD-сессия поднялась, необходимо включить поддержку BFD в настройках процесса маршрутизации BGP с обеих сторон. Еще одним способом является создание BFD-соседа вручную.

BGP-сессия успешно устанавливается и обновления передаются даже в ситуации, когда BFD-сессия не была установлена. Однако, если BFD-сессия была установлена, а затем разорвана, то автоматически будет разорвана и соответствующая BGP-сессия (после чего будут предприняты попытки переустановить BGP-сессию).

```
angie-va# sho bfd peers
BFD Peers:
peer 192.168.0.3 local-address 192.168.0.2 vrf default interface ens33
ID: 2213541843
Remote ID: 0
Active mode
Status: down
Downtime: 5 second(s)
Diagnostics: neighbor signaled session down
Remote diagnostics: neighbor signaled session down
Peer Type: dynamic
RTT min/avg/max: 0/0/0 usec
Local timers:
Detect-multiplier: 3
Receive interval: 300ms
Transmission interval: 300ms
Echo receive interval: 50ms
Echo transmission interval: disabled
Remote timers:
Detect-multiplier: 3
Receive interval: 300ms
Transmission interval: 300ms
Echo receive interval: 50ms
```

```
angie-va# sho ip bg su
```

```
IPv4 Unicast Summary:
BGP router identifier 2.2.2.2, local AS number 2 VRF default vrf-id 0
BGP table version 0
RIB entries 0, using 0 bytes of memory
Peers 1, using 24 KiB of memory
```

| Neighbor    | V      | AS   | MsgRcvd | MsgSent | TblVer | InQ | OutQ | Up/Down  | State/ |
|-------------|--------|------|---------|---------|--------|-----|------|----------|--------|
| ↪ PfxRcd    | PfxSnt | Desc |         |         |        |     |      |          |        |
| 192.168.0.3 | 4      | 3    | 27      | 30      | 0      | 0   | 0    | 00:00:06 | ↪      |
| ↪ 0         | 0      | N/A  |         |         |        |     |      |          |        |

```
Total number of neighbors 1
```

#### 5.4.4 Настройка с VRRP

На данный момент нет интеграции с BFD.

#### 5.4.5 Настройка со статическими маршрутами

Angie ADC поддерживает условное добавление статического маршрута (в зависимости от BFD-статуса соседа, который указан в качестве next-hop). То есть, если BFD-сессия до устройства, адрес которого указан как next-hop, поднимается, то статический маршрут добавляется в таблицу маршрутизации. Если сессия остается в статусе **down** (или падает в процессе эксплуатации позднее), маршрут не добавляется (удаляется) из таблицы маршрутизации.

Добавить статический маршрут с проверкой доступности next-hop по BFD можно, например, с помощью следующей команды:

```
ip route 3.3.3.3/32 192.168.0.3 bfd
```

Сразу после создания такого статического маршрута соответствующая BFD-сессия находится в статусе **down**:

```
angie-va# sho bfd pee
BFD Peers:
peer 192.168.0.3 vrf default
ID: 1753656330
Remote ID: 0
Active mode
Status: down
Downtime: 6 second(s)
Diagnostics: ok
Remote diagnostics: ok
Peer Type: dynamic
RTT min/avg/max: 0/0/0 usec
Local timers:
Detect-multiplier: 3
Receive interval: 300ms
Transmission interval: 300ms
Echo receive interval: 50ms
Echo transmission interval: disabled
Remote timers:
Detect-multiplier: 3
Receive interval: 1000ms
Transmission interval: 1000ms
Echo receive interval: disabled
```

Чтобы сессия поднялась, на соседе нужно либо настроить собственный статический маршрут с указанием адреса исходящего интерфейса в качестве next-hop, либо вручную создать статический пир, соответствующий системе балансировки (см. ниже).

### 5.4.6 Настройка независимого пира

Во всех приведенных выше случаях пир создавался динамически по требованию протокола динамической маршрутизации или в момент создания соответствующего статического маршрута. Ручное создание BFD-пира может потребоваться, например, в ситуации, когда необходимо поменять параметры работы какого-то одного соседа из группы автоматически обнаруженных.

Переход к созданию и настройке отдельного пира из режима конфигурирования протокола BFD:

```
angie-v# conf t
angie-v(config)# bfd
angie-v(config-bfd)# peer
A.B.C.D IPv4 peer address
X:X::X:X IPv6 peer address
angie-v(config-bfd)# peer
```

Некоторые настройки необходимо задать в момент создания пира (интерфейс, локальный адрес, multihop и vrf), остальные параметры можно будет изменить потом.

```
angie-v(config-bfd)# peer 192.168.0.3
<cr>
interface Interface information
local-address Configure local address
multihop Configure multihop
vrf Configure VRF
angie-v(config-bfd)# peer 192.168.0.3
angie-v(config-bfd-peer)#
detect-multiplier Configure peer detection multiplier
echo Configure peer echo intervals
echo-interval Configure peer echo intervals
echo-mode Configure echo mode
end End current mode and change to enable mode
exit Exit current mode and down to previous mode
find Find CLI command matching a regular expression
list Print command list
minimum-ttl Expect packets with at least this TTL
no Negate a command or set its defaults
output Direct vtysh output to file
passive-mode Don't attempt to start sessions
profile Use BFD profile settings
quit Exit current mode and down to previous mode
receive-interval Configure peer receive interval
shutdown Disable BFD peer
transmit-interval Configure peer transmit interval
```

После того, как с одной из сторон такой пир сконфигурирован, он будет находиться в статусе **down**, пока не будет создан симметричный пир с "противоположной" стороны.

### 5.4.7 Профиль

Чтобы создать новый профиль, отредактировать или удалить существующий профиль, необходимо перейти из режима глобальной конфигурации в режим настройки протокола BFD, а дальше уже либо удалять профиль, либо переходить в режим создания или настройки конкретного профиля.

```

angie-va(config)# bfd
angie-va(config-bfd)#
end End current mode and change to enable mode
exit Exit current mode and down to previous mode
find Find CLI command matching a regular expression
list Print command list
no Negate a command or set its defaults
output Direct vtysh output to file
peer Configure peer
profile BFD profile.
quit Exit current mode and down to previous mode
angie-va(config-bfd)# profile
BFDPROF BFD profile name.
test
angie-va(config-bfd)# no profile test
angie-va(config-bfd)# profile test_new
angie-va(config-bfd-profile)#
detect-multiplier Configure peer detection multiplier
echo Configure peer echo intervals
echo-interval Configure peer echo interval
echo-mode Configure echo mode
end End current mode and change to enable mode
exit Exit current mode and down to previous mode
find Find CLI command matching a regular expression
list Print command list
minimum-ttl Expect packets with at least this TTL
no Negate a command or set its defaults
output Direct vtysh output to file
passive-mode Don't attempt to start sessions
quit Exit current mode and down to previous mode
receive-interval Configure peer receive interval
shutdown Disable BFD peer
transmit-interval Configure peer transmit interval
angie-va(config-bfd-profile)#

```

## 5.5 Настройка RHI

RHI (Route Health Injection) позволяет динамически управлять маршрутами, анонсируемыми протоколами динамической маршрутизации, например BGP и OSPF, на основе данных о состоянии апстримов в балансировщике нагрузки. RHI позволяет автоматически отзывать префиксы при недоступности апстримов, минимизируя возможные циклы объявлений и отзывов. Апстрим считается недоступным, если все его бэкенды не прошли проверки работоспособности (health probes).

Для работы RHI необходимо:

1. Настроить BGP или OSPF в зависимости от свойств сети (через командную строку, примеры см. ниже).
2. Задать проверки работоспособности для балансировщика нагрузки (в консоли Angie ADC, пример см. ниже).
3. Задать конфигурацию RHI (в консоли Angie ADC, инструкцию см. ниже).



### 5.5.1 Настройка BGP

Настройка BGP проводится через *интерфейс командной строки*.

#### Пример

```
hostname angie-va
no ipv6 forwarding
ip prefix-list myapp1_pair seq 5 permit 2.2.2.2/32
interface lo
 ip address 2.2.2.2/32
exit
router bgp 1
 no bgp ebgp-requires-policy
 neighbor 10.0.0.1 remote-as 2
 address-family ipv4 unicast
 redistribute connected route-map myapp1_pair
 exit-address-family
exit
route-map myapp1_pair permit 10
 match ip address prefix-list myapp1_pair
exit
```

#### Проверка

Angie ADC должен увидеть свои объявления:

```
angie-va# sh ip bgp
BGP table version is 14, local router ID is 2.2.2.2, vrf id 0
Default local pref 100, local AS 1
Status codes: s suppressed, d damped, h history, u unsorted, * valid, > best, =_
↳multipath,
 i internal, r RIB-failure, S Stale, R Removed
Nexthop codes: @NNN nexthop's vrf id, < announce-nh-self
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

 Network Next Hop Metric LocPrf Weight Path
*> 2.2.2.2/32 0.0.0.0 0 32768 ?

Displayed 1 routes and 1 total paths
```

Соседний маршрутизатор должен увидеть объявления от Angie ADC:

```
abstract-router# sh ip bgp
BGP table version is 14, local router ID is 10.0.0.1, vrf id 0
Default local pref 100, local AS 2
Status codes: s suppressed, d damped, h history, u unsorted, * valid, > best, =_
↳multipath,
 i internal, r RIB-failure, S Stale, R Removed
Nexthop codes: @NNN nexthop's vrf id, < announce-nh-self
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

 Network Next Hop Metric LocPrf Weight Path
*> 2.2.2.2/32 10.21.20.6 0 0 1 ?
```

Displayed 1 routes and 1 total paths

В случае если объявления не видны, можно попробовать сбросить сессию BGP на обоих маршрутизаторах:

```
clear ip bgp * soft
```

### Отзыв префиксов

Отзыв префиксов реализован через автоматическое изменение префикс-листов.

Например:

```
ip prefix-list myapp1_pair seq 5 deny 2.2.2.2/32
```

### Объявление префиксов после восстановления работы апстрима

При восстановлении работоспособности апстримов префикс-листы не меняются автоматически.

Чтобы вручную вернуть префикс-лист к изначальному виду, выполните команды:

```
conf t
ip prefix-list myapp1_pair seq 5 permit 2.2.2.2/32
```

#### Примечание

Если вы хотите, чтобы префиксы автоматически анонсировались после восстановления апстримов, установите флажок **Объявлять префиксы при восстановлении апстримов** в настройках конфигурации RHI.

## 5.5.2 Настройка OSPF

Настройка OSPF также проводится через *интерфейс командной строки*.

### Пример

```
hostname angie-va
no ipv6 forwarding
ip prefix-list myapp1_pair seq 5 permit 2.2.2.2/32
interface ens33
ip address 192.168.0.20/24
ip ospf 1 area 0
exit
interface lo
ip address 2.2.2.2/32
exit
router ospf 1
ospf router-id 192.168.0.20
redistribute connected route-map myapp1_pair
exit
route-map myapp1_pair permit 10
```

```
match ip address prefix-list myapp1_pair
exit
```

## Проверка

Если серверы в бэкенде доступны:

```
angie-v#sho ip ospf 1 database external self-originate
OSPF Instance: 1
 OSPF Router with ID (2.2.2.2)
 AS External Link States
LS age: 28
Options: 0x2 : *|---|E|
LS Flags: 0xb
LS Type: AS-external-LSA
Link State ID: 2.2.2.2 (External Network Number)
Advertising Router: 192.168.0.20
LS Seq Number: 80000001
Checksum: 0x139b
Length: 36
Network Mask: /32
 Metric Type: 2 (Larger than any link state path)
 TOS: 0
 Metric: 20
 Forward Address: 0.0.0.0
 External Route Tag: 0
```

Если серверы в бэкенде недоступны:

```
angie-v#sho ip ospf 1 database external self-originate
OSPF Instance: 1
 OSPF Router with ID (192.168.0.20)
 AS External Link States
LS age: 3600
Options: 0x2 : *|---|E|
LS Flags: 0x8b
LS Type: AS-external-LSA
Link State ID: 2.2.2.2 (External Network Number)
Advertising Router: 2.2.2.2
LS Seq Number: 80000001
Checksum: 0x139b
Length: 36
Network Mask: /32
 Metric Type: 2 (Larger than any link state path)
 TOS: 0
 Metric: 20
 Forward Address: 0.0.0.0
 External Route Tag: 0
```

## Отзыв префиксов

Отзыв маршрута в OSPF происходит следующим образом: после удаления префикса OSPF изменяет значение возраста LSA в поле `LS age` на 3600, информируя соседей, что LSA устарела и нужно перестать ее использовать. Через некоторое время эта LSA будет полностью удалена из LSDB.

## Объявление префиксов после восстановления работы апстрима

При восстановлении работоспособности апстримов префикс-листы не меняются автоматически.

Чтобы вручную вернуть префикс-лист к изначальному виду, выполните команды:

```
conf t
ip prefix-list myapp1_pair seq 5 permit 2.2.2.2/32
```

### Примечание

Если вы хотите, чтобы префиксы автоматически анонсировались после восстановления апстримов, установите флажок **Объявлять префиксы при восстановлении апстримов** в настройках конфигурации RHI.

## 5.5.3 Настройка балансировщика нагрузки

Конфигурация балансировщика нагрузки настраивается в консоли Angie ADC (см. *инструкцию*).

Для работы RHI необходимо настроить следующие параметры:

- `upstream_probe`;
- параметры для получения информации о состоянии балансировщика нагрузки.

## Пример

Приведены только строки, добавленные в дефолтную конфигурацию.

```
Часть конфигурации опущена для краткости
}
Бэкенды
upstream myapp1 {
 zone myapp1 1m;
 server server1;
 server server2;
}
Переменная для health probe
map $upstream_status $good {
 200 "1";
}
server {
 ### Часть конфигурации опущена для краткости
 location / {
 proxy_pass http://myapp1;
 # Апстрим-пробы
 upstream_probe myapp1_probe
 interval=1s
 test=$good
 essential
```

```

 mode=always;
 }
 ### Часть конфигурации опущена для краткости
}
}

```

### 5.5.4 Настройка конфигурации RNI

Конфигурация RNI настраивается в консоли Angie ADC.

Чтобы настроить конфигурацию, выполните следующие действия:

1. Откройте консоль Angie ADC.
2. В правом верхнем углу выберите **Настройки** → **Конфигурация RNI**.  
Откроется окно настройки конфигурации RNI.
3. Задайте префиксы. Для этого нажмите **Добавить** и укажите следующие параметры:
  - **VIP-адрес (CIDR)** — виртуальный IP-адрес для апстрима. Указывается в формате **X.X.X.X/mask**.

#### **Примечание**

Значение должно быть уникальным в рамках одного префикс-листа.

- **Приоритет** — номер последовательности, определяющий порядок обработки правил. Маршрут с меньшим приоритетом обрабатывается первым.

#### **Примечание**

Значение должно быть уникальным в рамках одного префикс-листа.

