ООО «Веб-Сервер», ОГРН 1227700436578, ИНН 9704151517
ул. Вятская, д. 27, стр. 7, г. Москва, 127015

Руководство по эксплуатации

версия 0.2.1

ООО «Веб-Сервер»

апр. 24, 2025

Оглавление

ГЛАВА 1

Аннотация

Настоящий документ содержит информацию, необходимую для эксплуатации программного обеспечения 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).
- Динамическая 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

В этом разделе собраны примеры настройки конфигурации балансировщика нагрузки Angie ADC под разные задачи.

3.1 Настройка конфигурации

По умолчанию в конфигурационном файле балансировщика нагрузки заданы следующие настройки:

- 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.

Вы можете настроить балансировщик нагрузки под свои цели, отредактировав его конфигурацию. Тип балансировки нагрузки зависит от конкретных задач. Для работы на низком уровне (TCP, UDP) доступна stream-балансировка, для приложений и веб-сервисов — балансировка на уровне приложений (HTTP).

3.1.1 Редактирование конфигурации балансировщика нагрузки

Чтобы отредактировать конфигурацию балансировщика нагрузки, 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. Внесите необходимые изменения. Примеры настройки см. ниже.
5. Сохраните файл, нажав кнопку `Сохранить`.

   Изменения будут применены сразу.

3.1.2 О механизмах балансировки

В Angie ADC поддерживаются статические и динамические механизмы распределения запросов между серверами.

Механизм	Описание	HT˜ TCP/UDP
`round-rc`	Запросы распределяются по серверам последовательно (используется по умолчанию).	
weight	Запросы распределяются по серверам учетом веса каждого сервера.	
ip-hash	Сервер выбирается с помощью хэш-функции на основе IP-адреса клиента, что обеспечивает “липкость сессии”.	
least_co	Новый запрос направляется на сервер с минимальным количеством активных соединений (наименее загруженный).	
least_tin	Вероятность передачи соединения активному серверу обратно пропорциональна среднему времени его ответа.	
feedback	Балансировка нагрузки по обратной связи (по произвольному параметру из ответа на основной или проверочный запрос).	
least_ba	Балансировка на основе наименьшего потребления полосы пропускания.	
least_pa	Балансировка на основе наименьшего числа пакетов в единицу времени.	

3.1.3 Примеры HTTP-балансировки нагрузки

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.

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` Задает 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.

3.1.4 Примеры TCP/UDP-балансировки нагрузки

TCP/UDP-балансировка нагрузки позволяет распределять TCP- или UDP-трафик между несколькими серверами. Это полезно для таких задач, как балансировка баз данных, почтовых серверов или других TCP/UDP-сервисов.

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 сбоев 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 "липких 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.

3.1.5 Проверка работоспособности серверов

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

Пример настройки пассивных проверок

```
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.

Пример настройки активных проверок

```
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\n\n";
}
```

Синтаксис	`upstream_probe` *имя* [`port=`*число*] [`interval=`*время*] [`test=`*условие*] [`essential` [`persistent`]] [`fails=`*число*] [`passes=`*число*] [`max_response=`*размер*] [`mode=always` | `idle` | `onfail`] [`udp`] [`send=`*строка*];
По умолчанию	—
Контекст	server

Задает 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. Для одного апстрима можно определить несколько проверок.

Могут быть заданы следующие параметры:

`имя`	Обязательное имя проверки.
`port`	Альтернативный порт для запроса.
`interval`	Интервал между проверками. По умолчанию — `5s`.
`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 `essential`; серверы с `persistent`, работавшие до перезагрузки конфигурации, 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 значение, ожидание ответа отключается. По умолчанию — `256k`.
`mode`	Режим проверки в зависимости от работоспособности серверов: • `always` — серверы проверяются независимо от состояния; • `idle` — проверяются неработающие серверы, а также серверы, где с последнего клиентского запроса прошло время `interval`. • `onfail` — проверяются серверы только в неработающем состоянии. По умолчанию — `always`.
`udp`	Если параметр указан, используется протокол UDP. По умолчанию для проверок используется TCP.
`send`	Отправляемые для проверки данные; это может быть строка с префиксом `data:` или имя файла с данными (задается абсолютно или относительно каталога `/usr/local/angie/`).

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 `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`.

3.2 Резервное копирование и восстановление

В Angie ADC реализовано резервное копирование и хранение всех версий конфигурационного файла балансировщика нагрузки. При сохранении изменений в текущем файле Angie ADC автоматически применяет новую конфигурацию и сохраняет предыдущую версию на сервере.

Вы можете просмотреть список сохраненных конфигурационных файлов и применить нужную версию.

3.2.1 Просмотр списка резервных копий конфигурации

Чтобы просмотреть список резервных копий конфигурации балансировщика нагрузки:

1. 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), комментарий с причиной поломки, если есть, и дата последнего изменения.

3.2.2 Присвоение нерабочего статуса

Чтобы пометить резервную копию конфигурации как нерабочую:

1. Выберите `Панель мониторинга` → `<Имя_балансировщика_нагрузки>` → `Конфигурация` → `История`.

   Откроется таблица со списком сохраненных конфигурационных файлов.

2. Щелкните троеточие (`...`) напротив конфигурации, которую вы хотите пометить как нерабочую, и нажмите `Пометить как нерабочую`.

3. Укажите причину поломки и нажмите `Сохранить`.

   Статус конфигурации будет изменен.

3.2.3 Восстановление последней рабочей версии конфигурации

Чтобы откатиться к последней рабочей версии конфигурации:

1. Выберите `Панель мониторинга` → `<Имя_балансировщика_нагрузки>` → `Конфигурация` → `История`.

2. В верхней части экрана нажмите кнопку `Откатить к последней рабочей версии`.

   Будет применена последняя рабочая конфигурация.

3.2.4 Восстановление конфигурации из резервной копии

Чтобы применить произвольную конфигурацию из резервной копии:

1. Выберите `Панель мониторинга` → `<Имя_балансировщика_нагрузки>` → `Конфигурация` → `История`.

   Откроется таблица со списком сохраненных конфигурационных файлов.

2. Щелкните троеточие (`...`) напротив конфигурации, которую вы хотите использовать, и нажмите кнопку `Применить`.

   Конфигурация будет применена сразу.

3.3 Загрузка и раздача статических файлов

Вы можете загружать и раздавать статические файлы в Angie ADC с помощью HTTP-запросов через REST API. REST API предоставляет конечную точку (`POST /file/static`), через которую можно загружать файлы. При загрузке файлов через REST API их путь указывается относительно статической директории. Это позволяет автоматизировать процесс и легко управлять статическими ресурсами, которые должен раздавать Angie ADC.

Поддерживается:

- загрузка нескольких файлов за один запрос;
- возможность замены существующих файлов;
- текстовые и двоичные данные (обычный текст или кодирование Base64).

3.3.1 Загрузка статических файлов

Для загрузки статических файлов используйте следующий запрос:

```
POST /file/static HTTP/1.1
Authorization: Bearer <token>

{
  "items": [
    {
        "filepath": "path/to/file",  # расположение файла относительно статической␣
↪директории (static dir), которая будет определена позднее
        "content": {
            "plain-text": "zone \"test.example.com.\" {\n    type master\n    file \"/
↪etc/bind/master-zones/test.example.com.local.zone\"\n}\n"
        },
        "rewrite": true
    },
    {
        "filepath": "path/to/file",  # расположение файла относительно статической␣
↪директории (static dir), которая будет определена позднее
        "content": {
            "plain-text": "zone \"test.example.com.\" {\n    type master\n    file \"/
↪etc/bind/master-zones/test.example.com.local.zone\"\n}\n"
        },
        "rewrite": false
    }
  ]
}
```

После успешной обработки файлы загрузятся в `static dir` в указанное относительное расположение (`filepath`). API вернет абсолютные пути к загруженным файлам в `HTTP Response Body`:

```
{
    "items": [
        "/path/to/file/one",
        "/path/to/file/two",
        "/path/to/file/three"
    ]
}
```

> **Важно**
>
> Все файлы загружаются вместе: если с одним из файлов возникнет ошибка, загрузка всех файлов будет отменена.

Возможные ошибки обработки:

- некорректный запрос: Код состояния HTTP 400;
- ошибка записи файла: Код состояния HTTP 500;
- если в `static dir/filepath` уже есть соответствующий файл, а параметр `rewrite` установлен в `false`, загрузка завершится ошибкой: Код состояния HTTP 400.

3.3.2 Раздача статических файлов через балансировщик нагрузки

Чтобы раздавать загруженные файлы через балансировщик нагрузки, необходимо задать `location` в конфигурационном файле балансировщика нагрузки.

1. Откройте в браузере консоль Angie ADC.
2. Перейдите в `Панель мониторинга` → `<Имя балансировщика>` → `Конфигурация`.

   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.
3. Внесите изменения с учетом загруженных статических файлов, например:

   ```
   location ~ \.(mp3|mp4) {
       root /static/dir/file;
   }
   ```
4. Сохраните конфигурацию, нажав кнопку `Сохранить`.

   Изменения будут применены сразу.

3.3.3 Загрузка и раздача двоичных данных

Двоичные данные передаются в объекте `content.plain-text` без преобразований. Также можно использовать кодирование Base64. Значение `base64` должно содержать закодированные данные файла. Файлы, загруженные таким способом, также можно раздавать через балансировщик нагрузки.

Пример загрузки файла в `base64`:

```
POST /file/static HTTP/1.1
Authorization: Bearer <token>

{
  "items": [
    {
        "filepath": "path/to/file",  # расположение файла относительно статической␣
↪директории (static dir), которая будет определена позднее
        "content": {
            "base64":
↪"em9uZSBcInRlc3QuZXhhbXBsZS5jb20uXCIge1xuICAgIHR5cGUgbWFzdGVyXG4gICAgZmlsZSBcIi9ldGMvYmluZC9tYXN0ZX
↪"
        },
        "rewrite": true
```