- **Имя префикс-листа** — имя префикс-листа для маршрутизации.
  - **Имя апстрима** — апстрим, для которого настраивается отзыв префикса.
4. Установите флажок **Объявлять префиксы при восстановлении апстримов**, если хотите, чтобы Angie ADC автоматически анонсировал префиксы в маршрутизируемую сеть при восстановлении работоспособности апстримов. По умолчанию флажок снят.
  5. Нажмите **Сохранить**.

Настройки RNI будут сохранены и сразу применены.

### Удаление префиксов

Вы можете удалить неиспользуемые префиксы из конфигурации RNI, нажав на значок удаления напротив этих префиксов.

#### **Примечание**

При удалении префиксов через консоль Angie ADC они удаляются из базы данных и перестают влиять на объявление и отзыв маршрутов, но сохраняются в списках, отображаемых через командную строку.

Чтобы полностью удалить префикс-лист, необходимо выполнить следующие команды:

```
conf t
no ip prefix-list <имя префикс-листа>
```

Это поведение будет скорректировано в следующих релизах.

## ГЛАВА 6

### Высокая доступность

Высокая доступность (High Availability, HA) - это режим, при котором два устройства Angie ADC объединяются в пару высокой доступности (HA-пару), обеспечивающую автоматическую синхронизацию конфигураций двух узлов и сохранение работоспособности системы при сбоях на одном из узлов пары. Функциональность пары высокой доступности появилась в версии 0.4.0.

Пример типовой схемы работы пары высокой доступности:

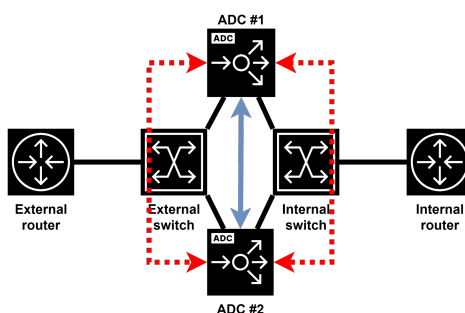


Рис. 1: Рис. 1

В этом разделе:

- *Обзор работы*
- *Создание пары высокой доступности*
- *Настройка высокой доступности в паре*
- *Управление парой*

#### **Примечание**

Рекомендуется резервировать все компоненты инфраструктуры. Отказ сопутствующей инфраструктуры (например выход из строя вышестоящего маршрутизатора или сбой серверов) выходит за рамки решения высокой доступности Angie ADC.

Пример отказоустойчивой инфраструктуры:

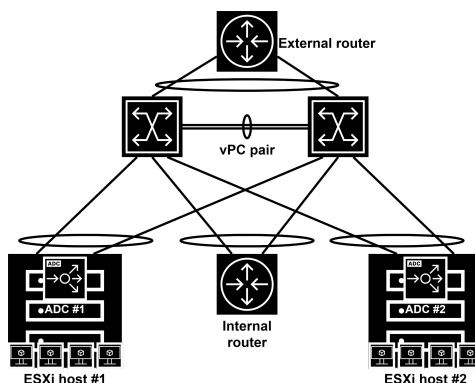


Рис. 2: Рис. 1

## 6.1 Обзор работы

Пара высокой доступности позволяет создать для Angie ADC горячий резерв в виде второго Angie ADC, полностью настроенного, синхронизированного и готового принять на себя всю нагрузку в случае сбоя основного устройства. После сборки пары можно в любой момент синхронизировать конфигурации узлов, входящих в пару, например при изменении настроек на основном узле.

### 6.1.1 Развертывание

Развертывание пары высокой доступности включает в себя несколько этапов:

1. Предварительная подготовка (см. *Предварительные действия*).
2. Создание пары высокой доступности (см. *Создание пары высокой доступности в режиме Active / Standby*).
3. Настройка механизмов, обеспечивающих отказоустойчивость (VRRP, OSPF, BGP), на узлах пары (см. *Настройка высокой доступности в паре*).

### 6.1.2 Режим Active / Standby

В первом релизе функциональности доступно создание пары высокой доступности в режиме Active / Standby. В дальнейшем будут добавлены другие режимы.

В режиме Active / Standby один узел пары является основным, а второй — резервным. Конфигурации обоих узлов синхронизированы. Основной узел принимает подключения и управляет серверами, а резервный узел отслеживает состояние основного узла и берет на себя его роль в случае сбоя. Сбой определяется как невозможность установить связь ни с одним из интерфейсов основного узла.

При перезагрузке первого узла после сбоя возможны следующие варианты:

- Стандартный сценарий: второй узел берет на себя роль Active, первый узел после перезагрузки переходит в Standby.
- При отсутствии связи со вторым узлом после перезагрузки первый узел снова берет на себя роль Active.
- Если оба узла пары перезагрузились одновременно, роль Active возьмет на себя тот узел, с которого создавалась пара.
- Если пара была удалена или второй узел теперь входит в другую пару, перезагрузившийся первый узел перейдет в состояние Standby.



### 6.1.3 Статус пары высокой доступности

У пары высокой доступности могут быть следующие статусы:

- **starting** – пара запускается. Статус отображается сразу после создания пары в процессе ее конфигурации и синхронизации.
- **on** – статус отображается при нормальной работе пары. Высокая доступность обеспечивается.
- **failed** – статус отображается после запуска пары при возникновении ошибок. Синхронизация и обмен данными между узлами в паре невозможны. Высокая доступность не обеспечивается.

### 6.1.4 Проверки доступности узлов пары

Резервный узел отслеживает состояние основного узла, периодически отправляя запросы (UDP-пакеты) на все активные интерфейсы основного узла. По умолчанию запросы отправляются с периодичностью в 250 мс, а таймаут ожидания ответа – 3 секунды. Эти значения можно настраивать. Если с момента получения последнего успешного ответа прошло больше времени, чем указано в таймауте ожидания, интерфейс будет считаться недоступным.

Если проверка доступности всех интерфейсов основного узла завершается неудачно, происходит переключение – резервный узел берет на себя роль основного.

Основной узел также отслеживает состояние резервного узла: если время, прошедшее с момента последнего запроса от резервного узла, превышает таймаут ожидания, то резервный узел будет считаться недоступным.

Запросы отправляются через UDP-порт 7007.

### 6.1.5 Синхронизация

При синхронизации основной узел передает свою конфигурацию на резервный узел. На резервном узле перезаписываются следующие файлы и папки:

- конфигурация балансировщика нагрузки;
- сертификаты и ключи балансировщика нагрузки;
- конфигурация маршрутизации (VRRP, BGP, OSPF);
- конфигурация GSLB;
- все файлы в папках `/etc/angie/crt` и `/etc/angie-lb/crt`;
- все файлы в папке `/var/transfer`.

Остальные файлы резервного узла удаляться не будут, если они находятся вне указанных папок (например файл лицензии).

Конфигурации узлов автоматически синхронизируются при создании пары. Начиная с версии 0.5.0 вы также можете запустить синхронизацию узлов пары вручную после создания пары.

## 6.2 Создание пары высокой доступности

Вы можете создать пару высокой доступности в консоли Angie ADC.

### 6.2.1 Требования

Для создания пары высокой доступности должны выполняться следующие условия:

- Оба устройства Angie ADC должны иметь одинаковую версию ПО и лицензию.

### 6.2.2 Предварительные действия

1. Разверните два устройства Angie ADC из будущей пары.

При этом должны выполняться следующие условия:

- В пару можно поставить устройства Angie ADC, у которых хотя бы одна общая подсеть.
- Интерфейсы самого устройства Angie ADC могут находиться в разных подсетях, но при формировании пар интерфейсов все IP-адреса попарно должны быть из одинаковых подсетей.
- Каждый узел должен иметь собственный набор уникальных IP-адресов. На данный момент поддерживаются только IPv4-адреса. Необходимо прописать IP-адреса для всех физических интерфейсов, которые будут использоваться для обеспечения высокой доступности.

Мы рекомендуем иметь одинаковое количество интерфейсов на обоих устройствах Angie ADC, чтобы каждый интерфейс получил пару на другом устройстве Angie ADC. Интерфейсы без пары не будут участвовать в обеспечении высокой доступности.

2. Настройте авторизацию на резервном узле Angie ADC с использованием логина **admin** (эти учетные данные необходимы для создания пары).
3. Настройте минимальную конфигурацию узла, который будет иметь роль основного. При создании пары запустится синхронизация конфигураций (конфигурация основного узла будет перенесена на резервный узел). Настройку также можно выполнить позже, после создания пары, при этом необходимо будет вручную запустить синхронизацию конфигураций после настройки.

### 6.2.3 Создание пары высокой доступности в режиме Active / Standby

При создании пары высокой доступности (HA-пары) роль основного узла (Active) автоматически назначается тому узлу, с которого создается пара, роль резервного (Standby) – добавляемому узлу.

Чтобы создать пару высокой доступности, выполните следующие действия:

1. Откройте консоль Angie ADC.
2. Перейдите на вкладку **Высокая доступность**.

Откроется экран подключения резервного узла к текущему узлу.

3. На экране **Создание пары высокой доступности** заполните следующие поля:

- **IP-адрес** – IP-адрес второго узла пары (IPv4) для удаленного администрирования и мониторинга этого узла. Это может быть выделенный интерфейс только для управления или один из интерфейсов, передающих продуктовый трафик. Этот адрес не распространяется и не синхронизируется.
- **Логин и Пароль** – данные для авторизации на резервном узле. Используйте логин **admin** (см. "Предварительные действия" выше).

4. Нажмите **Подключить**.

Между двумя узлами будет создано соединение. Откроется окно настройки соответствия интерфейсов.

5. В окне **Настройка соответствия интерфейсов** задайте соответствие между интерфейсами основного и резервного узлов. Для этого в раскрывающихся списках основного и резервного узла выберите интерфейсы, которые вы хотите поставить в пару. Чтобы добавить еще одну пару, нажмите значок **Добавить** и повторите действия.

**Примечание**

Адреса двух сопоставляемых интерфейсов должны быть из одной подсети.

Интерфейсу одного узла может соответствовать только один интерфейс другого узла. Минимально достаточно одной пары интерфейсов, но мы рекомендуем задать пары для всех интерфейсов обоих узлов.

В списках отображаются интерфейсы в состоянии **up**. В списках не отображаются и не доступны для выбора следующие интерфейсы: **loopback**, в состоянии **down**, без IP-адреса.

6. Нажмите **Сохранить**.

Запустится синхронизация конфигураций (основной узел передаст свою конфигурацию на резервный узел) и будет сформирована пара высокой доступности в режиме **Active / Standby**. Сразу после создания у пары будет отображаться статус **starting** (запуск). После успешной синхронизации статус поменяется на **on** (пара в рабочем состоянии).

## 6.3 Настройка высокой доступности в паре

После объединения двух устройств Angie ADC в пару необходимо настроить механизмы высокой доступности на узлах этой пары.

Для этого нужно разработать архитектуру сетевого сегмента, где устанавливаются два Angie ADC, и выбрать один из механизмов обеспечения высокой доступности. Поддерживаются следующие механизмы:

- VRRP;
- OSPF;
- BGP.

Можно одновременно использовать несколько механизмов на разных интерфейсах устройства. Настройку основного узла можно проводить до или после создания пары высокой доступности. При создании пары уже имеющиеся конфигурации синхронизируются, а после создания пары можно запустить ручную синхронизацию конфигураций.

### 6.3.1 VRRP

Настройки, относящиеся к VRRP, необходимо продублировать на двух узлах пары, запустив *синхронизацию пары*.

На активном узле виртуальный VRRP-интерфейс будет находиться в состоянии **UP**, а на резервном узле соответствующий виртуальный интерфейс будет выключен. При смене ролей меняется состояние виртуальных VRRP-интерфейсов на узлах (виртуальный VRRP-интерфейс узла, принявшего на себя роль **Active**, перейдет в состояние **UP**). Таким образом можно дополнительно организовать резервирование пары высокой доступности другим устройством или группой.

Пример вывода команды **ip a**, выполненной на активном узле (интерфейс **vrrp4-ens33-1@ens33** находится в состоянии **UP**):

```
ip a

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
 ↪qlen 1000
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
inet 127.0.0.1/8 scope host lo
 valid_lft forever preferred_lft forever
inet 203.0.113.1/32 brd 203.0.113.1 scope global lo
 valid_lft forever preferred_lft forever
inet6 ::1/128 scope host noprefixroute
 valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group
 ↪default qlen 1000
link/ether 00:0c:29:96:0c:dc brd ff:ff:ff:ff:ff:ff
altname enp2s1
inet 10.65.1.7/24 brd 10.65.1.255 scope global ens33
 valid_lft forever preferred_lft forever
inet6 fe80::20c:29ff:fe96:cdc/64 scope link
 valid_lft forever preferred_lft forever
3: vrrp4-ens33-1@ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
 ↪state UP group default qlen 1000
link/ether 00:00:5e:00:01:01 brd ff:ff:ff:ff:ff:ff
inet 10.65.1.5/24 metric 1000 scope global vrrp4-ens33-1
 valid_lft forever preferred_lft forever
inet6 fe80::200:5eff:fe00:101/64 scope link
 valid_lft forever preferred_lft forever
```

Пример вывода команды `ip a`, выполненной на резервном узле (интерфейс `vrrp4-ens33-1@ens33` находится в состоянии `DOWN`):

```
ip a

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
 ↪qlen 1000
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
inet 127.0.0.1/8 scope host lo
 valid_lft forever preferred_lft forever
inet 203.0.113.1/32 brd 203.0.113.1 scope global lo
 valid_lft forever preferred_lft forever
inet6 ::1/128 scope host noprefixroute
 valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group
 ↪default qlen 1000
link/ether 00:0c:29:c1:36:b4 brd ff:ff:ff:ff:ff:ff
altname enp2s1
inet 10.65.1.8/24 brd 10.65.1.255 scope global ens33
 valid_lft forever preferred_lft forever
inet6 fe80::20c:29ff:fec1:36b4/64 scope link
 valid_lft forever preferred_lft forever
5: vrrp4-ens33-1@ens33: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue
 ↪state DOWN group default qlen 1000
link/ether 00:00:5e:00:01:01 brd ff:ff:ff:ff:ff:ff protodown on protodown_reason <7>
inet 10.65.1.5/24 metric 1000 scope global vrrp4-ens33-1
 valid_lft forever preferred_lft forever
inet6 fe80::200:5eff:fe00:101/64 scope link
 valid_lft forever preferred_lft forever
```

Далее вы можете настроить высокую доступность двумя способами:

- Назначение адреса сервиса на виртуальный VRRP-интерфейс.
- Назначение адреса сервиса на интерфейс lo.

### Назначение адреса сервиса на виртуальный VRRP-интерфейс

При смене роли узла сервис будет перенесен на тот узел, который взял на себя роль Active. Допускается назначение нескольких IP-адресов одновременно на один виртуальный интерфейс. Для маршрутизатора адрес сервиса находится в подключенной сети, то есть доступен напрямую:

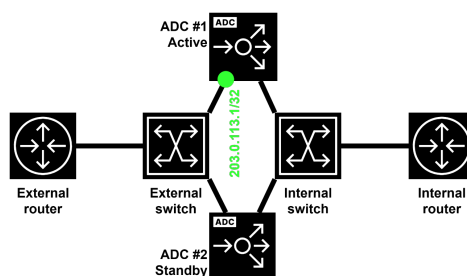


Рис. 3: Рис. 1

Если клиенты находятся в одной IP-сети с адресом сервиса, этот способ будет наиболее удобным:

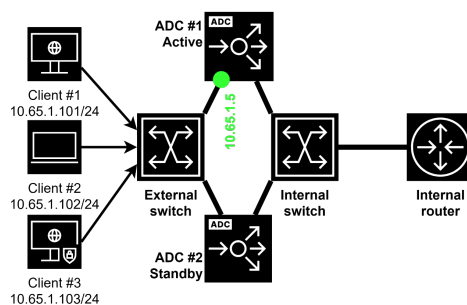


Рис. 4: Рис. 2

### Назначение адреса сервиса на интерфейс lo

При настройке статического маршрута на маршрутизаторе необходимо указать виртуальный VRRP-адрес как адрес следующего перехода (next-hop) для префикса, содержащего адрес сервиса. На интерфейс lo может быть назначено несколько адресов одновременно. Мы рекомендуем использовать этот способ как более гибкий:

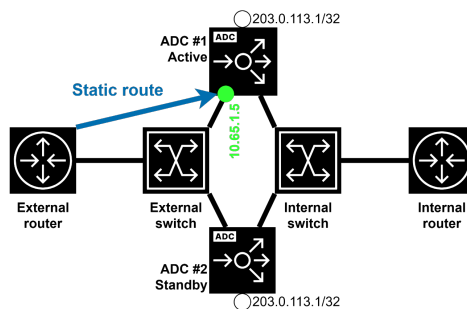


Рис. 5: Рис. 3

### 6.3.2 OSPF

Если в сети используются протоколы динамической маршрутизации, то вы можете добавить пару высокой доступности Angie ADC в OSPF-домен. При этом только основной узел пары будет анонсировать маршрут на адрес сервиса.

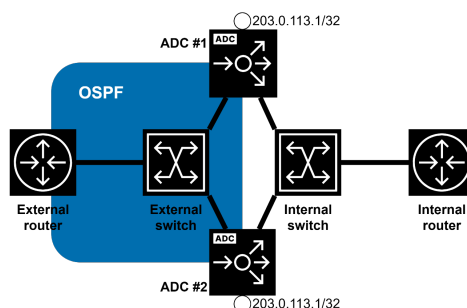


Рис. 6: Рис. 4

#### Пример

Сервис доступен по адресу 203.0.113.1. Необходимо, чтобы основной узел пары анонсировал маршрут на префикс 203.0.113.1/32.

Сначала нужно вручную назначить адрес 203.0.113.1/32 на интерфейсы loopback обоих узлов пары высокой доступности.

Интерфейсы основного узла:

```
ip a

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
 ↪ qlen 1000
 link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
 inet 127.0.0.1/8 scope host lo
 valid_lft forever preferred_lft forever
 inet 203.0.113.1/32 scope global lo
 valid_lft forever preferred_lft forever
 inet6 ::1/128 scope host noprefixroute
 valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group
 ↪ default qlen 1000
 link/ether 00:0c:29:96:0c:dc brd ff:ff:ff:ff:ff:ff
```

```
altname enp2s1
inet 10.65.1.7/24 brd 10.65.1.255 scope global ens33
valid_lft forever preferred_lft forever
inet6 fe80::20c:29ff:fe96:cdc/64 scope link
valid_lft forever preferred_lft forever
```

Интерфейсы резервного узла:

```
ip a

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default_
↪qlen 1000
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
inet 127.0.0.1/8 scope host lo
valid_lft forever preferred_lft forever
inet 203.0.113.1/32 scope global lo
valid_lft forever preferred_lft forever
inet6 ::1/128 scope host noprefixroute
valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group_
↪default qlen 1000
link/ether 00:0c:29:c1:36:b4 brd ff:ff:ff:ff:ff:ff
altname enp2s1
inet 10.65.1.8/24 brd 10.65.1.255 scope global ens33
valid_lft forever preferred_lft forever
inet6 fe80::20c:29ff:fec1:36b4/64 scope link
valid_lft forever preferred_lft forever
```

Далее необходимо добавить поддержку протокола OSPF на интерфейсе **ens33** и включить процесс маршрутизации OSPF, в котором анонсируются все подключенные сети.

```
interface ens33
ip ospf 1 area 0
router ospf 1
redistribute connected
distribute-list %ha%pair% out connected
```

После внесения изменений *синхронизируйте конфигурации в паре.*

#### **Примечание**

Последняя строка добавляется в момент выполнения синхронизации. Не нужно вносить эту строку вручную, а также удалять или изменять ее.

## Проверка

Проверка на маршрутизаторе или L3-коммутаторе (список OSPF-соседей):

```
R1#sho ip ospf nei
```

| Neighbor ID | Pri | State    | Dead Time | Address   | Interface          |
|-------------|-----|----------|-----------|-----------|--------------------|
| 1.1.1.1     | 1   | FULL/BDR | 00:00:39  | 10.65.1.7 | GigabitEthernet0/0 |
| 2.2.2.2     | 1   | FULL/DR  | 00:00:38  | 10.65.1.8 | GigabitEthernet0/0 |

Соседство по протоколу OSPF устанавливают оба узла пары одновременно:

```
R1#sho ip ro os
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2
i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
ia - IS-IS inter area, * - candidate default, U - per-user static route
o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
+ - replicated route, % - next hop override

Gateway of last resort is not set
203.0.113.0/32 is subnetted, 1 subnets
O E2 203.0.113.1 [110/20] via 10.65.1.7, 00:52:39, GigabitEthernet0/0
```

Маршрут анонсируется только одним узлом. В этом можно убедиться, если просмотреть список LSA5 в LSDB маршрутизатора:

```
R1#sho ip os database external
 OSPF Router with ID (10.65.1.140) (Process ID 1)

 Type-5 AS External Link States
Routing Bit Set on this LSA in topology Base with MTID 0
LS age: 1341
Options: (No TOS-capability, No DC, Upward)
LS Type: AS External Link
Link State ID: 203.0.113.1 (External Network Number)
Advertising Router: 1.1.1.1
LS Seq Number: 80000005
Checksum: 0x575B
Length: 36
Network Mask: /32
Metric Type: 2 (Larger than any link state path)
MTID: 0
Metric: 20
Forward Address: 0.0.0.0
External Route Tag: 0
```

Теперь выключите основной узел и проверьте изменения:

```
R1#sho ip os ne
```

| Neighbor ID | Pri | State   | Dead Time | Address   | Interface          |
|-------------|-----|---------|-----------|-----------|--------------------|
| 2.2.2.2     | 1   | FULL/DR | 00:00:33  | 10.65.1.8 | GigabitEthernet0/0 |

```
R1#sho ip os da ex
 OSPF Router with ID (10.65.1.140) (Process ID 1)

 Type-5 AS External Link States
Routing Bit Set on this LSA in topology Base with MTID 0
LS age: 114
Options: (No TOS-capability, No DC, Upward)
LS Type: AS External Link
Link State ID: 203.0.113.1 (External Network Number)
Advertising Router: 2.2.2.2
LS Seq Number: 80000001
Checksum: 0x139B
Length: 36
Network Mask: /32
```



```
Metric Type: 2 (Larger than any link state path)
MTID: 0
Metric: 20
Forward Address: 0.0.0.0
External Route Tag: 0
```

```
R1#sho ip ro os
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2
i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
ia - IS-IS inter area, * - candidate default, U - per-user static route
o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
+ - replicated route, % - next hop override

Gateway of last resort is not set
203.0.113.0/32 is subnetted, 1 subnets
O E2 203.0.113.1 [110/20] via 10.65.1.8, 00:01:57, GigabitEthernet0/0
R1#
```

Остался только один OSPF-сосед, который стал основным узлом пары. Теперь он анонсирует маршрут до префикса 203.0.113.1/32.

### 6.3.3 BGP

Маршрутизатор устанавливает BGP-сессии с обоими узлами пары. Обе BGP-сессии активны одновременно, но анонс префикса производится только с основного узла.

Пример типовой схемы:

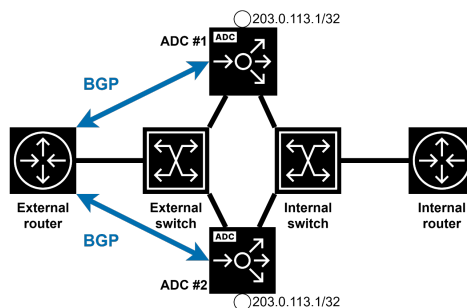


Рис. 7: Рис. 5

#### Пример

Сервис доступен по адресу 203.0.113.1. Основной узел пары будет анонсировать маршрут на префикс 203.0.113.1/32.

Сначала необходимо вручную назначить адрес 203.0.113.1/32 на интерфейсы loopback обоих узлов пары высокой доступности.

Интерфейсы основного узла:

```
ip a

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
 ↪qlen 1000
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
inet 127.0.0.1/8 scope host lo
 valid_lft forever preferred_lft forever
inet 203.0.113.1/32 scope global lo
 valid_lft forever preferred_lft forever
inet6 ::1/128 scope host noprefixroute
 valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group
 ↪default qlen 1000
link/ether 00:0c:29:96:0c:dc brd ff:ff:ff:ff:ff:ff
altname enp2s1
inet 10.65.1.7/24 brd 10.65.1.255 scope global ens33
 valid_lft forever preferred_lft forever
inet6 fe80::20c:29ff:fe96:cdc/64 scope link
 valid_lft forever preferred_lft forever
```

Интерфейсы резервного узла:

```
ip a

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
 ↪qlen 1000
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
inet 127.0.0.1/8 scope host lo
 valid_lft forever preferred_lft forever
inet 203.0.113.1/32 scope global lo
 valid_lft forever preferred_lft forever
inet6 ::1/128 scope host noprefixroute
 valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group
 ↪default qlen 1000
link/ether 00:0c:29:c1:36:b4 brd ff:ff:ff:ff:ff:ff
altname enp2s1
inet 10.65.1.8/24 brd 10.65.1.255 scope global ens33
 valid_lft forever preferred_lft forever
inet6 fe80::20c:29ff:fec1:36b4/64 scope link
 valid_lft forever preferred_lft forever
```

На основном узле настройте процесс BGP так, чтобы он устанавливал соединение с маршрутизатором (10.65.1.140). Также перераспределите подключенные сети в BGP:

```
router bgp 65001
neighbor 10.65.1.140 remote-as 65001
!
address-family ipv4 unicast
redistribute connected
exit-address-family
exit
```

Запустите *синхронизацию конфигураций* между узлами пары высокой доступности, чтобы настроить процесс маршрутизации BGP на резервном узле.

### Примечание

Ручная настройка конфигурации на резервном узле не должна использоваться.

## Проверка

Пример конфигурации со стороны маршрутизатора:

```
router bgp 65001
bgp log-neighbor-changes
neighbor 10.65.1.7 remote-as 65001
neighbor 10.65.1.8 remote-as 65001
```

Убедитесь, что маршрутную информацию отправляет только основной узел (10.65.1.7):

```
R1#sho ip bg su
BGP router identifier 10.65.1.140, local AS number 65001
BGP table version is 17, main routing table version 17
2 network entries using 296 bytes of memory
2 path entries using 128 bytes of memory
1/1 BGP path/bestpath attribute entries using 136 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
BGP using 560 total bytes of memory
BGP activity 4/2 prefixes, 13/11 paths, scan interval 60 secs

Neighbor V AS MsgRcvd MsgSent TblVer InQ OutQ Up/Down State/
↪PfxRcd
10.65.1.7 4 65001 3283 3206 17 0 0 02:43:57 2
10.65.1.8 4 65001 28 29 17 0 0 00:01:17 0

R1#sho ip bgp
BGP table version is 17, local router ID is 10.65.1.140
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found
Network Next Hop Metric LocPrf Weight Path
r>i 10.65.1.0/24 10.65.1.7 0 100 0 ?
*>i 203.0.113.1/32 10.65.1.7 0 100 0 ?

R1#sho ip ro bgp
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2
i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
ia - IS-IS inter area, * - candidate default, U - per-user static route
o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
+ - replicated route, % - next hop override

Gateway of last resort is not set
203.0.113.0/32 is subnetted, 1 subnets
B 203.0.113.1 [200/0] via 10.65.1.7, 02:45:34
R1#
```

Теперь выключите основной узел и проверьте изменения:

```
R1#sho ip bg su
BGP router identifier 10.65.1.140, local AS number 65001
BGP table version is 21, main routing table version 21
2 network entries using 296 bytes of memory
2 path entries using 128 bytes of memory
1/1 BGP path/bestpath attribute entries using 136 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
BGP using 560 total bytes of memory
BGP activity 6/4 prefixes, 15/13 paths, scan interval 60 secs
```

| Neighbor  | V | AS    | MsgRcvd | MsgSent | TblVer | InQ | OutQ | Up/Down  | State/ |
|-----------|---|-------|---------|---------|--------|-----|------|----------|--------|
| ↪ PfxRcd  |   |       |         |         |        |     |      |          |        |
| 10.65.1.7 | 4 | 65001 | 0       | 0       | 1      | 0   | 0    | 00:07:13 | Active |
| 10.65.1.8 | 4 | 65001 | 228     | 222     | 21     | 0   | 0    | 00:11:04 | 2      |

Остался только один BGP-сосед (10.65.1.8), который стал основным узлом пары. Теперь он анонсирует маршрут до префикса 203.0.113.1/32:

```
R1#sho ip bg
BGP table version is 21, local router ID is 10.65.1.140
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found
Network Next Hop Metric LocPrf Weight Path
r>i 10.65.1.0/24 10.65.1.8 0 100 0 ?
*>i 203.0.113.1/32 10.65.1.8 0 100 0 ?
R1#sho ip ro bg
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2
i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
ia - IS-IS inter area, * - candidate default, U - per-user static route
o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
+ - replicated route, % - next hop override

Gateway of last resort is not set
203.0.113.0/32 is subnetted, 1 subnets
B 203.0.113.1 [200/0] via 10.65.1.8, 00:05:50
```

За анонс отвечает distribute-list с именем %ha%pair%, который автоматически прикрепляется ко всем BGP-соседям, описанным в конфигурации. Соответствующая конфигурационная строка добавляется в момент синхронизации конфигурации между узлами пары:

```
router bgp 65001
neighbor 10.65.1.140 remote-as 65001
!
address-family ipv4 unicast
redistribute connected
neighbor 10.65.1.140 distribute-list %ha%pair% out
exit-address-family
exit
```

Если узел является основным, то distribute-list ссылается на разрешающий список доступа:

```
access-list %ha%pair% seq 5 permit any
```

Если узел является резервным, то distribute-list ссылается на запрещающий список доступа:

```
access-list %ha%pair% seq 5 deny any
```

## 6.4 Управление парой

В этой статье:

- *Просмотр информации о паре*
- *Синхронизация узлов*
- *Настройка протокола опроса доступности узлов*
- *Удаление пары*

### 6.4.1 Просмотр информации о паре

Добавлено в версии 0.4.0.