```
    }
  ]
}
```

3.4 Справочная информация

Справочные материалы по настройке Angie ADC.

HTTP upstream

Stream upstream

3.4.1 HTTP upstream

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

bind_conn

Синтаксис	`bind_conn` *значение*;
По умолчанию	—
Контекст	upstream

Позволяет привязать серверное соединение к клиентскому в момент, когда *значение*, заданное строкой из переменных, становится отличным от `""` и `"0"`.

Внимание

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`.

Внимание

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

Синтаксис	`feedback` *переменная* [`inverse`] [`factor=`*число*] [`account=`*условная_ переменная*] [`last_byte`];
По умолчанию	—
Контекст	upstream

Задает 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 сервер передает ее в заголовках или иным образом. Значение оценивается при каждом ответе от сервера и учитывается в скользящем среднем согласно настройкам `inverse` и `factor`.
`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`. Среднее рассчитывается по формуле экспоненциального сглаживания. 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, какие ответы учитываются при расчете. Среднее значение обновляется с учетом значения обратной связи из ответа, только если условная переменная этого ответа не равна `""` или `"0"`. **Примечание** По умолчанию ответы в ходе активных проверок не включаются в расчет; комбинация переменной `$upstream_probe` с `account` позволяет включить эти ответы или даже исключить все остальное.
`last_byte`	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

Синтаксис	`hash` *ключ* [`consistent`];
По умолчанию	—
Контекст	upstream

Задает 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.

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 равным 160.

ip_hash

Синтаксис	`ip_hash`;
По умолчанию	—
Контекст	upstream

Задает 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

Синтаксис	`keepalive` *соединения*;
По умолчанию	—
Контекст	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.

Внимание

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

Синтаксис	`keepalive_requests` *число*;
По умолчанию	`keepalive_requests 1000;`
Контекст	upstream

Задает 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

Синтаксис	`keepalive_time` *время*;
По умолчанию	`keepalive_time 1h;`
Контекст	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

Синтаксис	`keepalive_timeout` *время*;
По умолчанию	`keepalive_timeout 60s;`
Контекст	upstream

Задает таймаут, в течение которого неактивное постоянное соединение с сервером группы не будет закрыто.

least_conn

Синтаксис	`least_conn`;
По умолчанию	—
Контекст	upstream

Задает 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

Задает 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 *response_time_factor*, и переопределяет его, если параметр задан.
`account`	Указывает условную переменную, которая контролирует, какие ответы учитываются при расчете. Среднее значение обновляется, только если условная переменная ответа не равна `""` или `"0"`. **Примечание** По умолчанию ответы в ходе активных проверок не включаются в расчет; 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 удается 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 задает максимальное количество запросов в очереди `рабочего процесса`. Если очередь целиком заполнена, клиенту отдается 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 отдается ошибка `502 (Bad Gateway)`. Еще из очереди удаляются запросы от клиентов, преждевременно закрывших соединение; в API есть счетчики состояний запросов, проходящих через очередь.

Внимание

Директива `queue` должна использоваться после всех директив, задающих тот или иной метод балансировки нагрузки, иначе она не будет работать.

random

Синтаксис	`random` [`two`];
По умолчанию	—
Контекст	upstream

Задает 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

Синтаксис	`response_time_factor` *число*;
По умолчанию	`response_time_factor 90;`
Контекст	upstream

Задает 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.

> **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

Синтаксис	`server` *адрес* [*параметры*];
По умолчанию	—
Контекст	upstream

Задает 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, задает сразу несколько серверов.

Могут быть заданы следующие параметры:

`weight=`*число*	задает вес сервера по умолчанию 1.
`max_conns=`*число*	ограничивает максимальное число одновременных активных соединений к проксируемому серверу. Значение по умолчанию равно `0` и означает, что ограничения нет. Если группа не находится в *зоне* разделяемой памяти, то ограничение работает отдельно для каждого рабочего процесса.

Примечание

При включенных *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* — задает число неудачных попыток связи с сервером, которые должны произойти в течение заданного `fail_timeout <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 признается 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.

`max_fails=1`	число попыток по умолчанию
`max_fails=0`	отключает учет попыток

`fail_timeout=`*время* — задает период времени, в течение которого должно произойти определенное число неудачных попыток связи с сервером (`max_fails <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.

`backup`	помечает сервер как запасной. На него будут передаваться запросы в случае, если не работают основные серверы.
`down`	помечает сервер как постоянно недоступный.
`drain`	помечает сервер как разгружаемый (draining); это значит, что он получает только запросы сессий, привязанных ранее через *sticky*. В остальном поведение такое же, как в режиме `down`.

Осторожно

Параметр `backup` нельзя использовать совместно с методами балансировки нагрузки *hash*, *ip_hash* и *random*.

Параметры `down` и `drain` взаимно исключающие.

`resolve`	позволяет отслеживать изменения списка IP-адресов, соответствующего доменному имени, и обновлять его без перезагрузки конфигурации. При этом группа должна находиться в *зоне разделяемой памяти*; также должен быть определен `преобразователь имен в адреса`.
`service=`*имя*	включает преобразование SRV-записей DNS и задает имя сервиса. Для работы параметра необходимо задать параметр `resolve` у сервера, не указывая порт сервера при имени хоста. Если в имени службы нет точек, формируется имя по стандарту RFC: к имени службы добавляется префикс `_`, затем через точку добавляется `_tcp`. Так, имя службы `http` даст в результате `_http._tcp`. 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. • SRV-записи с наивысшим приоритетом (те, которые имеют минимальное значение приоритета) разрешаются как основные серверы, а прочие записи становятся запасными серверами. Если `backup` установлено с `server`, SRV-записи с наивысшим приоритетом разрешаются как запасные серверы, а прочие записи игнорируются. • Вес аналогичен параметру `weight` директивы `server`. Если вес задан как в самой директиве, так и в SRV-записи, используется вес, установленный в директиве.

В этом примере выполняется поиск записи `_http._tcp.backend.example.com`:

```
server backend.example.com service=http resolve;
```

`sid=`*id*	задает ID сервера в группе. Если параметр не задан, то ID задается как шестнадцатеричный MD5-хэш IP-адреса и порта или пути UNIX-сокета.

`slow_start=`*время*	задает время восстановления веса сервера, возвращающегося к работе при балансировке нагрузки методом `round-robin` или `least_conn`. Если параметр задан и сервер после сбоя снова считается работающим с точки зрения `max_fails <max_fails>` и `upstream_probe`, то такой сервер равномерно набирает указанный для него вес в течение заданного времени. Если параметр не задан, то в аналогичной ситуации сервер сразу начинает работу с указанным для него весом.

Примечание

Если в апстриме задан только один `server`, `slow_start` не работает и будет игнорироваться.

state

Синтаксис	`state` *файл*;
По умолчанию	—
Контекст	upstream

Указывает *файл*, где постоянно хранится список серверов апстрима. При установке из *наших пакетов* 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 или перезагрузке конфигурации.

Осторожно

Чтобы использовать директиву `state` в блоке `upstream`, в нем не должно быть директив `server`, но нужна зона разделяемой памяти (*zone*).

sticky

Синтаксис	`sticky` cookie *name* [*attr=значение*]...; `sticky` route *$variable*...; `sticky` learn `zone=`*зона* `create=`*$create_var1*... `lookup=`*$lookup_var1*... [`header`] [`timeout=`*время*];
По умолчанию	—
Контекст	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*.

Внимание

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 удается 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 и клиенту, и серверу. 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 удается 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; он более гибок, так как назначает серверы на ходу, хранит сеансы в зоне общей 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 нескольку раз.

Идентификатором сессии служит значение первой непустой переменной, указанной с `create`; например, это может быть `cookie с проксируемого сервера`.

Сессии хранятся в зоне общей памяти; ее имя и размер задаются параметром `zone`. Если к сессии не было обращений в течение времени `timeout`, она удаляется. Значение по умолчанию — 10 минут.

Последующие запросы от клиентов, желающих использовать сессию, должны содержать ее идентификатор, причем так, чтобы он попал в непустую переменную, указанную с `lookup`; тогда его значение будет сопоставлено с сессиями в общей памяти. Если выбрать сервер не удается 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;
}
```

sticky_secret

Синтаксис	`sticky_secret` *строка*;
По умолчанию	—
Контекст	upstream

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	`sticky_strict on | off;`
По умолчанию	`sticky_strict off;`
Контекст	upstream

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

Синтаксис	`upstream` *имя* { ... }
По умолчанию	—
Контекст	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

Синтаксис	`zone` *имя* [*размер*];
По умолчанию	—
Контекст	upstream

Задает 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:

`$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.

3.4.2 Stream upstream

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	`upstream` *имя* { ... }
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	`server` *адрес* [*параметры*];
По умолчанию	—
Контекст	upstream

Задает 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, задает сразу несколько серверов.

Могут быть заданы следующие параметры:

`weight=`*число*	задает вес сервера по умолчанию 1.
`max_conns=`*число*	ограничивает максимальное число одновременных активных соединений к проксируемому серверу. Значение по умолчанию равно `0` и означает, что ограничения нет. Если группа не находится в *зоне* разделяемой памяти, то ограничение работает отдельно для каждого рабочего процесса.

`max_fails=`*число* — задает 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.

`max_fails=1`	число попыток по умолчанию
`max_fails=0`	отключает учет попыток

`fail_timeout=`*время* — задает:

`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.

`backup`	помечает сервер как запасной. На него будут передаваться запросы в случае, если не работают основные серверы.
`down`	помечает сервер как постоянно недоступный.
`drain`	помечает сервер как разгружаемый (draining); это значит, что он получает только запросы сессий, привязанных ранее через *sticky*. В остальном поведение такое же, как в режиме `down`.