Чтобы просмотреть информацию об имеющейся паре высокой доступности:

1. Откройте консоль Angie ADC на одном из узлов пары.
2. Перейдите на вкладку **Высокая доступность**.

Откроется экран просмотра узлов, входящих в пару.

Для каждого узла отображаются следующие параметры:

|                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ID                  | Идентификатор узла: <ul style="list-style-type: none"> <li>• 0 – текущий узел (узел, на котором в данный момент открыта консоль Angie ADC со свойствами пары);</li> <li>• 1 – соседний узел.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| Статус пары         | Статус пары высокой доступности. Возможны следующие варианты: <ul style="list-style-type: none"> <li>• <b>starting</b> – пара запускается. Статус отображается сразу после создания пары в процессе ее конфигурации и синхронизации.</li> <li>• <b>on</b> – статус отображается при нормальной работе пары. Высокая доступность обеспечивается.</li> <li>• <b>failed</b> – статус отображается после запуска пары при возникновении ошибок. Синхронизация и обмен данными между узлами в паре невозможны. Высокая доступность не обеспечивается.</li> </ul> <div> <p><b>Примечание</b></p> <p>Если в процессе работы пары произошло какое-либо событие, на иконке статуса появится знак вопроса. При наведении указателя мыши на иконку со знаком вопроса отображается сообщение о последнем событии в работе пары или о возникшей ошибке.</p> </div> |
| IP-адрес узла       | IP-адрес узла, используемый для взаимодействия в паре.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| Роль узла           | Роль узла пары. Возможны следующие варианты: <ul style="list-style-type: none"> <li>• <b>Основной</b> (находится в режиме Active – обрабатывает трафик);</li> <li>• <b>Резервный</b> (находится в режиме Standby, отслеживает состояние основного узла).</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| Последнее изменение | Дата и время последнего изменения в паре, например смены ролей узлов.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |

## 6.4.2 Синхронизация узлов

Добавлено в версии 0.5.0.

После внесения изменений в конфигурацию основного узла вы можете вручную запустить синхронизацию пары, чтобы перенести внесенные изменения на резервный узел.

На резервном узле перезаписываются следующие файлы и папки:

- конфигурация балансировщика нагрузки;
- сертификаты и ключи балансировщика нагрузки;
- конфигурация маршрутизации (VRRP, BGP, OSPF);
- конфигурация GSLB;
- все файлы в папках `/etc/angie/crt` и `/etc/angie-lb/crt`;
- все файлы в папке `/var/transfer`.

Чтобы синхронизировать узлы пары, выполните следующие действия:

1. Откройте консоль Angie ADC на одном из узлов пары.
2. Перейдите на вкладку **Высокая доступность**.  
Откроется экран просмотра свойств пары.
3. Нажмите **Синхронизировать**.

Конфигурация основного узла будет перенесена на резервный узел, на экране отобразится сообщение **Пара синхронизирована**. Если синхронизация завершится с ошибкой, на экране отобразится сообщение **Ошибка синхронизации**.

### 6.4.3 Настройка протокола опроса доступности узлов

Добавлено в версии 0.6.0.

Чтобы изменить настройки протокола опроса доступности узлов, выполните следующие действия:

1. Откройте консоль Angie ADC на одном из узлов пары.
2. Перейдите на вкладку **Высокая доступность**.

Откроется экран просмотра узлов, входящих в пару.

3. В блоке **Проверка доступности узлов** укажите следующие данные:

- **Интервал отправки запросов (мс)**

Интервал отправки запросов резервным узлом для проверки доступности основного узла. По умолчанию запросы отправляются каждые 250 мс. Допустимый диапазон значений – от 250 до 1000 мс.

- **Таймаут ожидания ответа (с)**

Таймаут ожидания ответа от интерфейса основного узла. Если с момента получения последнего успешного ответа прошло больше заданного времени интерфейс будет считаться недоступным. При недоступности всех интерфейсов основного узла произойдет переключение ролей. По умолчанию – 2 секунды. Допустимый диапазон значений – от 1 до 3 секунд.

4. Нажмите **Сохранить**.

Новые настройки будут применены сразу.

### 6.4.4 Удаление пары

Добавлено в версии 0.4.0.

Чтобы удалить пару, выполните следующие действия:

1. Откройте консоль Angie ADC на одном из узлов пары.
2. Перейдите на вкладку **Высокая доступность**.

Откроется экран просмотра свойств пары.

3. Нажмите **Расформировать** и подтвердите удаление.

Конфигурация пары высокой доступности будет удалена. Конфигурации узлов сохранятся, узлы продолжат функционировать независимо друг от друга.

## ГЛАВА 7

Управление

---

Вы можете управлять пользователями системы и выбрать тип репозитория для хранения резервной копии балансировщика нагрузки через веб-консоль Angie ADC.

В этом разделе:

- *Управление пользователями*
- *Выбор репозитория для резервного копирования*
- *Запуск командной строки (CLI)*
- *Справочник команд (CLI)*

## 7.1 Управление пользователями

### 7.1.1 Требования

- Логин — допускаются латинские буквы, цифры, а также специальные символы: `_ - . @ ! # $ % ^ & * ( ) = +`.
- Пароль — допускаются латинские буквы, цифры, а также специальные символы: `_ - . @ ! # $ % ^ & * ( ) = +`.  
Минимум восемь символов.
- Имя — допускаются только буквы латиницы и кириллицы, а также дефис и пробел.
- Фамилия — допускаются только буквы латиницы и кириллицы, а также дефис и пробел.



### 7.1.2 Добавление нового пользователя

Чтобы добавить нового пользователя консоли, выполните следующие действия:

1. Откройте консоль Angie ADC.
2. На вкладке **Пользователи** нажмите кнопку **+ Добавить пользователя**.
3. В открывшемся окне укажите логин, пароль, имя и фамилию нового пользователя и нажмите **Добавить**.

Новый пользователь отобразится в таблице.

### 7.1.3 Изменение учетных данных существующего пользователя

Чтобы изменить данные существующего пользователя консоли, выполните следующие действия:

1. Откройте консоль Angie ADC.
2. На вкладке **Пользователи** нажмите на ... (многоточие) → **Изменить** рядом с пользователем, данные которого вы хотите изменить.
3. В открывшемся окне укажите новые данные и нажмите **Сохранить**.

Данные пользователя будут обновлены.

### 7.1.4 Изменение статуса существующего пользователя

Возможны следующие статусы пользователя:

- **Активный** — пользователь имеет доступ к системе.
- **Неактивный** — у пользователя временно отключен доступ к системе, например, на время отпуска.
- **Удален** — при выборе этого статуса пользователь будет удален.

Чтобы изменить статус существующего пользователя консоли, выполните следующие действия:

1. Откройте консоль Angie ADC.
2. На вкладке **Пользователи** нажмите на ... (многоточие) → **Изменить** рядом с пользователем, данные которого вы хотите изменить.
3. В открывшемся окне поменяйте статус пользователя и нажмите **Сохранить**.

Данные пользователя будут обновлены.

### 7.1.5 Удаление пользователя

Чтобы удалить существующего пользователя консоли, выполните следующие действия:

1. Откройте консоль Angie ADC.
2. На вкладке **Пользователи** нажмите на ... (многоточие) → **Удалить** рядом с пользователем, запись о котором вы хотите удалить.

Пользователь будет удален из списка пользователей консоли. В дальнейшем вы сможете просмотреть список удаленных пользователей при аудите системы.

## 7.2 Выбор репозитория для резервного копирования

В Angie ADC поддерживаются два типа репозитория для хранения *резервных копий конфигурации балансировщика нагрузки*:

- локальный репозиторий `filesystem` (подключен по умолчанию);
- внешний репозиторий, подключаемый по протоколу S3.

Резервная копия в репозитории обновляется автоматически при внесении изменений и сохранении новой конфигурации балансировщика нагрузки.

### 7.2.1 Локальный репозиторий (filesystem)

При использовании локального репозитория `filesystem` резервные копии хранятся в файловой системе внутри Angie ADC. Локальный репозиторий подключен по умолчанию. Для надежного хранения данных рекомендуется перейти на внешний репозиторий.

#### Примечание

При подключении нового репозитория (локального или внешнего) резервная копия в локальном репозитории автоматически удаляется.

### 7.2.2 Внешний репозиторий (S3)

Мы рекомендуем использовать внешний S3-совместимый репозиторий для хранения резервных копий конфигурации. В отличие от локального репозитория, при отключении внешнего репозитория резервные копии в нем сохраняются.

### 7.2.3 Управление репозиториями

#### Просмотр информации о текущем репозитории

1. Откройте консоль Angie ADC.
2. В правом верхнем углу выберите **Настройки** → **Резервное копирование**.  
Откроется страница с информацией о текущем репозитории.

#### Подключение локального репозитория

1. Откройте консоль Angie ADC.
2. В правом верхнем углу выберите **Настройки** → **Резервное копирование**.  
Откроется страница с информацией о текущем репозитории.
3. Перейдите на вкладку **Filesystem**.
4. Нажмите **Подключить**.

Будет подключен локальный репозиторий и создана резервная копия конфигурации балансировщика.

#### Примечание

Если у вас до этого уже была сохранена резервная копия в `filesystem`, она будет удалена.

## Подключение внешнего репозитория (S3)

1. Откройте консоль Angie ADC.
2. В правом верхнем углу выберите Настройки → Резервное копирование.  
Откроется страница с информацией о текущем репозитории.
3. Перейдите на вкладку S3.
4. Заполните поля для подключения репозитория:

- Конечная точка
- Имя корзины
- Ключ доступа
- Секретный токен

5. Нажмите Подключить.

Если вы подключаете репозиторий, в котором уже была резервная копия конфигурации, то будет использована эта копия. Если подключается новый репозиторий, будет создана резервная копия текущей конфигурации балансировщика.

### Примечание

Если у вас до этого уже была сохранена резервная копия в `filesystem`, она будет удалена.

## 7.3 Запуск командной строки (CLI)

В Angie ADC доступно два интерфейса командной строки (CLI):

- CLI на порту 2222 (новая версия, рекомендуемый интерфейс для всех операций, кроме настройки VRRP);
- CLI на порту 2022 (старая версия).

### Важно

Настройка VRRP поддерживается только через CLI на порту 2022.

Через интерфейс командной строки вы можете управлять настройками и функциями Angie ADC, например:

- загружать сертификаты для включения безопасного соединения;
- настраивать маршрутизацию по протоколам *BGP*, *OSPF*, *VRRP*, *BFD*;
- настраивать *RHI*.

См. также *справочник команд командной строки*.

### 7.3.1 Запуск новой версии CLI (на порту 2222)

При запуске Angie ADC интерфейс командной строки Angie ADC будет доступен по SSH-протоколу на порту 2222. Доступ осуществляется с помощью SSH-клиента (если SSH-сервер запущен, то в журнале отобразится сообщение **starting ssh server on port <номер порта>**).

Чтобы запустить интерфейс командной строки (CLI), выполните следующие действия:

1. Подключитесь к SSH-серверу:

```
ssh -p 2222 user@localhost
```

2. После подключения сервер запросит пароль для авторизации. Введите пароль:

```
user@localhost's password:
#
```

Если авторизация прошла успешно, откроется терминал Angie ADC. Справка доступна по ?.

3. Для настройки протоколов BGP, OSPF и BFD дополнительно перейдите в настройки vtysh:

```
vttysh
```

Команда переключит вас в конфигурационный режим, где можно вносить изменения.

### 7.3.2 Запуск старой версии CLI (на порту 2022)

При запуске Angie ADC интерфейс командной строки Angie ADC будет доступен по SSH-протоколу на порту 2022. Доступ осуществляется с помощью SSH-клиента (если SSH-сервер запущен, то в журнале отобразится сообщение **starting ssh server on port <номер порта>**).

Чтобы запустить интерфейс командной строки (CLI), выполните следующие действия:

1. Подключитесь к SSH-серверу:

```
ssh -p 2022 user@localhost
```

2. После подключения сервер запросит пароль для авторизации. Введите пароль:

```
user@localhost's password:
#
```

Если авторизация прошла успешно, откроется терминал Angie ADC. Справка доступна по ?.

3. Для настройки протоколов BGP, OSPF и BFD дополнительно перейдите в настройки vtysh:

```
vttysh
```

Команда переключит вас в конфигурационный режим, где можно вносить изменения.

## 7.4 Справочник команд (CLI)

В Angie ADC доступно два интерфейса командной строки (CLI), позволяющих управлять настройками Angie ADC:

- CLI на порту 2222 (новая версия; рекомендуемый интерфейс для большинства операций, кроме настройки VRRP);
- CLI на порту 2022 (старая версия для настройки VRRP).

Возможности: автодополнение команд по TAB; поддержка истории команд; подсказки по синтаксису команд по нажатию ?; поддержка цветовой схемы; поддержка сокращенных команд.

См. также *инструкцию по запуску командной строки*.

### 7.4.1 Поддерживаемые команды

Ниже приведен список команд в алфавитном порядке для новой версии CLI (на порту 2222).

Основные команды:

|                      |                                                                                                   |
|----------------------|---------------------------------------------------------------------------------------------------|
| <i>cert-config</i>   | Управление сертификатами                                                                          |
| <i>configuration</i> | Переход в режим настройки Angie ADC                                                               |
| <i>diagnostics</i>   | Сетевая диагностика                                                                               |
| <i>firewall</i>      | Управление правилами брандмауэра                                                                  |
| <i>ip</i>            | Стандартное управление сетевыми интерфейсами, маршрутами, адресами и правилами IP-трафика в Linux |
| <i>logs</i>          | Просмотр журналов                                                                                 |
| <i>modules</i>       | Просмотр информации о динамических модулях                                                        |
| <i>ps</i>            | Отображение информации о процессах, запущенных в системе                                          |
| <i>settings</i>      | Просмотр и изменение системных настроек Angie ADC (NTP, отладка, настройка времени и т. д.)       |
| <i>system</i>        | Перезагрузка и обновление Angie ADC                                                               |
| <i>transfer</i>      | Работа с директорией <b>transfer</b>                                                              |
| <i>vttysh</i>        | Переход к настройкам маршрутизации по протоколам BGP и OSPF                                       |

Служебные команды:

|                        |                          |
|------------------------|--------------------------|
| <b>exit</b> или Ctrl+Z | Выход из контекста       |
| <b>quit</b> или Ctrl+Q | Завершение сессии        |
| <b>show</b>            | Вывод текущего состояния |
| <b>?</b>               | Список доступных команд  |

### 7.4.2 Синтаксис

```
command option0 {option1|option2|...|option N} [option3] [{option4|option5|...|option M}] [option6 <user-input1>] [<user-input2>] <user-input3>
```

Параметры:

|                                         |                                                       |
|-----------------------------------------|-------------------------------------------------------|
| <b>command</b>                          | команда                                               |
| <b>option0</b>                          | обязательный параметр                                 |
| <b>{option1 option2 ... option N}</b>   | выбор из списка обязательных параметров               |
| <b>[option3]</b>                        | необязательный параметр                               |
| <b>[{option4 option5 ... option M}]</b> | выбор из списка необязательных параметров             |
| <b>[option6 &lt;user-input1&gt;]</b>    | необязательный параметр с пользовательским аргументом |
| <b>[&lt;user-input2&gt;]</b>            | необязательный пользовательский аргумент              |
| <b>&lt;user-input3&gt;</b>              | обязательный пользовательский аргумент                |

### 7.4.3 cert-config

Контекст **cert-config** позволяет управлять сертификатами, используемыми Angie ADC.

Поддерживаемые команды:

|                    |                                                                                                                 |
|--------------------|-----------------------------------------------------------------------------------------------------------------|
| <b>cert-config</b> | Вход в контекст                                                                                                 |
| <i>disable</i>     | Отключение TLS<br>Добавлено в версии 0.6.0.                                                                     |
| <i>enable</i>      | Включение TLS<br>Добавлено в версии 0.6.0.                                                                      |
| <i>info</i>        | Просмотр информации о сертификате<br>Добавлено в версии 0.4.0.                                                  |
| <i>list</i>        | Вывод списка сертификатов <b>traffic</b> Angie ADC<br>Добавлено в версии 0.4.0.                                 |
| <i>match</i>       | Сопоставление пары <b>ключ-сертификат</b><br>Добавлено в версии 0.4.0.                                          |
| <i>tls-upload</i>  | Контекст загрузки сертификатов и ключей Angie ADC<br>Добавлено в версии 0.4.0.                                  |
| <i>tls-use</i>     | Контекст копирования сертификатов и ключей Angie ADC из директории <b>transfer</b><br>Добавлено в версии 0.4.0. |

#### disable

Выключает управление TLS-сертификатами.

Синтаксис: **disable management.**

#### enable

Включает управление TLS-сертификатами.

Синтаксис: **enable management.**

#### info

Просмотр информации о сертификате.

Синтаксис: **info {management | traffic [name <filename>]}.**

Параметры:

- **management** — просмотр информации о сертификате для управления Angie ADC;
- **traffic** — просмотр информации о сертификате для передачи данных Angie ADC;
- **name <filename>** — имя сертификата.

Пример:

```
(cert-config)$$ info traffic name site1.crt
Certificate:
 Data:
 Version: 3 (0x2)
 Serial Number: 1 (0x1)
 Signature Algorithm: sha1WithRSAEncryption
 Issuer: C = RU, ST = Moscow, O = Angie, OU = EVO, CN = Root CA
 Validity
```

```

 Not Before: Feb 18 12:23:08 2025 GMT
 Not After : Feb 18 12:23:08 2027 GMT
 Subject: CN = Demo Certificate, ST = Moscow, C = RU, O = Angie, OU = EVO
 Subject Public Key Info:
 Public Key Algorithm: rsaEncryption
 Public-Key: (4096 bit)
 Modulus:
...
(cert-config)$$

```

## list

Вывод списка сертификатов **traffic** Angie ADC.

Синтаксис: **list**.

Пример:

```

(cert-config)$$ list
cp.crt : invalid
cp.key : invalid
site1.crt: certificate
site1.key: key

```

Параметры:

- **invalid** — сертификат или ключ не прошли проверку;
- **certificate** — сертификат;
- **key** — ключ.

## match

Сопоставление пары **ключ-сертификат**.

Синтаксис: **match** <certificate> <key>.

Параметры:

- <certificate> — имя файла сертификата;
- <key> — имя файла ключа.

Пример:

```

(cert-config)$$ match site1.crt site1.key
Certificate and key match
(cert-config)$$ match site1.crt site2.key
Certificate and key DO NOT match
(cert-config)$$

```

## tls-upload

Контекст загрузки сертификатов и ключей Angie ADC.

Поддерживаемые команды:

|            |                                                              |
|------------|--------------------------------------------------------------|
| management | Загрузка сертификата и ключа для управления Angie ADC        |
| tls-upload | Вход в контекст                                              |
| traffic    | Загрузка сертификатов и ключей для передачи данных Angie ADC |

## management

Загрузка сертификата и ключа для управления Angie ADC.

Синтаксис: `management {crt|key} [name <filename>]`.

Параметры:

- `crt` — загрузка сертификата;
- `key` — загрузка ключа;
- `name <filename>` — имя сохраняемого файла (игнорируется для команды `management`).

Для загрузки сертификата или ключа необходимо вставить текст сертификата или ключа в строку, затем вставить пустую строку или написать `END` в новой строке.

Пример:

```
(cert-config-tls-upload)$$ management crt
Warning: filename will be ignored for management certificate
Insert the certificate line by line at the end of the input, add an empty line or
↪enter END
-----BEGIN CERTIFICATE-----
MIIF9TCCA92gAwIBAgIBATANBgkqhkiG9w0BAQUFADBOMQswCQYDVQQGEwJSVTEP
...
Z7SIEeCr84fIlxFMpRxNGZhJQ67xWLwQmjVcphi037Wpx+6dmQRtp8Q=
-----END CERTIFICATE-----
END
Successfully wrote to /etc/angie/crt/cp.crt
(cert-config-tls-upload)$$
```

При загрузке ключа и сертификата проверяется их корректность. В случае некорректного значения сертификат или ключ не будет сохранен:

Пример вывода:

```
(cert-config-tls-upload)$$ management crt
Warning: filename will be ignored for management certificate
Insert the certificate line by line at the end of the input, add an empty line or
↪enter END
simple cert

Validation error: Invalid certificate format
(cert-config-tls-upload)$$
```



## traffic

Загрузка сертификатов и ключей для передачи данных Angie ADC.

Синтаксис: **traffic {crt|key} [name <filename>]**.

Параметры:

- **crt** — загрузка сертификата;
- **key** — загрузка ключа;
- **name <filename>** — имя сохраняемого файла (если отсутствует, то будет использовано имя **cr.crt** для сертификата и **cr.key** для ключа).

Для загрузки сертификата или ключа необходимо вставить текст сертификата или ключа в строку, затем вставить пустую строку или написать **END** в новой строке.

Пример загрузки сертификата с именем:

```
(cert-config-tls-upload)$$ traffic crt name site1.crt
Insert the certificate line by line at the end of the input, add an empty line or
↪enter END
-----BEGIN CERTIFICATE-----
MIIF9TCCA92gAwIBAgIBATANBgqhkiG9w0BAQUFADBOMQswCQYDVQQGEwJSVTEP
...
Z7SIEeCr84fIlxFMpRxNGZhJQ67xWLwQmjVcphi037Wpx+6dmQRtp8Q=
-----END CERTIFICATE-----

Successfully wrote to /etc/angie-lb/crt/site1.crt
```

При загрузке ключа и сертификата проверяется их корректность. В случае некорректного значения сертификат или ключ не будет сохранен.

## tls-use

Контекст копирования сертификатов и ключей Angie ADC из директории **transfer**.

Поддерживаемые команды:

|                   |                                                                                               |
|-------------------|-----------------------------------------------------------------------------------------------|
| <b>management</b> | Копирование сертификата и ключа для управления Angie ADC из директории <b>transfer</b>        |
| <b>tls-use</b>    | Вход в контекст                                                                               |
| <b>traffic</b>    | Копирование сертификатов и ключей для передачи данных Angie ADC из директории <b>transfer</b> |

## management

Копирование сертификата и ключа для управления Angie ADC из директории **transfer**.

Синтаксис: **management {crt|key} <source\_filename> [name <destination\_filename>]**.

Параметры:

- **crt** — копирование сертификата;
- **key** — копирование ключа;
- **<source\_filename>** — имя файла в директории **transfer**;

- **name** <destination-filename> — имя сохраняемого файла (игнорируется для команды **management**).

Пример:

```
(cert-config-tls-use)$$ management crt cert1.crt
Warning: filename will be ignored for management certificate
Successfully wrote to /etc/angie/crt/cp.crt
(cert-config-tls-use)$$
```

При копировании сертификата или ключа проверяется их корректность. В случае некорректного значения сертификат или ключ не будет сохранен.

## traffic

Копирование сертификатов и ключей для передачи данных Angie ADC из директории **transfer**.

Синтаксис: **traffic** {crt|key} <source\_filename> [name <destination\_filename>].

Параметры:

- **crt** — копирование сертификата;
- **key** — копирование ключа;
- <source-filename> — имя файла в директории **transfer**;
- **name** <destination-filename> — имя сохраняемого файла (если отсутствует, то будет использовано имя **cp.crt** для сертификата и **cp.key** для ключа).

Пример копирования сертификата с именем:

```
(cert-config-tls-use)$$ traffic crt cert1.crt name site1.crt
Successfully wrote to /etc/angie-lb/crt/site1.crt
(cert-config-tls-use)$$
```

При копировании ключа и сертификата проверяется их корректность. В случае некорректного значения сертификат или ключ не будет сохранен.

## 7.4.4 configuration

Режим настройки Angie ADC. В этом режиме можно изменять текущую конфигурацию Angie ADC и сохранять изменения в конфигурации-кандидате (команда **set**). Затем конфигурацию-кандидат можно просмотреть (команда **show**), отредактировать (команды **set**, **unset**) и применить (команда **commit**) или откатить (команда **rollback**).

Пример:

```
$$ configuration
set system interface enp0s2 type dhcp
show
-> Current configuration
set system interface enp0s2 type static
set system interface enp0s2 ip address 10.21.20.31/22
set system interface enp0s2 ip gateway 10.21.20.1
set system interface enp0s2 ip dns 8.8.8.8
set system interface enp0s2 ip dns 1.1.1.1

-> Candidate configuration
set system interface enp0s2 type dhcp
##
```

Поддерживаемые команды:

|                      |                                                                                                |
|----------------------|------------------------------------------------------------------------------------------------|
| <b>configuration</b> | Вход в режим настройки<br>Добавлено в версии 0.6.0.                                            |
| <i>commit</i>        | Применение конфигурации-кандидата вместо текущей конфигурации.<br>Добавлено в версии 0.6.0.    |
| <i>end</i>           | Выход из режима настройки <b>configuration</b> .<br>Добавлено в версии 0.6.0.                  |
| <i>rollback</i>      | Откат конфигурации-кандидата.<br>Добавлено в версии 0.6.0.                                     |
| <i>set</i>           | Изменение текущей конфигурации (создание конфигурации-кандидата).<br>Добавлено в версии 0.6.0. |
| <i>show</i>          | Просмотр текущей конфигурации и конфигурации-кандидата.<br>Добавлено в версии 0.6.0.           |
| <i>unset</i>         | Удаление изменения из конфигурации-кандидата.<br>Добавлено в версии 0.6.0.                     |

## **commit**

Применение конфигурации-кандидата в качестве активной конфигурации.

## **end**

Выход из режима **configuration**.

## **rollback**

Удаляет конфигурацию-кандидат.

## **set**

Переход к изменению текущей конфигурации Angie ADC (созданию конфигурации-кандидата).

Синтаксис: **set system <system\_input> | ip <ip\_input>**, где:

- **system <system\_input>** — изменение параметров конфигурации;
- **ip <ip\_input>** — настройка маршрутов и правил IP-трафика.

Возможные варианты **<system\_input>**:

- **set system hostname <name>** — установка имени хоста для Angie ADC.
- **set system interface <name> ip <keyword> <value> [type dhcp | static]** — настройка интерфейса, где **keyword** — настраиваемый параметр, **value** — его значение, **type** — тип настройки интерфейса.

Значения для **keyword**:

- **address** — установка IP-адреса для интерфейса.
- **gateway** — установка шлюза для интерфейса.
- **dns** — установка DNS-адреса для интерфейса.
- **searches** — установка доменов поиска для интерфейса (до 10 доменов).

Значения для **type**:

- **static** — указывает, что IP-адрес будет назначен статически (используется по умолчанию, если тип не указан явно).
- **dhcp** — удаляет все IP-параметры интерфейса и включает режим автоматического получения настроек с DHCP-сервера.
- **set system sysctl <variable> <value>** — настройка параметров ядра Linux (**variable** — настраиваемый параметр, **value** — его значение).

Пример:

```
$$ configuration
set system hostname adc-home
set system interface enp0s2 ip address 10.21.20.39/22
set system interface enp0s2 ip gateway 10.21.20.1
set system interface enp0s2 ip dns 8.8.8.8
set system interface enp0s2 ip searches angie.software.ru
set system sysctl net.ipv4.ip_forward 1
set system sysctl net.core.somaxconn 1024
```

Возможные варианты **<ip\_input>**:

- **rule from <address> table <value>** — установка правила.
- **route <address> {nexthop <address> | interface <name>} [table <value>] [type local]** — установка маршрута. Минимально необходимо задать адрес и **nexthop** или **interface**, остальные параметры (таблица маршрутизации и тип маршрута) опциональны.

Пример:

```
$$ configuration
set ip rule from 0.0.0.0/0 table 100
set ip route 0.0.0.0/0 nexthop 10.90.90.90
```

## show

Просмотр текущей конфигурации и конфигурации-кандидата.

## unset

Удаляет параметр из конфигурации-кандидата.

Пример:

```
$$ configuration
unset ip rule from 0.0.0.0/0 table 100
show
-> Current configuration
set system interface enp0s2 type static
set system interface enp0s2 ip address 10.21.20.31/22
set ip rule from 0.0.0.0/0 table 100

-> Candidate configuration
set system interface enp0s2 type static
set system interface enp0s2 ip address 10.21.20.31/22
```

## 7.4.5 diagnostics

Контекст **diagnostics** позволяет проводить диагностику сети и анализировать сетевой трафик.