Осторожно

Параметр `backup` нельзя использовать совместно с методами балансировки нагрузки *hash* и *random*.

Параметры `down` и `drain` взаимно исключающие.

`resolve`	Позволяет отслеживать изменения списка IP-адресов, соответствующего доменному имени, и обновлять его без перезагрузки конфигурации. При этом группа должна находиться в *зоне разделяемой памяти*; также должен быть определен `преобразователь имен в адреса`.
`service=`*имя*	Включает преобразование SRV-записей DNS и задает имя сервиса. При указании этого параметра необходимо также задать параметр *resolve*, не указывая порт сервера при имени хоста. Если в имени службы нет точек, формируется имя по стандарту RFC: к имени службы добавляется префикс `_`, затем через точку добавляется `_tcp`. Так, имя службы `http` даст в результате `_http._tcp`. 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. • SRV-записи с наивысшим приоритетом (те, которые имеют минимальное значение приоритета) разрешаются как основные серверы, а прочие записи становятся запасными серверами. Если `backup` установлено с `server`, SRV-записи с наивысшим приоритетом разрешаются как запасные серверы, а прочие записи игнорируются. • Вес аналогичен параметру `weight` директивы `server`. Если вес задан как в самой директиве, так и в SRV-записи, используется вес, установленный в директиве.

В этом примере выполняется поиск записи `_http._tcp.backend.example.com`:

```
server backend.example.com service=http resolve;
```

`slow_start=`*время*	задает время восстановления веса сервера, возвращающегося к работе при балансировке нагрузки методом `round-robin` или `least_conn`. Если параметр задан и сервер после сбоя снова считается работающим с точки зрения `max_fails` и `upstream_probe`, то такой сервер равномерно набирает указанный для него вес в течение заданного времени. Если параметр не задан, то в аналогичной ситуации сервер сразу начинает работу с указанным для него весом.

Примечание

Если в апстриме задан только один `server`, `slow_start` не работает и будет игнорироваться.

state

Синтаксис	`state` *файл*;
По умолчанию	—
Контекст	upstream

Указывает *файл*, где постоянно хранится список серверов апстрима. При установке из наших пакетов 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 `s_server`. Содержимое файла изменяется при любом изменении серверов в разделе `/config/stream/upstreams/` через API конфигурации. Файл считывается при запуске Angie или перезагрузке конфигурации.

Осторожно

Чтобы использовать директиву `state` в блоке `upstream`, в нем не должно быть директив `server`, но нужна зона разделяемой памяти (*zone*).

zone

Синтаксис	`zone` *имя* [*размер*];
По умолчанию	—
Контекст	upstream

Задает 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.

feedback

Синтаксис	`feedback` *переменная* [`inverse`] [`factor=`*число*] [`account=`*условная_переменная*];
По умолчанию	—
Контекст	upstream

Задает 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. Значение оценивается при каждом ответе от сервера и учитывается в скользящем среднем согласно настройкам `inverse` и `factor`.
`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`. Среднее рассчитывается по формуле экспоненциального сглаживания. 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

Синтаксис	`hash` *ключ* [`consistent`];
По умолчанию	—
Контекст	upstream

Задает 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. Пример использования:

```
hash $remote_addr;
```

Метод совместим с библиотекой Perl Cache::Memcached.

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 равным 160.

least_conn

Синтаксис	`least_conn`;
По умолчанию	—
Контекст	upstream

Задает для группы метод балансировки нагрузки, при котором соединение 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` `connect` | `first_byte` | `last_byte` [`factor=`*number*] [`account=`*condition_variable*];
По умолчанию	—
Контекст	upstream

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.

`connect`	Директива учитывает среднее время установки соединения.
`first_byte`	Директива использует среднее время получения первого байта ответа.
`last_byte`	Директива использует среднее время получения полного ответа.
`factor`	Выполняет ту же функцию, что и *response_time_factor*, и переопределяет его, если параметр задан.
`account`	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.

least_bandwidth

Синтаксис	`least_bandwidth both` | `upstream` | `downstream` [`factor=`*number*]
По умолчанию	`both`
Контекст	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.

`upstream`	Директива учитывает только пропускную способность, используемую **от клиента к серверу**.
`downstream`	Директива учитывает только пропускную способность, используемую **от сервера к клиенту**.
`both`	Директива учитывает сумму пропускной способности, используемой вверх (от клиента к серверу) и вниз по потоку (от сервера к клиенту). Режим по умолчанию.
`factor`	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_packets

Синтаксис	`least_packets both | upstream | downstream [factor=`*number*`]`
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.

`upstream`	Директива учитывает ттолько пакеты, отправленные **от клиента к серверу**.
`downstream`	Директива учитывает только пакеты, отправленные **от сервера к клиенту**.
`both`	Директива учитывает сумму пакетов, отправленных вверх (от клиента к серверу) и вниз по потоку (от сервера к клиенту).
`factor`	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.

random

Синтаксис	`random [two];`
По умолчанию	—
Контекст	upstream

Задает 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

Синтаксис	`response_time_factor` *число*`;`
По умолчанию	`response_time_factor 90;`
Контекст	upstream

Задает 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

Синтаксис	`sticky` route *$variable...*; `sticky` learn `zone`=*zone* `create`=*$create_var1...* `lookup`=*$lookup_var1...* [connect] [`timeout`=*time*];
По умолчанию	—
Контекст	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*.

Внимание

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 удается 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; он более гибок, так как назначает серверы на ходу, хранит сеансы в зоне общей 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 нескольку раз.

Идентификатором сессии служит значение первой непустой переменной, указанной с `create`; например, это может быть `имя проксируемого сервера`.

Сессии хранятся в зоне общей 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.

Последующие соединения от клиентов, желающих использовать сессию, должны содержать ее идентификатор, причем так, чтобы он попал в непустую переменную, указанную с `lookup`; тогда его значение будет сопоставлено с сессиями в общей памяти. Если выбрать сервер не удается или выбранный сервер не может обработать соединение, то будет выбран другой сервер согласно настроенному методу балансировки.

Параметр `connect` позволяет создать сессию сразу после получения заголовков ответа от проксируемого сервера. Без него сессия создается только после завершения обработки со.

В примере Angie создает и ищет сессии, используя переменную `$rdp_cookie`:

```
stream {

    upstream backend {

        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;
    }
```

```
    server {

        listen 127.0.0.1:3389;

        ssl_preread on;

        proxy_pass backend;
    }
}
```

sticky_strict

Синтаксис	`sticky_strict on | off;`
По умолчанию	`sticky_strict off;`
Контекст	upstream

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.

sticky_secret

Синтаксис	`sticky_secret` *строка*;
По умолчанию	—
Контекст	upstream

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
```

Встроенные переменные

Модуль `stream_upstream` поддерживает следующие встроенные переменные:

`$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*.

ГЛАВА 4

Глобальная балансировка

Глобальная балансировка, также GSLB (Global Server Load Balancing) или GTM (Global Traffic Management) – это сервис, который управляет ответами на DNS-запросы на основе доступности серверов приложений и производительности центров обработки данных.

Возможности:

- Гибкость настройки: использование разных алгоритмов балансировки для разных зон.
- Отказоустойчивость: быстрое переключение между серверами благодаря низкому TTL.
- Масштабируемость: легкое добавление серверов в группы и правил в зоны.

4.1 Настройка конфигурации

Для высокоуровневой маршрутизации реализована DNS-балансировка с помощью GSLB. GSLB-сервис балансирует серверы методом `round-robin`. Далее будут поддерживаться и другие методы балансировки.

В фоновом режиме проводятся проверки работоспособности, заданные в *конфигурации балансировщика нагрузки*. Серверы могут быть временно удалены из пула балансировки в случае, если не прошли проверку. При балансировке также учитываются данные по времени ответа сервера, полученные во время проверок работоспособности.

GSLB настраивается на том же виртуальном сервере Angie ADC, что и балансировщик нагрузки. Файлы конфигурации модуля GSLB находятся по пути `/etc/angie-adc-gslb/Corefile` и `/etc/angie-adc-gslb/gslbd.yaml`.

4.1.1 Пример конфигурации GSLB-модуля

```
options:
  ttl: 1

zones:
  www.example.org: # wildcard zone *.example.org
#    hostname: www.example.org # for wildcard zone checks
    rules:
      - rule: rr
        servers:
          - group: www

  test.example.com:
    rules:
      - rule: rr
        servers:
          - server: www5

rules:
  rr:
    type: round_robin
    servers:
      - server: www1
      - server: www2

groups:
  www:
    servers:
      - 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

  www5:
    addresses:
      - 172.22.0.10
```

В этом примере зоны `www.example.org` и `test.example.com` используют правило `rr` (round-robin), определенное в блоке `rules`. Если один из серверов, заданных в `servers`, станет недоступным, запросы автоматически перераспределятся на оставшиеся доступные серверы. Для серверов `www1`, `www2`, `www3` и `www4` заданы пассивные проверки работоспособности.

4.1.2 Глобальные параметры

```
options:
    ttl: 1
```

Параметр `ttl` задает значение TTL (Time to Live) для DNS-записей — как часто должны обновляться записи, что позволяет оценить доступность серверов. Значение `1` означает, что клиенты и кэширующие DNS-серверы должны обновлять записи каждую секунду.

4.1.3 Настройки зон

```
zones:
    www.example.org:
    rules:
       - rule: rr
         servers:
           - group: www

    test.example.com:
    rules:
       - rule: rr
         servers:
           - server: www5
```

Для зон `www.example.org` и `test.example.com` используются правила, определенные в блоке `rules`.

Особенности:

- Для каждой зоны требуется указать хотя бы одно правило `rule`.
- Если серверы для зоны не указаны, то будут взяты серверы из соответствующего правила.
- Если серверы для зоны указаны, то они имеют приоритет и будут использоваться вместо тех, которые заданы в правиле.
- Для `rule`, `group`, `server` можно задавать произвольные имена.

4.1.4 Правила распределения нагрузки