Поддерживаемые команды:

|                    |                                                                                                              |
|--------------------|--------------------------------------------------------------------------------------------------------------|
| <b>diagnostics</b> | Вход в контекст                                                                                              |
| <i>arping</i>      | Отправка ARP-запросов сетевым хостам.<br>Добавлено в версии 0.6.0.                                           |
| <i>diag</i>        | Общая диагностика системы.<br>Добавлено в версии 0.6.0.                                                      |
| <i>nslookup</i>    | Выполнение DNS-запросов.<br>Добавлено в версии 0.6.0.                                                        |
| <i>ping</i>        | Выполнение команды <b>ping</b> .<br>Добавлено в версии 0.4.0.                                                |
| <i>pingc</i>       | Выполнение команды <b>ping</b> в стиле Cisco.<br>Добавлено в версии 0.4.0.                                   |
| <i>ss</i>          | Просмотр информации о сетевых соединениях, портах и статистике сетевых сокетов.<br>Добавлено в версии 0.6.0. |
| <i>ssldump</i>     | Анализ SSL/TLS-трафика.<br>Добавлено в версии 0.6.0.                                                         |
| <i>tcpdump</i>     | Анализ сетевого трафика.<br>Добавлено в версии 0.6.0.                                                        |
| <i>traceroute</i>  | Трассировка маршрута следования пакетов.<br>Добавлено в версии 0.6.0.                                        |

### arping

Отправка ARP-запросов сетевым хостам.

Синтаксис: **arping** [-c <count>] [-w <deadline>] [-t <interval>] [-if <interface>] [-src <source>] [-d] [-u] [-a] [-f] [-b] [-q] <target>

Параметры:

- -c <count> — количество ARP-запросов для отправки перед завершением команды;
- -w <deadline> — максимальное время ожидания ответа в секундах;
- -t <interval> — интервал между отправкой запросов в секундах;
- -if <interface> — сетевой интерфейс, через который отправлять запросы;
- -src <source> — исходный IP-адрес для использования в запросах;
- -d — режим обнаружения дубликатов адресов;
- -u — режим незапрошенного ARP: отправка ARP-пакетов без ожидания ответа (для обновления ARP-кэша);
- -a — режим ответа ARP: команда будет отвечать на входящие ARP-запросы;
- -f — завершить выполнение при получении первого ответа;
- -b — широковещательный режим: отправка запросов на широковещательный адрес;
- -q — тихий режим: минимальный вывод информации;
- <target> — IP-адрес или имя хоста, к которому отправляются ARP-запросы (обязательный параметр).

Примеры:

- (diagnostics)\$ arping 192.168.1.1 — проверка доступности хоста.

- (diagnostics)\$\$ `arping -c 5 -w 10 192.168.1.100` — отправка до 5 ARP-запросов и ожидание ответа не более 10 секунд.
- (diagnostics)\$\$ `arping -d 192.168.1.50` — проверка на дублирование адреса.
- (diagnostics)\$\$ `arping -u -if eth0 192.168.1.1` — незапрошенный ARP для обновления кэша.

## diag

Выполнение общей диагностики системы. Информация о состоянии компонентов системы сохраняется в файл `diag_<timestamp>.log`, например в папке `/var/transfer`.

Синтаксис: `diag`

## nslookup

Выполнение DNS-запросов.

### Разовый запрос

Синтаксис: `nslookup <host>`

Примеры:

- (diagnostics)\$\$ `nslookup example.com` — прямой DNS-запрос.
- (diagnostics)\$\$ `nslookup 93.184.216.34` — обратный DNS-запрос (PTR).

## Интерактивный режим

В интерактивном режиме можно выполнять множественные DNS-запросы.

Переход в интерактивный режим: (diagnostics)\$\$ `nslookup`

Поддерживаемые команды интерактивного режима:

- `host [<server>]` — поиск информации о хосте с использованием текущего сервера по умолчанию или указанного сервера.
- `server <domain>` — изменение сервера по умолчанию. Для поиска информации о домене используется текущий сервер.
- `lserver <domain>` — изменение сервера по умолчанию. Для поиска информации о домене используется начальный сервер (тот, с которым запущен `nslookup`).
- `set keyword[=value]` — изменение параметров конфигурации, где `keyword` — имя параметра, `value` — значение (если поддерживается).

Возможные варианты:

- `set all` — вывод всех текущих настроек.
- `set class=value` — изменение класса запроса.

Доступные значения:

- \* `IN` — класс Internet (по умолчанию);
- \* `CH` — класс Chaos;
- \* `HS` — класс Hesiod;
- \* `ANY` — подстановочный знак.

- **set [no]debug** — включение или выключение отображения полного пакета ответа. По умолчанию **nodebug**.
- **set [no]d2** — включение или выключение режима отладки. По умолчанию **nod2**.
- **set domain=<name>** — установка списка поиска доменов по умолчанию.
- **set [no]search** — если запрос содержит хотя бы одну точку, но не заканчивается точкой, добавляет имена доменов из списка поиска доменов к запросу до получения ответа. По умолчанию **search**.
- **set port=<value>** — изменение порта TCP/UDP-сервера имен (по умолчанию 53).
- **set querytype=<value>** — изменение типа запроса информации. По умолчанию **A**, затем **AAAA**.
- **set type=<value>** — изменение типа запроса информации. По умолчанию **A**, затем **AAAA**.
- **set [no]recurse** — включение или выключение рекурсивных запросов. По умолчанию **recurse**.
- **set ndots=<number>** — установка количества точек (разделителей меток) в домене, при котором поиск отключается. Абсолютные имена всегда останавливают поиск.
- **set retry=<number>** — установка количества повторных попыток.
- **set timeout=<number>** — изменение начального интервала ожидания ответа (в секундах).
- **set [no]vc** — указывает, должен ли использоваться виртуальный канал при отправке запросов к серверу. По умолчанию **novc**.
- **set [no]fail** — использовать ли следующий сервер имен, при ошибке на первом сервере. По умолчанию **nofail**.

## ping

Выполнение команды **ping**.

Синтаксис: **ping [-c <count>] [-n] <hostname>**

Параметры:

- **-c <count>** — количество пакетов (по умолчанию бесконечно);
- **-n** — не резолвить имя хоста;
- **<hostname>** — имя хоста или IP-адрес.

## pingc

Выполнение команды **ping** в стиле Cisco.

Синтаксис: **pingc {ip|ip6} <hostname> [count <n>]**

Параметры:

- **ip** — выполнение пинга через ipv4;
- **ip6** — выполнение пинга через ipv6;
- **<hostname>** — имя хоста или IP-адрес;
- **count <n>** — количество пакетов (по умолчанию 4).

## ss

Позволяет получать детальную информацию о TCP/UDP-соединениях, открытых портах, процессах, использующих сеть, и статистике сетевых интерфейсов.

Синтаксис: `ss [-t] [-u] [-l] [-p] [-a] [-n] [-r] [-s] [-e] [-i] [-4] [-6] [-o] [-m] [-h] [<expression>]`

### Параметры

- `-t` — отображение только TCP-соединений.
- `-u` — отображение только UDP-соединений.
- `-l` — отображение только прослушивающих сокетов.
- `-p` — отображение процесса (PID и имя программы), использующего сокет.
- `-a` — отображение всех сокетов: как установленных соединений, так и прослушивающих.
- `-n` — выводить только IP-адреса и номера портов.
- `-r` — разрешать IP-адреса и порты в имена (обратное `-n`).
- `-s` — отображение краткой статистики по протоколам (количество соединений, пакетов и т.д.).
- `-e` — расширенный вывод статистики с дополнительной информацией о соединениях.
- `-i` — отображение информации о сетевых интерфейсах.
- `-4` — отображение только IPv4-соединений.
- `-6` — отображение только IPv6-соединений.
- `-o` — отображение таймеров сокетов в выводе.
- `-m` — отображение использования памяти сокетами.
- `-h` — не отображать заголовки таблицы.
- `<expression>` — фильтрация результатов по `state` и по заданным критериям, оформляется кавычками `"`.

### Примеры

Примеры:

- `(diagnostics)$$ ss -tulpn` — все TCP/UDP-соединения с процессами и портами.
- `(diagnostics)$$ ss -tlnp` — TCP-соединения в состоянии LISTEN.
- `(diagnostics)$$ ss "state listening"` — только прослушивающие соединения.
- `(diagnostics)$$ ss "dport = 80"` — соединения на порт 80.
- `(diagnostics)$$ ss "sport >= 1024"` — соединения с портом источника `>= 1024`.
- `(diagnostics)$$ ss "dst 192.168.1.1"` — соединения к хосту 192.168.1.1.
- `(diagnostics)$$ ss "src 10.0.0.0/8"` — соединения из сети 10.0.0.0/8.
- `(diagnostics)$$ ss "state established dport = 443"` — соединения, установленные на порт 443.
- `(diagnostics)$$ ss "fwmark = 0x01/0x03"` — соединения с определенной меткой `fwmark`.
- `(diagnostics)$$ ss -s` — статистика по протоколам.



## Фильтрация по state

- `state all` — все состояния.
- `state connected` — все соединенные состояния.
- `state synchronized` — синхронизированные состояния.
- `state bucket` — состояния ожидания.
- `state big` — все возможные состояния.
- `state established` — установленное соединение.
- `state syn-sent` — отправлен SYN.
- `state syn-recv` — получен SYN.
- `state fin-wait-1` — ожидание первого FIN.
- `state fin-wait-2` — ожидание второго FIN.
- `state time-wait` — ожидание закрытия.
- `state closed` — закрытое соединение.
- `state close-wait` — ожидание закрытия.
- `state last-ack` — последний ACK.
- `state listening` — прослушивание.
- `state closing` — закрытие соединения.

## Фильтрация по критериям

Фильтрация сокетов на основе определенных критериев. Выражение состоит из серии предикатов, объединенных логическими операторами.

Логические операторы (в порядке возрастания приоритета):

|                                     |                |
|-------------------------------------|----------------|
| <code>or,  ,   </code>              | логическое ИЛИ |
| <code>and, &amp;, &amp;&amp;</code> | логическое И   |
| <code>not, !</code>                 | логическое НЕ  |

Если между последовательными предикатами нет оператора, предполагается неявный оператор `and`. Подвыражения можно группировать с помощью скобок "(" и ")".

Предикаты:

|                                                        |                                                                                                                                                                                                                          |
|--------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>dst HOST   src HOST   dst=HOST   src=HOST</code> | Проверка соответствия адреса назначения или источника хосту HOST. HOST может быть IP-адресом, именем хоста или сетью в формате CIDR.                                                                                     |
| <code>dport OP PORT   sport OP PORT</code>             | Сравнение порта назначения или источника с PORT. OP (оператор) может быть следующим: <, <=, =, ==, !=, >=, >. Например: <code>dport = 80</code> , <code>sport &gt;= 1024</code> , <code>dport &lt; 8080</code> .         |
| <code>dev DEVICE   dev=DEVICE   dev!=DEVICE</code>     | Совпадение по устройству (интерфейсу), которое использует соединение. DEVICE может быть именем устройства или индексом интерфейса.                                                                                       |
| <code>fwmark MASK   fwmark=MASK   fwmark!=MASK</code>  | Совпадение по значению fwmark для соединения. MASK может быть конкретным значением метки или значением с маской битов. Например: <code>fwmark=0x01</code> , <code>fwmark=0x01/0x03</code> (проверка двух младших битов). |
| <code>cgroup PATH   cgroup=PATH   cgroup!=PATH</code>  | Совпадение, если соединение является частью cgroup по указанному пути.                                                                                                                                                   |
| <code>cautobound</code>                                | Совпадение, если порт или путь исходного адреса был автоматически выделен (а не явно указан).                                                                                                                            |

Операторы сравнения (все эквивалентны):

|                                     |                  |
|-------------------------------------|------------------|
| <code>=, ==, eq</code>              | равно            |
| <code>!=, ne, neq</code>            | не равно         |
| <code>&gt;, gt</code>               | больше           |
| <code>&lt;, lt</code>               | меньше           |
| <code>&gt;=, ge, geq</code>         | больше или равно |
| <code>&lt;=, le, leq</code>         | меньше или равно |
| <code>!, not</code>                 | отрицание        |
| <code> ,   , or</code>              | логическое ИЛИ   |
| <code>&amp;, &amp;&amp;, and</code> | логическое И     |

Если оператор не указан, предполагается оператор `=`.

## ssldump

Анализ SSL/TLS-трафика. Показывает содержимое SSL/TLS-пакетов, включая информацию о рукопожатии (handshake), сертификатах, используемых шифрах и зашифрованные данные. Результаты сохраняются в файлы `ssl_capture_<timestamp>.txt` и `ssl_capture_<timestamp>.pcap`.

Синтаксис: `ssldump [-i <interface>] [-n] [-e] [-h] [<expression>]`

Параметры:

- `-i <interface>` — сетевой интерфейс для захвата SSL/TLS-трафика.
- `-n` — показывать только IP-адреса (не выполнять разрешение доменных имен).
- `-e` — показывать время в читаемом формате вместо Unix timestamp.
- `-h` — показывать полные SSL-заголовки пакетов для детального анализа.
- `<expression>` — фильтр захвата, оформляется кавычками `"`. Синтаксис выражений фильтров см. документацию `pcap-filter`.

Примеры:

- `(diagnostics)$$ ssldump -i eth0` — захват всего SSL-трафика на eth0.
- `(diagnostics)$$ ssldump -i eth0 "host example.com"` — SSL-трафик для/от example.com.
- `(diagnostics)$$ ssldump -i eth0 "port 443"` — SSL-трафик на порт 443.

- (diagnostics)\$\$ ssldump -h -i eth0 "tcp and port 443" — полные заголовки SSL-пакетов.

## tcpdump

Анализ сетевого трафика в реальном времени.

Синтаксис: `tcpdump [-i <interface>] [-c <count>] [-w <file>] [-v] [-vv] [-vvv] [-n] [-nn] [-x] [-e] [<expression>]`

Параметры:

- `-i <interface>` — указание сетевого интерфейса для захвата трафика (по умолчанию используется первый доступный интерфейс). Если интерфейс не указан, то команда будет использовать все доступные интерфейсы.
- `-c <count>` — количество пакетов для захвата перед автоматическим завершением команды.
- `-w <file>` — запись захваченных пакетов в файл в формате pcap.
- `-v` — подробный вывод: отображение дополнительной информации о пакетах.
- `-vv` — очень подробный вывод: расширенная информация о пакетах.
- `-vvv` — максимально подробный вывод: максимальный уровень детализации.
- `-n` — не выполнять разрешение доменных имен (DNS lookup): отображать только IP-адреса.
- `-nn` — не выполнять разрешение доменных имен и портов: отображать только IP-адреса и номера портов.
- `-x` — вывод пакетов в шестнадцатеричном формате для детального анализа содержимого.
- `-e` — отображение информации о канальном уровне (Ethernet-заголовки): MAC-адреса, тип фрейма и т.д.
- `<expression>` — фильтр захвата для ограничения типа перехватываемых пакетов, оформляется кавычками "". Синтаксис выражений фильтров см. документацию pcap-filter.

Примеры:

- (diagnostics)\$\$ tcpdump -i eth0 — захват всего трафика на eth0.
- (diagnostics)\$\$ tcpdump -i eth0 "host 192.168.1.1" — трафик только для/от хоста 192.168.1.1.
- (diagnostics)\$\$ tcpdump -i eth0 port 80 — трафик на порт 80 (HTTP).
- (diagnostics)\$\$ tcpdump -i eth0 "tcp and port 443" — TCP-трафик на порт 443 (HTTPS).
- (diagnostics)\$\$ tcpdump -w capture.pcap -c 100 — записать 100 пакетов в файл.