```
rules:
  rr:
    type: round_robin
    servers:
      - server: www1
      - server: www2
```

Правило `rr`:

- тип: `round_robin` (круговое распределение);
- серверы `www1` и `www2` принимают запросы по очереди без учета их веса;
- для `rule` можно задать произвольное имя;
- в `servers` можно указывать как серверы, так и группы серверов.

4.1.5 Группы серверов

```
groups:
  www:
    servers:
      - server: www3
      - server: www4
```

Задает состав группы серверов:

- для `group` можно задать произвольное имя;
- можно указывать как серверы, так и другие группы.

4.1.6 Конфигурация серверов

```
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

  www5:
    addresses:
      - 172.22.0.10
```

Задает конфигурацию серверов, используемых для балансировки нагрузки:

- Имя сервера.
- IP-адрес (адрес, на который будут направляться запросы при балансировке, можно указать несколько адресов списком).
- Пассивные проверки работоспособности сервера:
  - `healthpeers` — путь до Angie ADC API.
  - `healthinterval` — интервал между проверками; рекомендуемые значения от `5s` до `10m` в зависимости от ваших задач; параметр обязателен, если указан параметр `healthpeers`.
  - `healthtimeout` — таймаут на каждую проверку; рекомендуемые значения от `1s` до `1m` в зависимости от ваших задач; параметр обязателен, если указан параметр `healthpeers`.

  Если хотя бы один апстрим проходит проверку, то сервер считается рабочим. При непрохождении проверки сервер будет удален из пула балансировки до успешного прохождения проверки.

4.2 Просмотр и редактирование конфигурации

Файлы конфигурации модуля GSLB находятся по пути `/etc/angie-adc-gslb/Corefile` и `/etc/angie-adc-gslb/gslbd.yaml`.

Вы можете редактировать их в консоли Angie ADC.

Чтобы изменить конфигурацию GSLB, 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 домены, для которых будет настраиваться сервис GSLB, и нажмите `Сохранить`.

   Например:

   ```
   www.example.org {
       gslb
   }
   ```
4. В раскрывающемся списке в верхней части окна выберите файл `gslbd.yaml`.

   Откроется файл с конфигурацией GSLB для указанных доменов.
5. Внесите необходимые изменения и нажмите `Сохранить`.

   Изменения будут применены сразу.

Подробнее о настройке GSLB смотрите в статье *Настройка конфигурации*.

ГЛАВА 5

Обеспечение высокой доступности

Режим высокой доступности обеспечивается протоколами динамической маршрутизациии (BGP, OSPF, RIP, PBR) и настраивается вручную. Также доступно использование протокола BFD для уменьшения времени реакции на какое-либо событие.

- *HA-решение в режиме резервирования с помощью протокола BGP*
- *HA-решение в режиме резервирования с помощью протокола OSPF*
- *Использование протокола BFD*

Примечание

В настоящем разделе рассматриваются вопросы обеспечения высокой доступности системы балансировки Angie ADC. Обеспечение отказоустойчивости сопутствующей сетевой инфраструктуры выходит за рамки решения.

5.1 Решение в режиме резервирования с использованием протокола BGP

В этом разделе описана настройка высокой доступности с использованием протокола BGP в режиме резервирования (Active-Standby).

Примечание

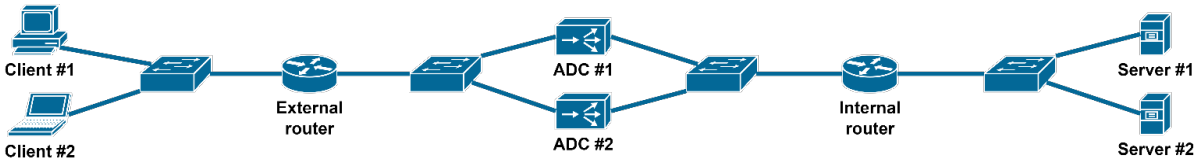
Для настройки используется *командная строка Angie ADC CLI*.

5.1.1 Динамическая маршрутизация

В качестве протокола динамической маршрутизации используется BGPv4 для адресного семейства IPv4 unicast. Отказоустойчивость в рамках IPv6 unicast настраивается аналогично.

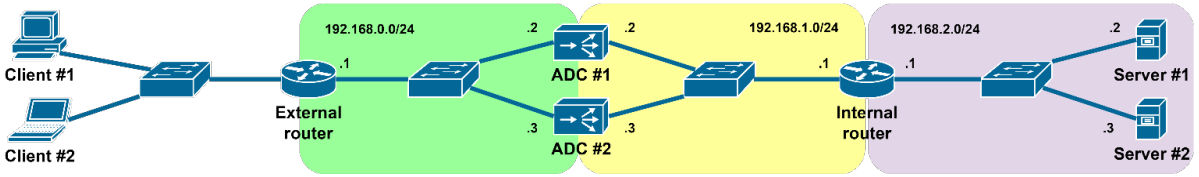
В рассматриваемом примере будем считать, что в нормальном режиме работы ADC #1 выполняет функции активной системы балансировки трафика, тогда как ADC #2 остается в резерве.

Пример типового участка сети с системами балансировки Angie ADC представлен на схеме ниже.



Введем IP-адресацию:

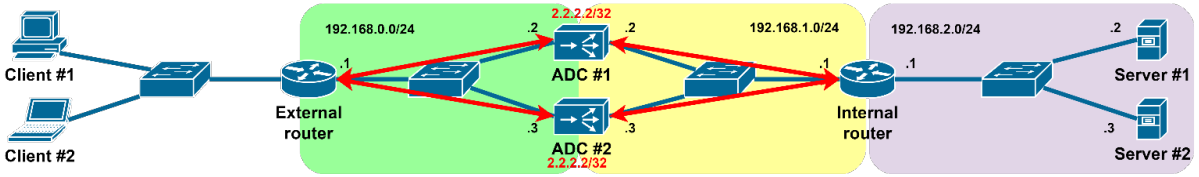
- В сегменте сети, куда подключаются External router, ADC #1 и ADC #2, будет использоваться подсеть 192.168.0.0/24.
- В сегменте сети, куда подключаются Internal router, ADC #1 и ADC #2, будет использоваться подсеть 192.168.1.0/24.
- Четвертый октет адреса для каждого устройства (в рамках соответствующей подсети) показан на схеме ниже.



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.

BGP-сессии

Устанавливаемые BGP-сессии представлены на схеме ниже:



Красным цветом на схеме 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, манипулируя атрибутами пути в BGP.

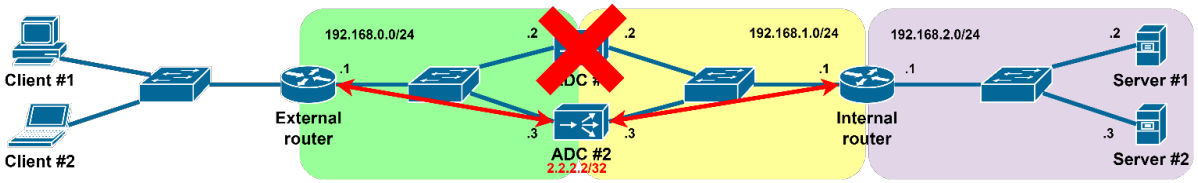
Так как все BGP-взаимодействие производится строго в рамках одной автономной системы, мы будем использовать атрибут Local preference (чем больше, тем лучше) для выбора наиболее оптимального пути. Трафик от серверов в сторону клиента должен проходить через ту же систему балансировки, что и трафик от клиентов к серверам. Добиться этого можно также путем манипулирования значением атрибута Local preference для префикса 0.0.0.0/0.

Механизмы переключения трафика на резервную систему балансировки в случае сбоя

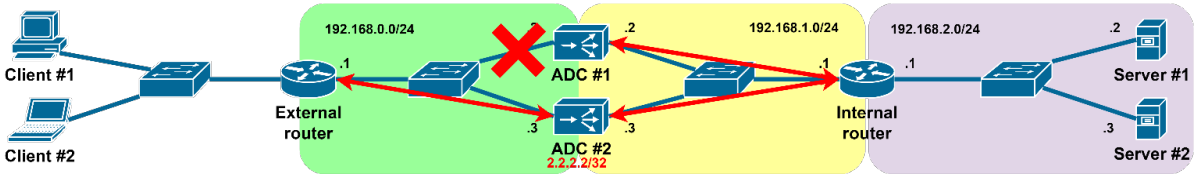
Ниже описаны сценарии работы системы балансировки при сбоях.

Полный отказ ADC #1

Рассмотрим самый простой вариант – полный отказ или полное отключение от сети ADC #1. В этом случае обе BGP-сессии с ADC #1 разрываются, никакие объявления маршрутов через ADC #1 не пролетают. Остается связь по BGP только с ADC #2. То есть переключение на резервную систему производится полностью автоматически, никакие дополнительные настройки не требуются.

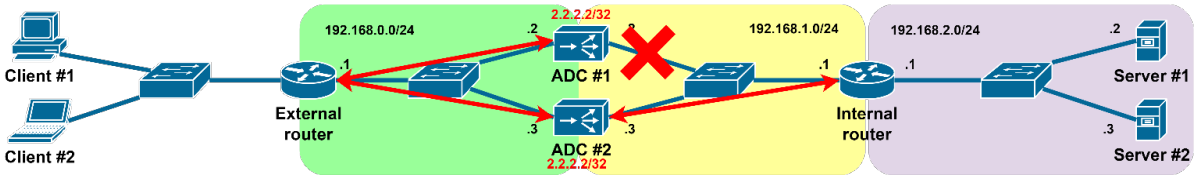


Отказ внешнего интерфейса ADC #1



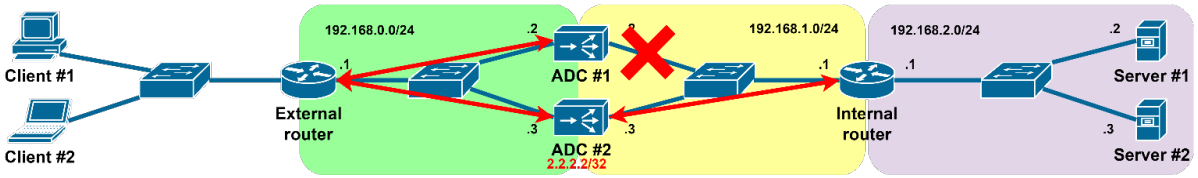
В этом случае разрывается BGP-сессия между ADC #1 и External router. ADC #1 теперь не может объявить маршрут до адреса виртуального сервера в сторону External router. Несмотря на наличие BGP-сессии между ADC #1 и Internal router, система балансировки ADC #1 не отправляет маршрут по умолчанию в сторону 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



Если BGP-сессия 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 недостижимы. Таким образом, мы получаем классический black hole для трафика.

Чтобы избежать указанной проблемы, 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.



5.1.2 Настройки 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 системы балансировки 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
!Интерфейс лупбек 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 системы балансировки 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
!Интерфейс лупбек 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                  192.168.1.3              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 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
```

5.1.3 Заключение

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 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.

5.2 Решение в режиме резервирования с использованием протокола OSPF

В этом разделе описана настройка высокой доступности с использованием протокола OSPF в режиме резервирования (Active-Standby).

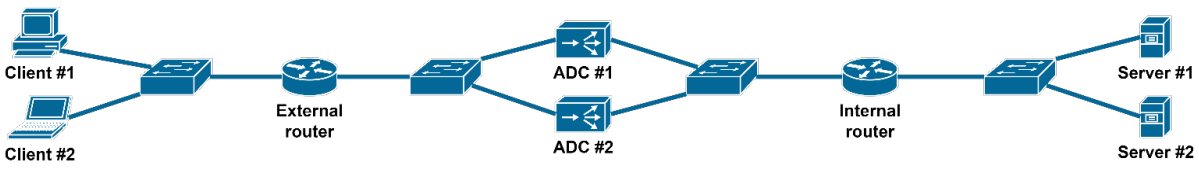
> **Примечание**
>
> Для настройки используется *командная строка Angie ADC CLI*.

5.2.1 Динамическая маршрутизация

В качестве протокола динамической маршрутизации используется OSPFv2 для адресного семейства IPv4 unicast. Отказоустойчивость в рамках IPv6 unicast настраивается аналогично.

В рассматриваемом примере будем считать, что в нормальном режиме работы ADC #1 выполняет функции активной системы балансировки трафика, тогда как ADC #2 остается в резерве.

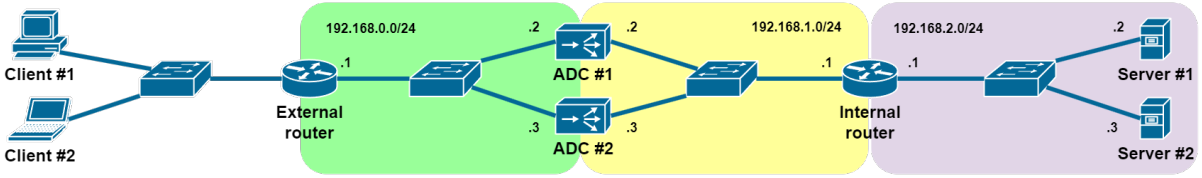
Пример типового участка сети с системами балансировки Angie ADC представлен на схеме ниже.



Введем IP-адресацию:

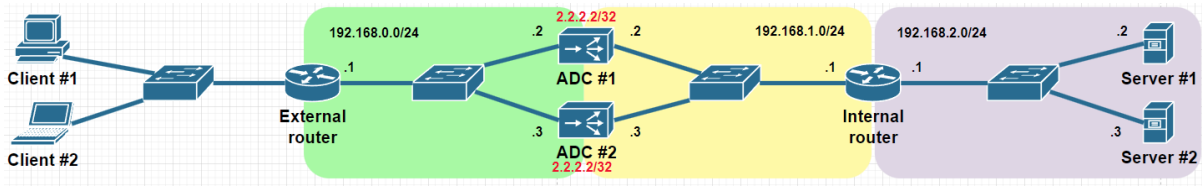
- В сегменте сети, куда подключаются External router, ADC #1 и ADC #2, будет использоваться подсеть 192.168.0.0/24.

- 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.



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. В данном документе будем использовать OSPF-домен, включающий в себя четыре устройства: External router, Internal router, ADC #1 и ADC #2.

IP-адреса виртуальных серверов



Красным текстом на схеме 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.

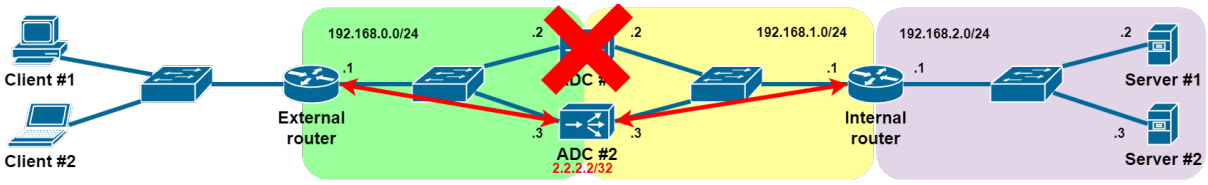
Трафик от серверов в сторону клиента должен проходить через ту же систему балансировки, что и трафик от клиентов к серверам. Добиться этого можно также путем манипулирования значением метрики для префикса 0.0.0.0/0. Манипуляции с маршрутом по умолчанию нужны при включении опции ip transparency. В противном случае трафик и так будет возвращаться через ту же систему балансировки, так как поведение ADC напоминает операцию sNAT.

Механизмы переключения трафика на резервную систему балансировки в случае сбоя

Ниже описаны сценарии работы системы балансировки при сбоях.

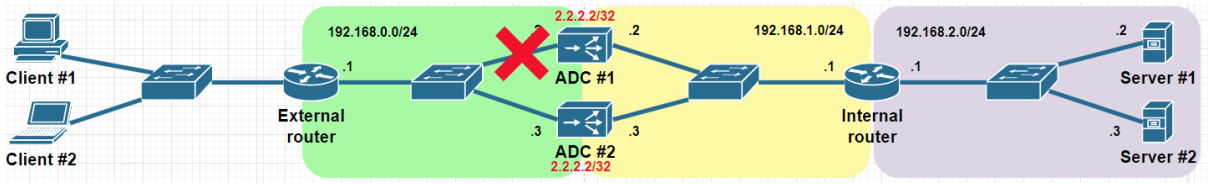
Полный отказ ADC #1

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.



Отказ внешнего интерфейса Angie ADC #1

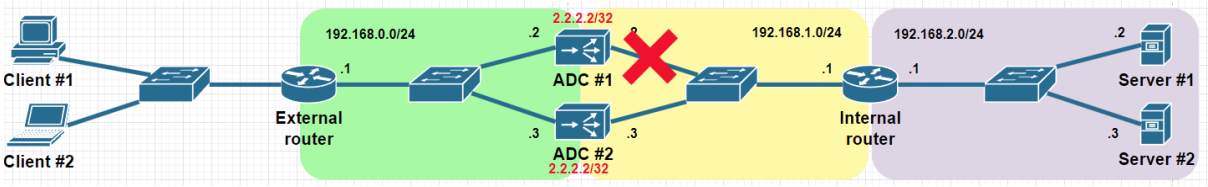
Предлагаемая схема не имеет возможности отслеживания и реагирования на частичные отказы. Пример такого отказа представлен на схеме ниже.



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.

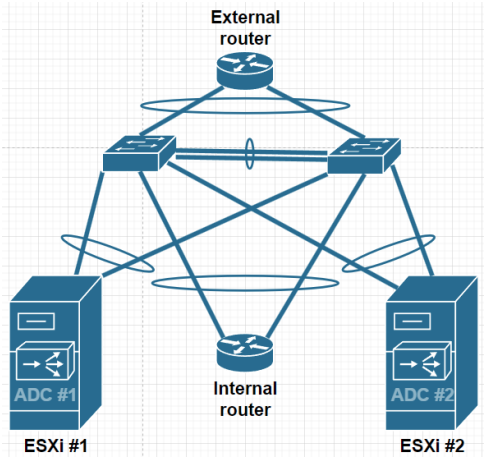
Отказ линка между ADC #1 и Internal router

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 недостижимы. Таким образом, мы получаем классический black hole для трафика.



Чтобы избежать указанной проблемы, необходимо обеспечить отказоустойчивое подключение хоста виртуализации (на котором работает ADC) к сети так, чтобы ситуация отказа только одного из линков была невозможной.

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 тегированными 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.



5.2.2 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
!Интерфейс лупбек 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
!Интерфейс лупбек 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 10 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
```

Маршрутизатора 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 [110/10] via 192.168.0.3, 00:00:10, GigabitEthernet0/0
```

После возвращения в строй системы балансировки ADC #1 маршрутизация через это устройство восстанавливается.

5.2.3 Заключение

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.

5.3 Использование протокола BFD для уменьшения времени реакции

5.3.1 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 сбои, 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.

Примечание

Для настройки используется *командная строка Angie ADC CLI*.

5.3.2 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-va# 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
```

5.3.3 Настройка с BGP

По умолчанию при указании BGP-соседа BFD-сессия между маршрутизаторами не устанавливается.

Пример такой конфигурации:

```
angie-va# 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-va# 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-va# sho bfd pe
BFD Peers:
angie-va#
```

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
```

5.3.4 Настройка с VRRP

На данный момент нет интеграции с BFD.