## traceroute

Трассировка маршрута следования пакетов до указанного сетевого узла.

Синтаксис: `traceroute [-i] [-t] [-u] [-4] [-6] [-n] [-d] [-m <max_hops>] [-q <nqueries>] [-p <port>] [-w <waittime>] [-if <interface>] [-s <source>] <target>`

Параметры:

- `-i` — ICMP-режим для трассировки (режим по умолчанию).
- `-t` — TCP-режим: отправка TCP-пакетов вместо ICMP.
- `-u` — UDP-режим: отправка UDP-пакетов.
- `-4` — принудительное использование IPv4.

- **-6** — принудительное использование IPv6.
- **-n** — не выполнять разрешение доменных имен: показывать только IP-адреса.
- **-d** — включение отладочной информации в вывод.
- **-m <max\_hops>** — максимальное количество промежуточных узлов (hops) для проверки (по умолчанию 30).
- **-q <nqueries>** — количество запросов, отправляемых на каждый промежуточный узел (по умолчанию 3).
- **-p <port>** — указание порта для использования в TCP/UDP-режимах.
- **-w <waittime>** — время ожидания ответа от каждого узла в секундах (по умолчанию 3).
- **-if <interface>** — указание сетевого интерфейса для отправки пакетов.
- **-s <source>** — указание исходного IP-адреса для отправки пакетов.
- **<target>** — целевой IP-адрес или доменное имя узла, до которого выполняется трассировка (обязательный параметр).

Примеры:

- (diagnostics)\$ **traceroute -n -w 2 192.168.1.1** — трассировка без DNS, время ожидания 2 секунды.
- (diagnostics)\$ **traceroute -t -p 80 example.com** — TCP-трассировка на порт 80.

## 7.4.6 firewall

Контекст **firewall** позволяет управлять правилами брандмауэра Angie ADC.

Поддерживаемые команды:

|                 |                                                                        |
|-----------------|------------------------------------------------------------------------|
| <b>firewall</b> | Вход в контекст                                                        |
| <i>list</i>     | Вывод списка правил брандмауэра Angie ADC<br>Добавлено в версии 0.4.0. |
| <i>no open</i>  | Закрытие порта в брандмауэре Angie ADC<br>Добавлено в версии 0.4.0.    |
| <i>open</i>     | Открытие порта в брандмауэре Angie ADC<br>Добавлено в версии 0.4.0.    |
| <i>save</i>     | Сохранение правил брандмауэра Angie ADC<br>Добавлено в версии 0.4.0.   |

### list

Вывод списка правил брандмауэра Angie ADC.

Синтаксис: **list {ip|ip6}**.

Параметры:

- **ip** — вывод списка правил для IPv4;
- **ip6** — вывод списка правил для IPv6.

Пример вывода:

```
(firewall)$ list ip
tcp 9999 enp0s2
(firewall)$
```

## no open

Заккрытие порта в брандмауэре Angie ADC.

Синтаксис: `no open {ip|ip6} {tcp|udp} <port> <interface>`.

Параметры:

- `ip` — порт для IPv4;
- `ip6` — порт для IPv6;
- `tcp` — порт для протокола TCP;
- `udp` — порт для протокола UDP;
- `<port>` — номер порта;
- `<interface>` — интерфейс.

Пример:

```
(firewall)$$ no open ip tcp 9999 enp0s2
Port 9999/tcp successfully closed on interface enp0s2
(firewall)$$
```

## open

Открытие порта в брандмауэре Angie ADC.

Синтаксис: `open {ip|ip6} {tcp|udp} <port> <interface>`.

Параметры:

- `ip` — порт для IPv4;
- `ip6` — порт для IPv6;
- `tcp` — порт для протокола TCP;
- `udp` — порт для протокола UDP;
- `<port>` — номер порта;
- `<interface>` — интерфейс.

Пример:

```
(firewall)$$ open ip tcp 9999 enp0s2
Port 9999/tcp successfully opened on interface enp0s2
(firewall)$$
```

### Примечание

При настройке конфигурации не занимайте следующие порты (эти порты используются внутренними сервисами Angie ADC):

TCP: 22, 25, 53, 2022, 2222, 2601, 2602, 2603, 2604, 2605, 2606, 2615, 2616, 2617, 2619, 2623, 3000, 3050, 3051, 3053, 3081, 3101, 3111, 5432, 8080, 9090, 9898

UDP: 53, 323, 3784, 3785, 4784

## save

Сохранение правил брандмауэра Angie ADC.

Синтаксис: `save`.

## 7.4.7 ip

Стандартное управление сетевыми интерфейсами, маршрутами, адресами и правилами IP-трафика в Linux.

Добавлено в версии 0.4.0.

## 7.4.8 logs

Контекст `logs` позволяет просматривать журналы событий Angie ADC. Используйте эти команды для получения диагностической информации при обращении в службу технической поддержки.

|                    |                                                                                                                                      |
|--------------------|--------------------------------------------------------------------------------------------------------------------------------------|
| <i>adc-gslb</i>    | События модуля GSLB (глобальная балансировка)<br>Добавлено в версии 0.4.0.                                                           |
| <i>adc-server</i>  | События модуля, обеспечивающего выполнение всего предоставляемого функционала Angie ADC (входная точка)<br>Добавлено в версии 0.4.0. |
| <i>adc-system</i>  | События модуля, отвечающего за выполнение системных команд Angie ADC<br>Добавлено в версии 0.4.0.                                    |
| <i>adc-tracker</i> | События модуля, отвечающего за RHI-функционал<br>Добавлено в версии 0.4.0.                                                           |
| <i>kern</i>        | События ядра<br>Добавлено в версии 0.4.0.                                                                                            |
| <i>lb</i>          | События модуля балансировщика нагрузки<br>Добавлено в версии 0.4.0.                                                                  |
| <b>logs</b>        | Вход в контекст                                                                                                                      |
| <i>mgmt</i>        | События модуля, отвечающего за управление Angie ADC<br>Добавлено в версии 0.4.0.                                                     |
| <i>upgrade</i>     | События обновления Angie ADC<br>Добавлено в версии 0.4.0.                                                                            |

## adc-gslb

События модуля GSLB (глобальная балансировка).

Синтаксис: `adc-gslb [{page|transfer}]`.

Параметры:

- **page** — вывод записей в режиме постраничного просмотра;
- **transfer** — копирование записей в директорию **transfer**.

## adc-server

События модуля, обеспечивающего выполнение всего предоставляемого функционала Angie ADC (входная точка).

Синтаксис: `adc-server [{page|transfer}]`.

Параметры:

- **page** — вывод записей в режиме постраничного просмотра;
- **transfer** — копирование записей в директорию **transfer**.

## adc-system

События модуля, отвечающего за выполнение системных команд Angie ADC.

Синтаксис: `adc-system [{page|transfer}]`.

Параметры:

- **page** — вывод записей в режиме постраничного просмотра;
- **transfer** — копирование записей в директорию **transfer**.

## adc-tracker

События модуля, отвечающего за RHI-функционал.

Синтаксис: `adc-tracker [{page|transfer}]`.

Параметры:

- **page** — вывод записей в режиме постраничного просмотра;
- **transfer** — копирование записей в директорию **transfer**.

## kern

События ядра.

Синтаксис: `kern [{follow|page|transfer}]`.

Параметры:

- **follow** — вывод записей в режиме реального времени;
- **page** — вывод записей в режиме постраничного просмотра;
- **transfer** — копирование записей в директорию **transfer**.

## lb

События модуля балансировщика нагрузки.

Синтаксис: `lb {access|error} [{follow|page|transfer}]`.

Параметры:

- **access** — вывод записей журнала запросов;
- **error** — вывод записей журнала ошибок;
- **follow** — вывод записей в режиме реального времени;
- **page** — вывод записей в режиме постраничного просмотра;

- **transfer** — копирование записей в директорию **transfer**.

## mgmt

События модуля, отвечающего за управление Angie ADC.

Синтаксис: **mgmt {access|error} [{follow|page|transfer}]**.

Параметры:

- **access** — вывод записей журнала запросов;
- **error** — вывод записей журнала ошибок;
- **follow** — вывод записей в режиме реального времени;
- **page** — вывод записей в режиме страничного просмотра;
- **transfer** — копирование записей в директорию **transfer**.

## upgrade

События обновления Angie ADC.

Синтаксис: **upgrade [{follow|page|transfer}]**.

Параметры:

- **follow** — вывод записей в режиме реального времени;
- **page** — вывод записей в режиме страничного просмотра;
- **transfer** — копирование записей в директорию **transfer**.

## Фильтрация вывода

Вывод записей поддерживает фильтрацию через pipe. Допускается использование нескольких фильтров в виде:

```
command | filter1 | filter2 | ... | filterN
```

Поддерживаются следующие фильтры:

- **grep** — фильтрация по подстроке;
- **exclude** — исключение строк, содержащих указанную подстроку;
- **include** — вывод только строк, содержащих указанную подстроку;
- **head** — вывод первых N строк;
- **tail** — вывод последних N строк.

## grep

Синтаксис: **grep -i -o -v <pattern>**

Параметры:

- **-i** — игнорировать регистр;
- **-o** — показать только совпавшую часть вместо всей строки;
- **-v** — инвертировать результат;
- **<pattern>** — подстрока для поиска.



Пример:

```
(logs)$$ lb access | grep GET
127.0.0.1 - - [29/Aug/2025:14:34:03 +0300] "GET /metrics HTTP/1.1" 200 1800 "-"
↪ "Prometheus/" "-"
(logs)$$ lb access | grep -v GET
2025/08/29 10:19:42 [notice] 976#976: signal 29 (SIGIO) received
(logs)$$ lb access | grep -i get
127.0.0.1 - - [29/Aug/2025:14:34:03 +0300] "GET /metrics HTTP/1.1" 200 1800 "-"
↪ "Prometheus/" "-"
(logs)$$ lb access | grep -o GET
GET
```

## include

Синоним `grep <pattern>`.

Синтаксис: `include <pattern>`.

Параметры:

- `<pattern>` — подстрока для поиска.

Пример:

```
(logs)$$ lb access | include GET
127.0.0.1 - - [29/Aug/2025:14:34:03 +0300] "GET /metrics HTTP/1.1" 200 1800 "-"
↪ "Prometheus/" "-"
```

## exclude

Синоним `grep -v <pattern>`.

Синтаксис: `exclude <pattern>`.

Параметры:

- `<pattern>` — подстрока для поиска.

Пример:

```
(logs)$$ lb error | exclude exit
2025/08/29 10:19:42 [notice] 976#976: signal 29 (SIGIO) received
```

## head

Вывод первых N строк.

Синтаксис: `head <n>`.

Параметры:

- `<n>` — количество строк.

Пример:

```
(logs)$$ lb access | head 2
127.0.0.1 - - [28/Aug/2025:14:53:48 +0300] "GET /metrics HTTP/1.1" 200 1798 "-"
↪ "Prometheus/" "-"
```

```
127.0.0.1 - - [28/Aug/2025:14:53:58 +0300] "GET /metrics HTTP/1.1" 200 1798 "-"
↪ "Prometheus/" "-"
```

## tail

Вывод последних N строк.

Синтаксис: `tail <n>`.

Параметры:

- `<n>` — количество строк.

Пример:

```
(logs)$$ lb access | tail 2
127.0.0.1 - - [28/Aug/2025:14:53:48 +0300] "GET /metrics HTTP/1.1" 200 1798 "-"
↪ "Prometheus/" "-"
127.0.0.1 - - [28/Aug/2025:14:53:58 +0300] "GET /metrics HTTP/1.1" 200 1798 "-"
↪ "Prometheus/" "-"
```

## 7.4.9 modules

Контекст `modules` позволяет просматривать информацию о динамических модулях Angie ADC.

|                      |                                                                                        |
|----------------------|----------------------------------------------------------------------------------------|
| <i>info</i>          | Получение информации о динамическом модуле Angie ADC<br>Добавлено в версии 0.4.0.      |
| <i>list</i>          | Получение списка доступных динамических модулей Angie ADC<br>Добавлено в версии 0.4.0. |
| <code>modules</code> | Вход в контекст                                                                        |

### info

Получение информации о динамическом модуле Angie ADC.

Синтаксис: `info <module>`.

Параметры:

- `<module>` — имя модуля.

### list

Получение списка доступных динамических модулей Angie ADC.

Синтаксис: `list`.

### 7.4.10 ps

Отображение информации о запущенных процессах в системе. Показывает список активных процессов с различной степенью детализации в зависимости от используемых параметров.

Добавлено в версии 0.6.0.

Синтаксис: `ps [-f] [-p <pid>] [-o <format>] [-c <command>] [-s <sortspec>] [-t] [-w] [-h]`.

Параметры:

- `-f` — полный формат вывода: отображение расширенной информации о процессах.
- `-p <pid>` — отображение информации только о процессе с указанным PID (Process ID).
- `-o <format>` — пользовательский формат вывода. Формат задается как список полей, разделенных запятыми (например: `pid,cmd,mem`). Доступные поля описаны ниже.
- `-c <command>` — поиск и отображение процессов по имени команды (фильтрация по имени процесса).
- `-s <sortspec>` — сортировка вывода по указанным полям (например, по использованию CPU, памяти).
- `-t` — отображение процессов в древовидной структуре: показывает родительно-дочерние связи между процессами.
- `-w` — широкий вывод: позволяет отображать полные команды без обрезания.
- `-h` — не показывать заголовки столбцов в выводе.

Примеры:

- `$$ ps` — список процессов текущего пользователя.
- `$$ ps -p 1234` — информация о процессе с PID 1234.
- `$$ ps -c angie` — поиск процессов `angie`.
- `$$ ps -o pid,ppid,cmd,mem` — пользовательский формат вывода.