5.3.5 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 эксплуатакции 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).

5.3.6 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-va# conf t
angie-va(config)# bfd
angie-va(config-bfd)# peer
A.B.C.D   IPv4 peer address
X:X::X:X  IPv6 peer address
angie-va(config-bfd)# peer
```

Некоторые настройки необходимо задать в момент создания пира (интерфейс, локальный адрес, multihop и vrf), остальные параметры можно будет изменить потом.

```
angie-va(config-bfd)# peer 192.168.0.3
<cr>
interface      Interface information
local-address  Configure local address
multihop       Configure multihop
vrf            Configure VRF
angie-va(config-bfd)# peer 192.168.0.3
angie-va(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.

5.3.7 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)#
```

ГЛАВА 6

Работа в веб-консоли Angie ADC

Текущая версия: 1.2.0

В веб-консоли Angie ADC вы можете просмотреть:

- общую статистику работы балансировщика нагрузки;
- конфигурацию балансировщика нагрузки;
- конфигурацию GSLB.

Дополнительно можно перейти в консоль Console Light, чтобы просмотреть детальную статистику работы серверов в режиме реального времени.

Также доступна настройка модулей и управление пользователями.

6.1 Просмотр статистики балансировщика нагрузки

Чтобы просмотреть статистику для балансировщика нагрузки, выполните следующие действия:

1. Откройте консоль Angie ADC.
2. На вкладке `Панель мониторинга` выберите нужный вам балансировщик нагрузки и нажмите на соотвествующий виджет.

   Откроется экран мониторинга, предоставляющий доступ к детализированным графикам со статистикой. Подробнее см. *экран мониторинга балансировщика нагрузки*.

6.2 Просмотр конфигурации балансировщика нагрузки

Чтобы просмотреть конфигурацию балансировщика нагрузки, выполните следующие действия:

1. Откройте консоль Angie ADC.
2. На вкладке `Панель мониторинга` нажмите на виджет балансировщика нагрузки.

   Откроется экран со статистикой для этого балансировщика.
3. Нажмите кнопку `Конфигурация` в верхней правой части экрана.

   Откроется конфигурационный файл балансировщика нагрузки.

Подробнее о настройке балансировщика нагрузки смотрите в статье *Настройка конфигурации*.

6.3 Просмотр конфигурации GSLB

Файлы конфигурации модуля GSLB находятся по пути `/etc/angie-adc-gslb/Corefile` и `/etc/angie-adc-gslb/gslbd.yaml`. Вы можете просматривать и редактировать их в консоли Angie ADC.

Чтобы просмотреть или изменить конфигурацию GSLB, 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 домены, для которых будет настраиваться сервис GSLB, и нажмите `Сохранить`.

   Например:

   ```
   www.example.org {
       gslb
   }
   ```
4. В раскрывающемся списке в верхней части окна выберите файл `gslbd.yaml`.

   Откроется файл с конфигурацией GSLB для указанных доменов.
5. Внесите необходимые изменения и нажмите `Сохранить`.

   Изменения будут применены сразу.

Подробнее о настройке GSLB смотрите в статье *Настройка конфигурации*.

6.4 Управление пользователями

6.4.1 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.

6.4.2 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.

6.4.3 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 `...` (троеточие) → `Изменить` рядом с пользователем, данные которого вы хотите изменить.
3. В открывшемся окне укажите новые данные и нажмите `Сохранить`.

   Данные пользователя будут обновлены.

6.4.4 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 и не будет больше отображаться в списке пользователей консоли.

Чтобы изменить статус существующего пользователя консоли, выолните следующие действия:

1. Откройте консоль Angie ADC.
2. На вкладке `Пользователи` нажмите на `...` (троеточие) → `Изменить` рядом с пользователем, данные которого вы хотите изменить.
3. В открывшемся окне поменяйте статус пользователя и нажмите `Сохранить`.

   Данные пользователя будут обновлены.

6.4.5 Удаление пользователя

Чтобы удалить существующего пользователя консоли, выполните следующие действия:

1. Откройте консоль Angie ADC.
2. На вкладке `Пользователи` нажмите на `...` (троеточие) → `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.

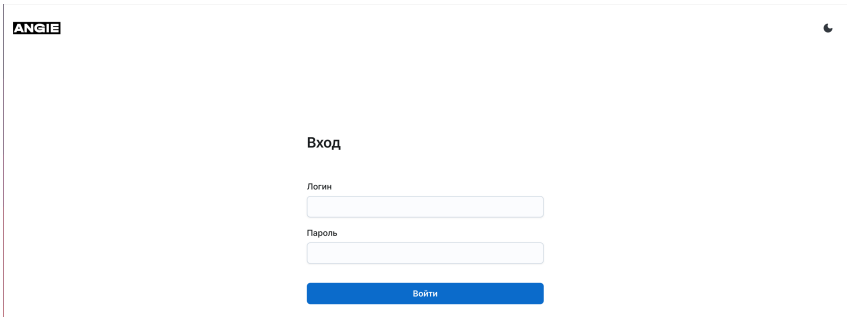
6.5 Интерфейс веб-консоли Angie ADC

Веб-консоль Angie ADC представляет собой единый экран с набором вкладок, каждая из которых содержит ряд виджетов с данными мониторинга нагрузки. Также в веб-консоли вы можете *управлять пользователями консоли*, настраивать *конфигурацию балансировщика нагрузки* и *глобальную балансировку (GSLB)*.

Текущая версия: 1.2.0

В разделах ниже описания элементов интерфейса даны в порядке слева направо.

6.5.1 Экран "Вход"

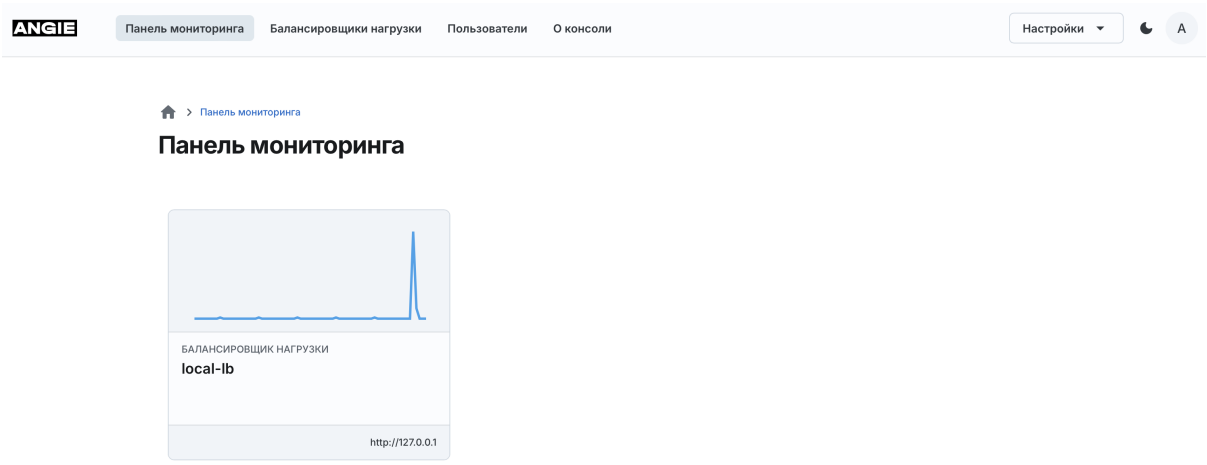


Чтобы войти в консоль и получить доступ к ее функциям, здесь необходимо ввести логин и пароль. Реквизиты для первого входа предоставляются по запросу. Рекомендуется сменить пароль после первого входа.

Элементы интерфейса:

`Логин`	Поле для ввода логина
`Пароль`	Поле для ввода пароля
`Войти`	Кнопка входа

6.5.2 Вкладка "Панель мониторинга"

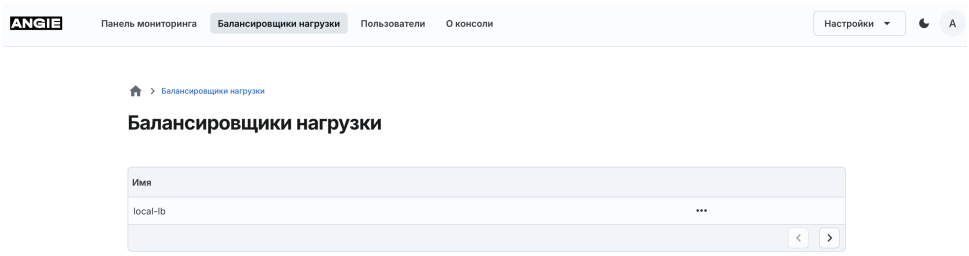


Панель мониторинга предоставляет доступ к функциональности мониторинга Angie ADC. Здесь представлены балансировщики нагрузки, настроенные в системе.

Элементы интерфейса:

`Панель мониторинга`	Текущая вкладка
`Балансировщики нагрузки`	Вкладка балансировщиков нагрузки
`Пользователи`	Вкладка пользователей
`О консоли`	Вкладка сведений
`Балансировщик нагрузки`	Виджет балансировщика нагрузки
`Настройки`	Кнопка перехода к редактору конфигурации GSLB
Значок "солнце-луна"	Переключатель темного и светлого режимов интерфейса
Инициал пользователя	Контекстное меню пользователя, в том числе команда "Выйти"

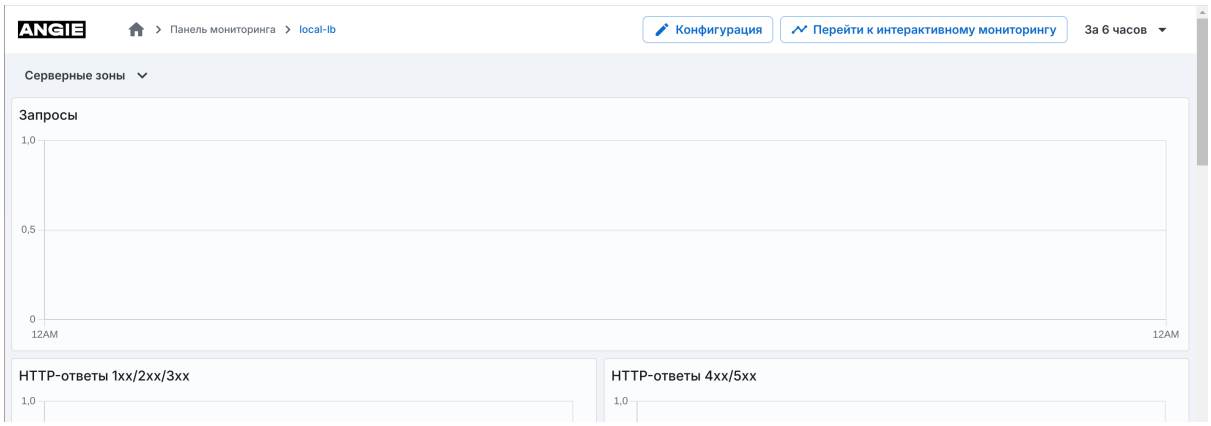
Виджет "Балансировщик нагрузки"



Здесь приведен сводный график нагрузки балансировщика в исторической перспективе, при наведении указателя мыши на который появляется плавающая подсказка с датой и соответствующим выбранному показателю числа запросов. Также здесь указан адрес сервера-балансировщика.

При щелчке открывается экран мониторинга балансировщика.

6.5.3 Экран мониторинга "Балансировщик нагрузки"



Экран предоставляет доступ к детализированным графикам со статистикой по отдельному балансировщику.

Элементы интерфейса:

`Серверные зоны`	Раскрывающийся виджет с графиками серверных зон
`Зоны апстримов`	Раскрывающийся виджет с графиками зон апстримов
Список `Все`	Раскрывающийся список для выбора всех или отдельных серверов при показе статистики
Список `За 6 часов`	Раскрывающийся список для выбора временного периода при показе статистики (30 минут, 3 часа, 6 часов)

Виджет "Серверные зоны"

Здесь собрана статистика по серверным зонам разделяемой памяти.

Элементы интерфейса:

`Запросы`	График числа запросов с разделением по отдельным серверам
`HTTP-ответы`	Графики числа HTTP-ответов с определенными кодами состояния с разделением по группам кодов (1xx, 2xx, 3xx и 4xx, 5xx)
`Отправленные данные`	График объема отправленных данных с разделением по отдельным серверам
`Полученные данные`	График объема полученных данных с разделением по отдельным серверам
`Успешные SSL-рукопожатия`	График числа успешных SSL-рукопожатий с разделением по отдельным серверам
`Неудачные SSL-рукопожатия`	График числа неудачных SSL-рукопожатий с разделением по отдельным серверам
`Серверные зоны`	Таблица серверных зон со статистикой запросов, ответов и данных с разделением по отдельным зонам
`Зоны путей (Location)`	Таблица зон location со статистикой запросов, ответов и данных с разделением по отдельным location

Виджет "Зоны апстримов"

Здесь собрана статистика по зонам разделяемой памяти для апстримов.

Элементы интерфейса:

`Запросы`	График числа запросов с разделением по отдельным серверам
`HTTP-ответы`	Графики числа HTTP-ответов с определенными кодами состояния с разделением по группам кодов (1xx, 2xx, 3xx и 4xx, 5xx)
`Отправленные данные`	График объема отправленных данных с разделением по отдельным серверам
`Полученные данные`	График объема полученных данных с разделением по отдельным серверам

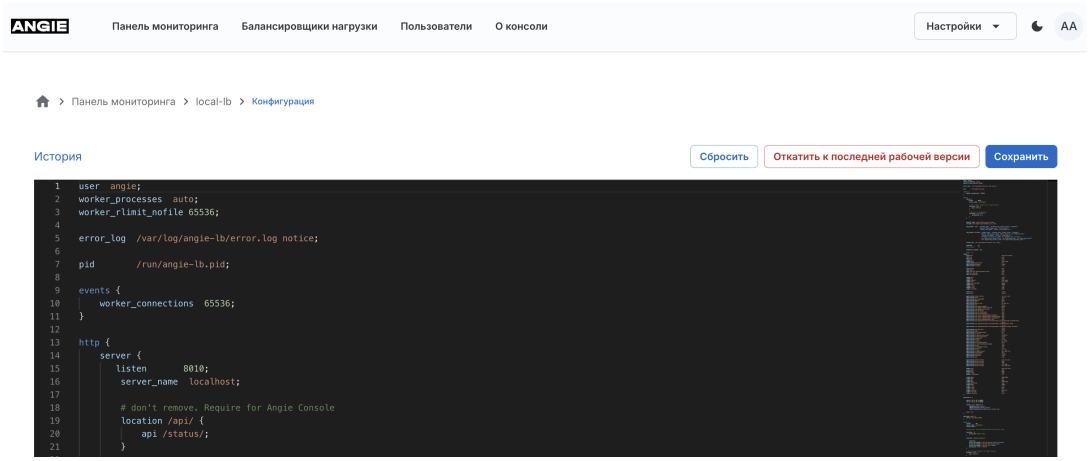
Кнопка "Перейти к интерактивному мониторингу"

Нажатие кнопки открывает *консоль Console Light* для этого балансировщика.

Кнопка "Конфигурация"

Нажатие кнопки открывает интерактивный редактор конфигурации балансировщика, где можно видеть все настройки в сводном текстовом виде и при необходимости изменить их вручную.

6.5.4 Экран "Конфигурация"



На этом экране можно просмотреть файл конфигурации балансировщика нагрузки и отредактировать его. Также доступна история изменения файла конфигурации.

Элементы интерфейса:

`История`	Ссылка на список сохраненных файлов конфигурации балансировщика нагрузки. Позволяет выбрать сохраненную версию конфигурации и применить ее.
`Сбросить`	Кнопка сброса изменений в конфигурации при редактировании.
`Сохранить`	Кнопка сохранения изменений в конфигурации при редактировании.
`Откатить к последней рабочей версии`	Кнопка возврата к последней рабочей версии конфигурации.

6.5.5 Вкладка "Балансировщики нагрузки"

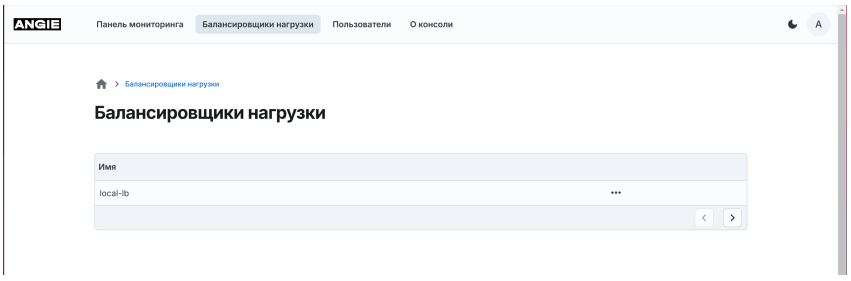


Таблица на этой вкладке содержит список балансировщиков нагрузки, зарегистрированных в Angie ADC.

Элементы интерфейса:

`Имя`	Имя, данное в системе балансировщику; щелчок этой ссылки открывает экран изменения свойств балансировщика
`...` (троеточие)	Контекстное меню с командой: • `Перейти к мониторингу` открывает экран мониторинга балансировщика

6.5.6 Вкладка "Пользователи"

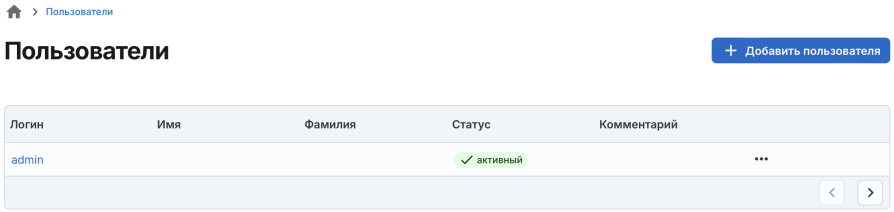
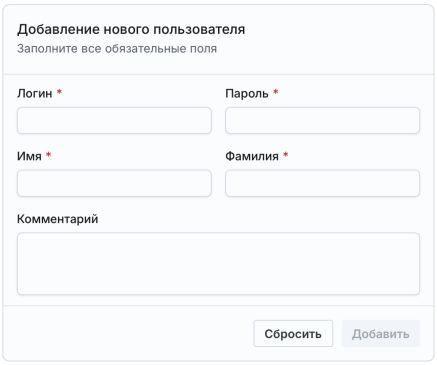


Таблица на этой вкладке содержит список пользователей, зарегистрированных в Angie ADC, и позволяет управлять как их составом, так и свойствами отдельных пользователей.

Элементы интерфейса:

`+ Добавить пользователя`	Нажатие этой кнопки открывает экран добавления нового пользователя