Форматы для параметра `-o`:

| Ключ | Полное имя | Описание                                                    |
|------|------------|-------------------------------------------------------------|
| c    | cmd        | Простое имя исполняемого файла                              |
| C    | pcpu       | Использование CPU (в процентах)                             |
| f    | flags      | Флаги как в длинном формате (поле F)                        |
| g    | pgrp       | Идентификатор группы процессов                              |
| G    | tpgid      | Идентификатор группы процессов управляющего терминала       |
| j    | cutime     | Совокупное пользовательское время                           |
| J    | cstime     | Совокупное системное время                                  |
| k    | utime      | Пользовательское время                                      |
| m    | min_ft     | Количество малых ошибок страниц (minor page faults)         |
| M    | maj_ft     | Количество больших ошибок страниц (major page faults)       |
| n    | cmin_ft    | Совокупные малые ошибки страниц                             |
| N    | cmaj_ft    | Совокупные большие ошибки страниц                           |
| o    | session    | Идентификатор сессии                                        |
| p    | pid        | Идентификатор процесса                                      |
| P    | ppid       | Идентификатор родительского процесса                        |
| r    | rss        | Размер резидентного набора (resident set size) в килобайтах |
| R    | resident   | Резидентные страницы                                        |
| s    | size       | Размер памяти в килобайтах                                  |
| S    | share      | Количество разделяемых страниц                              |
| t    | tty        | Номер устройства управляющего терминала                     |
| T    | start_time | Время запуска процесса                                      |
| U    | uid        | Числовой идентификатор пользователя                         |
| u    | user       | Имя пользователя                                            |
| v    | vsize      | Общий размер виртуальной памяти в килобайтах                |
| y    | priority   | Приоритет планировщика ядра                                 |

Примеры:

- `$$ ps -o pid,cmd` — отобразить только PID и команду.
- `$$ ps -o user,pid,pcpu,rmem,cmd` — пользователь, PID, CPU, память, команда.
- `$$ ps -o pid,ppid,cmd,start_time` — PID, PPID, команда, время запуска.

### 7.4.11 settings

Контекст **settings** позволяет просматривать и изменять настройки Angie ADC.

Поддерживаемые команды:

|                 |                                                                                         |
|-----------------|-----------------------------------------------------------------------------------------|
| <i>angie</i>    | Включение режима отладки<br>Добавлено в версии 0.4.0.                                   |
| <i>ntp</i>      | Управление службой NTP и настройками синхронизации времени<br>Добавлено в версии 0.6.0. |
| <i>no angie</i> | Отключение режима отладки<br>Добавлено в версии 0.4.0.                                  |
| <i>no ntp</i>   | Сброс конфигурации NTP к значениям по умолчанию<br>Добавлено в версии 0.6.0.            |
| <i>reload</i>   | Перезагрузка элементов Angie ADC<br>Добавлено в версии 0.4.0.                           |
| <b>settings</b> | Вход в контекст                                                                         |
| <i>sysctl</i>   | Просмотр параметров ядра Linux в реальном времени<br>Добавлено в версии 0.6.0.          |
| <i>time</i>     | Просмотр и установка системного времени<br>Добавлено в версии 0.6.0.                    |
| <i>timezone</i> | Просмотр и установка часового пояса системы<br>Добавлено в версии 0.6.0.                |

## angie

Включение режима отладки.

Синтаксис: **angie debug**.

## ntp

Управление службой NTP (Network Time Protocol) и настройками синхронизации времени с удаленными NTP-серверами.

Синтаксис: **ntp {enable | disable | status | servers | server add <server> | server del <server> | pool add <pool> | pool del <pool> | show-config | makestep}**.

Параметры:

- **enable** — включение службы NTP для автоматической синхронизации времени в фоновом режиме.
- **disable** — отключение службы NTP.
- **status** — отображение текущего статуса службы NTP и информации о синхронизации.
- **servers** — отображение списка всех настроенных NTP-серверов.
- **server add <server>** — добавление нового NTP-сервера в список источников времени. Сервер может быть указан как IP-адрес или доменное имя.
- **server del <server>** — удаление NTP-сервера из списка.
- **pool add <pool>** — добавление NTP-пула серверов. Пул представляет собой группу серверов, из которых автоматически выбирается доступный.
- **pool del <pool>** — удаление NTP-пула из конфигурации.
- **show-config** — отображение полной текущей конфигурации NTP: список серверов, пулов, настройки синхронизации.
- **makestep** — выполнение немедленной синхронизации времени с NTP-серверами.

## no angie

Отключение режима отладки **angie**.

Синтаксис: **no angie debug**.

## no ntp

Сброс конфигурации NTP к значениям по умолчанию.

Синтаксис: **no ntp**.

## reload

Перезагрузка элементов Angie ADC.

Синтаксис: **reload {angie|prometheus}**.

Параметры:

- **angie** — перезагрузка **angie-mgmt**.
- **prometheus** — перезагрузка **prometheus**.

## sysctl

Просмотр конкретного параметра ядра Linux или всех параметров в реальном времени.

Синтаксис: **sysctl show {<variable> | all}**.

## time

Просмотр и установка системного времени.

Синтаксис: **time {show | set <YYYY-MM-DD HH:MM:SS>}**.

Подкоманды:

- **show** — отображение текущего времени системы.
- **set <YYYY-MM-DD HH:MM:SS>** — установка системного времени в указанном формате.

Примеры:

- **(settings)\$ time show** — показать текущее время.
- **(settings)\$ set 2025-01-15 14:30:00** — установить время на 15 января 2025, 14:30:00.

## timezone

Просмотр и установка часового пояса системы. Часовой пояс определяет смещение локального времени относительно UTC. Изменение часового пояса не изменяет системное время (Unix timestamp), а только влияет на его отображение.

Синтаксис: **timezone {show | set <timezone>}**.

Подкоманды:

- **show** — отображение текущего часового пояса системы.
- **set <timezone>** — установка часового пояса. Часовой пояс должен быть указан в формате IANA Time Zone Database (например, **Europe/Moscow**) или в виде аббревиатуры (например, **UTC**, **EST**, **PST**).

Примеры:

- `(settings)$$ timezone show` — показать текущий часовой пояс.
- `(settings)$$ timezone set UTC` — установить часовой пояс UTC.
- `(settings)$$ timezone set Europe/Moscow` — установить московское время.

## 7.4.12 system

Контекст **system** для управления системой Angie ADC.

Поддерживаемые команды:

|                |                                                                    |
|----------------|--------------------------------------------------------------------|
| <i>reboot</i>  | Перезагрузка системы Angie ADC<br>Добавлено в версии 0.4.0.        |
| <i>stat</i>    | Просмотр статистики системы Angie ADC<br>Добавлено в версии 0.4.0. |
| <b>system</b>  | Вход в контекст                                                    |
| <i>upgrade</i> | Обновление Angie ADC<br>Добавлено в версии 0.4.0.                  |
| <i>version</i> | Просмотр версии Angie ADC<br>Добавлено в версии 0.4.0.             |

### reboot

Перезагрузка системы Angie ADC.

Синтаксис: `reboot [force]`

Параметры:

- **force** — принудительная перезагрузка.

### stat

Просмотр статистики системы Angie ADC.

Синтаксис: `stat [count <n>] [delay <m>]`

Параметры:

- **count <n>** — количество обновлений статистики;
- **delay <m>** — задержка между обновлениями статистики в секундах.

### upgrade

Обновление Angie ADC.

Синтаксис: `upgrade <filename>`

Параметры:

- **<filename>** — имя файла.

## version

Просмотр версии Angie ADC.

Синтаксис: `version [detail]`

Параметры:

- `detail` — подробный вывод версии.

## 7.4.13 transfer

Контекст **transfer** для работы с директорией **transfer**.

Поддерживаемые команды:

|                 |                                                                                      |
|-----------------|--------------------------------------------------------------------------------------|
| <i>edit</i>     | Редактирование файла в текущей директории.<br>Добавлено в версии 0.6.0.              |
| <i>ls</i>       | Получение списка файлов в директории <b>transfer</b><br>Добавлено в версии 0.4.0.    |
| <i>rm</i>       | Удаление файла в директории <b>transfer</b><br>Добавлено в версии 0.4.0.             |
| <b>transfer</b> | Вход в контекст                                                                      |
| <i>view</i>     | Просмотр содержимого файла в директории <b>transfer</b><br>Добавлено в версии 0.4.0. |

## edit

Позволяет редактировать файлы в текущей директории.

Синтаксис: `edit [<filename>]`.

Параметры:

- `<filename>` — имя файла, который необходимо отредактировать (обязательный параметр).

## ls

Получение списка файлов в директории **transfer**.

Синтаксис: `ls`.

## rm

Удаление файла в директории **transfer**.

Синтаксис: `rm <filename>`.

Параметры:

- `<filename>` — имя файла.



## **view**

Просмотр содержимого файла в директории **transfer**.

Синтаксис: **view** <filename> [page].

Параметры:

- <filename> — имя файла;
- page — вывод записей в режиме постраничного просмотра.

### **7.4.14 vtysh**

Переход к настройкам маршрутизации по протоколам BGP и OSPF.

Добавлено в версии 0.4.0.

Синтаксис: **vtys**.

## ГЛАВА 8

---

## Права на интеллектуальную собственность

---

Документация на программный продукт Angie ADC является интеллектуальной собственностью ООО «Веб-Сервер».

Copyright © 2025, ООО «Веб-Сервер». Все права защищены.

---

## Алфавитный указатель

---

### A

absolute\_redirect (*http*), 43  
 accept\_mutex (*core*), 32  
 accept\_mutex\_delay (*core*), 32  
 aio (*http*), 43  
 aio\_write (*http*), 45  
 alias (*http*), 45  
 auth\_delay (*http*), 46  
 auto\_redirect (*http*), 46  
 autoindex (*http*), 133  
 autoindex\_exact\_size (*http*), 133  
 autoindex\_format (*http*), 133  
 autoindex\_localtime (*http*), 135

### B

backup\_switch (*http*), 312  
 backup\_switch (*stream*), 419  
 bind\_conn (*http*), 313

### C

chunked\_transfer\_encoding (*http*), 46  
 client (*http*), 46  
 client\_body\_buffer\_size (*http*), 48  
 client\_body\_in\_file\_only (*http*), 48  
 client\_body\_in\_single\_buffer (*http*), 48  
 client\_body\_temp\_path (*http*), 48  
 client\_body\_timeout (*http*), 49  
 client\_header\_buffer\_size (*http*), 49  
 client\_header\_timeout (*http*), 49  
 client\_max\_body\_size (*http*), 50  
 connection\_pool\_size (*http*), 50

### D

daemon (*core*), 32  
 debug\_connection (*core*), 33  
 debug\_points (*core*), 33  
 default\_type (*http*), 50  
 directio (*http*), 50  
 directio\_alignment (*http*), 51  
 disable\_symlinks (*http*), 51

### E

env (*core*), 33

error\_log (*core*), 34  
 error\_page (*http*), 52  
 etag (*http*), 53  
 events (*core*), 35

### F

feedback (*http*), 314  
 feedback (*stream*), 420

### H

hash (*http*), 315  
 hash (*stream*), 421  
 http (*http*), 53

### I

if\_modified\_since (*http*), 54  
 ignore\_invalid\_headers (*http*), 54  
 include (*core*), 35  
 internal (*http*), 54  
 ip\_hash (*http*), 315

### K

keepalive (*http*), 316  
 keepalive\_disable (*http*), 55  
 keepalive\_requests (*http*), 56, 318  
 keepalive\_time (*http*), 56, 318  
 keepalive\_timeout (*http*), 56, 318

### L

large\_client\_header\_buffers (*http*), 56  
 least\_bandwidth (*http*), 318  
 least\_bandwidth (*stream*), 422  
 least\_conn (*http*), 319  
 least\_conn (*stream*), 423  
 least\_packets (*stream*), 423  
 least\_time (*http*), 319  
 least\_time (*stream*), 423  
 limit\_except (*http*), 57  
 limit\_rate (*http*), 57  
 limit\_rate\_after (*http*), 58  
 lingering\_close (*http*), 58  
 lingering\_time (*http*), 59  
 lingering\_timeout (*http*), 59

listen, 449  
listen (*http*), 59  
location (*http*), 63  
lock\_file (*core*), 36  
log\_not\_found (*http*), 65  
log\_subrequest (*http*), 65

## M

mail, 451  
master\_process (*core*), 36  
max\_commands, 451  
max\_errors, 451  
max\_headers (*http*), 66  
max\_ranges (*http*), 66  
merge\_slashes (*http*), 66  
msie\_padding (*http*), 67  
msie\_refresh (*http*), 67  
multi\_accept (*core*), 36

## O

open\_file\_cache (*http*), 67  
open\_file\_cache\_errors (*http*), 68  
open\_file\_cache\_min\_uses (*http*), 68  
open\_file\_cache\_valid (*http*), 68  
output\_buffers (*http*), 68

## P

pcre\_jit (*core*), 36  
pid (*core*), 37  
port\_in\_redirect (*http*), 69  
postpone\_output (*http*), 69  
protocol, 452

## Q

queue (*http*), 320

## R

random (*http*), 321  
random (*stream*), 424  
read\_ahead (*http*), 69  
recursive\_error\_pages (*http*), 69  
request\_pool\_size (*http*), 70  
reset\_timedout\_connection (*http*), 70  
resolver, 452  
resolver (*http*), 70  
resolver\_timeout, 453  
resolver\_timeout (*http*), 71  
response\_time\_factor (*http*), 321  
response\_time\_factor (*stream*), 424  
root (*http*), 71

## S

satisfy (*http*), 72  
send\_lowat (*http*), 72  
send\_timeout (*http*), 72  
sendfile (*http*), 73  
sendfile\_max\_chunk (*http*), 73

server, 453  
server (*http*), 73, 321  
server (*stream*), 416  
server\_name, 453  
server\_name (*http*), 74  
server\_name\_in\_redirect (*http*), 76  
server\_names\_hash\_bucket\_size (*http*), 76  
server\_names\_hash\_max\_size (*http*), 76  
server\_tokens (*http*), 76  
ssl\_engine (*core*), 37  
ssl\_object\_cache\_inheritable (*core*), 37  
state (*http*), 324  
state (*stream*) ((*stream* upstream module), 418  
status\_zone (*http*), 77  
sticky (*http*), 324  
sticky (*stream*), 425  
sticky\_secret (*http*), 330  
sticky\_secret (*stream*), 429  
sticky\_strict (*http*), 330  
sticky\_strict (*stream*), 430  
subrequest\_output\_buffer\_size (*http*), 78

## T

tcp\_nodelay (*http*), 78  
tcp\_nopush (*http*), 78  
thread\_pool (*core*), 38  
timeout, 454  
timer\_resolution (*core*), 38  
try\_files (*http*), 79  
types (*http*), 81  
types\_hash\_bucket\_size (*http*), 82  
types\_hash\_max\_size (*http*), 82

## U

underscores\_in\_headers (*http*), 82  
upstream (*http*), 330  
upstream (*stream*), 415  
upstream\_probe (*http*), 335  
upstream\_probe (*stream*), 432  
upstream\_probe\_timeout (*stream*), 434  
use (*core*), 39  
user (*core*), 39

## V

variables\_hash\_bucket\_size (*http*), 83  
variables\_hash\_max\_size (*http*), 83

## W

worker\_aio\_requests (*core*), 39  
worker\_connections (*core*), 39  
worker\_cpu\_affinity (*core*), 40  
worker\_priority (*core*), 40  
worker\_processes (*core*), 41  
worker\_rlimit\_core (*core*), 41  
worker\_rlimit\_nofile (*core*), 41  
worker\_shutdown\_timeout (*core*), 41  
working\_directory (*core*), 42

## Z

zone (*http*), 331

zone (*stream*), 419