`Логин`	Учетная запись пользователя в системе; щелчок этой ссылки открывает экран изменения данных пользователя
`Имя`	Собственное имя пользователя
`Фамилия`	Фамилия пользователя
`Статус`	Статус пользователя в системе (`активный`, `неактивный`, `удален`)
`Комментарий`	Примечание
`...` (троеточие)	Контекстное меню с двумя командами: • `Изменить` открывает экран изменения данных пользователя • `Удалить` удаляет запись о пользователе

6.5.7 Экран "Добавление пользователя"



Экран предоставляет возможность добавить запись о пользователе в системе.

Элементы интерфейса:

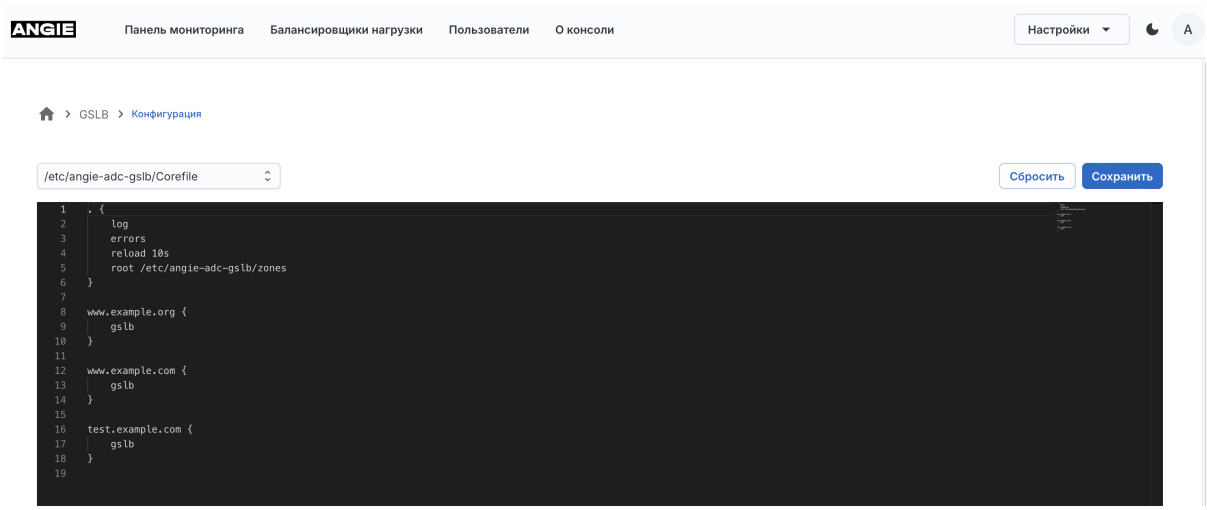
`Логин`	Логин пользователя в системе
`Пароль`	Пароль пользователя в системе (проверьте вводимое значение!)
`Имя`	Собственное имя пользователя
`Фамилия`	Фамилия пользователя
`Комментарий`	Примечание
`Сбросить`	Кнопка сброса данных
`Добавить`	Кнопка добавления пользователя

6.5.8 Экран "Изменение данных пользователя"

Экран предоставляет возможность изменить существующую запись о пользователе в системе.

Элементы интерфейса аналогичны тем, что представлены на экране "Добавление нового пользователя", за исключением кнопки "Сохранить", изменяющей данные пользователя.

6.5.9 Экран "Глобальная балансировка"

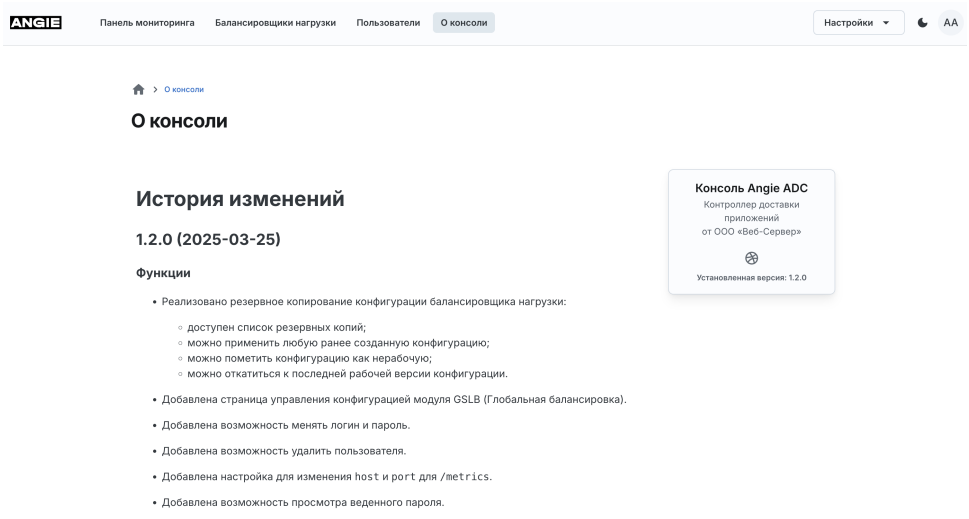


На этом экране можно просмотреть файлы конфигурации GSLB и отредактировать их.

Элементы интерфейса:

Раскрывающийся список файлов конфигурации	Позволяет выбрать файл конфигурации GSLB для просмотра и редактирования
`Сбросить`	Кнопка сброса изменений
`Сохранить`	Кнопка сохранения изменений в конфигурации

6.5.10 Вкладка "О консоли"



На этой вкладке представлены сведения об используемой версии Angie ADC, а также краткий перечень изменений и функций, добавленных в каждой версии.

6.6 Мониторинг и статистика в Console Light

Console Light — это консоль для мониторинга активности в реальном времени, отображающая ключевые показатели нагрузки и производительности сервера. Консоль основана на возможностях API-интерфейса Angie ADC; данные мониторинга активности генерируются в реальном времени.

Пример развернутой и настроенной консоли: https://console.angie.software/

При использовании в составе продукта Angie ADC консоль Console Light устанавливается и настраивается автоматически.

Текущая версия: 1.7.2

6.6.1 Переход в Console Light

1. Откройте консоль Angie ADC.
2. На вкладке `Панель мониторинга` щелкните виджет балансировщика нагрузки.

   Откроется экран со статистикой для этого балансировщика.
3. В верхней части экрана нажмите кнопку `Перейти к интерактивному мониторингу`.

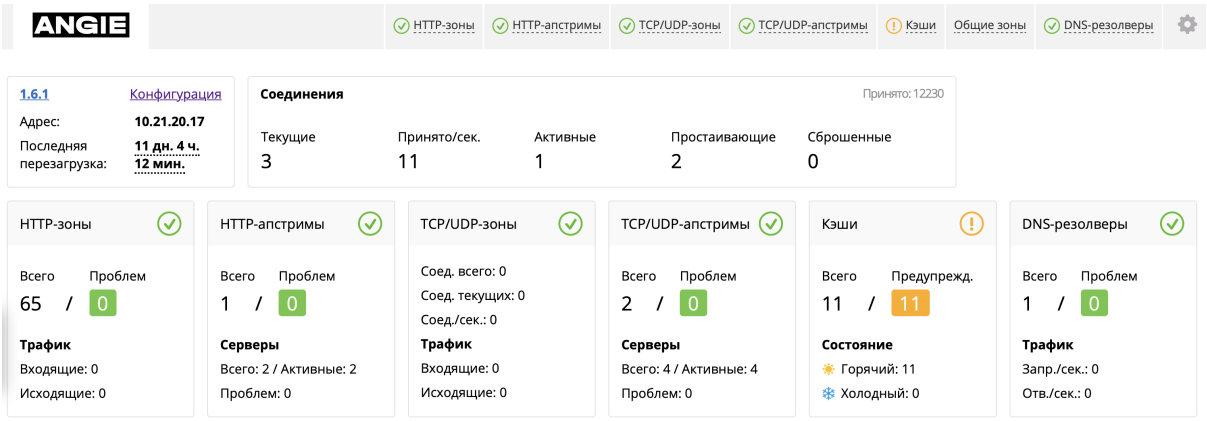
   Откроется облегченная консоль Console Light с данными мониторинга активности в реальном времени для этого балансировщика.

6.6.2 Интерфейс

Console Light представляет собой единый экран с набором вкладок, каждая из которых содержит ряд виджетов с данными мониторинга.

> **Подсказка**
>
> В разделах ниже описания элементов интерфейса даны в порядке слева направо.

Вкладка "Angie"



Это основная вкладка, где в сводном виде отображаются основные показатели мониторинга Angie ADC, сведенные на основе данных из нескольких разделов API.

6.6.3 Виджет "<версия>"

Здесь выводится номер версии Console Light со ссылкой на соответствующую документацию, а также адрес сервера и время последней перезагрузки конфигурации.

Ссылка `Конфигурация` открывает список файлов конфигурации, загруженных на сервере. Затем каждый файл можно просмотреть в компактном виде с подсветкой синтаксиса.

6.6.4 Виджет "Соединения"

Здесь представлена основная статистика серверных соединений, формируемая на основе раздела API `/status/connections/`:

`Текущие`	Текущее количество соединений
`Принято/сек.`	Число принимаемых за секунду соединений
`Активные`	Число активных соединений
`Простаивающие`	Число соединений в состоянии ожидания
`Сброшенные`	Количество сброшенных соединений

Также доступно:

`Принято`	Общее число соединений, принятых с последней перезагрузки сервера

6.6.5 Виджет "HTTP-зоны"

Здесь представлена статистика зон разделяемой памяти контекста `http`, формируемая на основе раздела API `/status/http/server_zones/`:

`Всего`	Общее количество зон
`Проблем`	Количество зон с какими-либо проблемами
`Трафик`	Общий объем входящего и исходящего трафика

6.6.6 Виджет "HTTP-апстримы"

Здесь представлена статистика апстримов контекста `http`, формируемая на основе раздела API `/status/http/upstreams/`:

`Всего`	Общее количество апстримов
`Проблем`	Количество апстримов с какими-либо проблемами
`Серверы`	Статистика серверов с разделением по состоянию

6.6.7 Виджет "TCP/UDP-зоны"

Здесь представлена статистика зон разделяемой памяти контекста `stream`, формируемая на основе раздела API `/status/stream/server_zones/`:

`Соед. всего`	Общее количество клиентских соединений
`Соед. текущих`	Текущее количество клиентских соединений
`Соед./сек.`	Количество обрабатываемых в секунду соединений

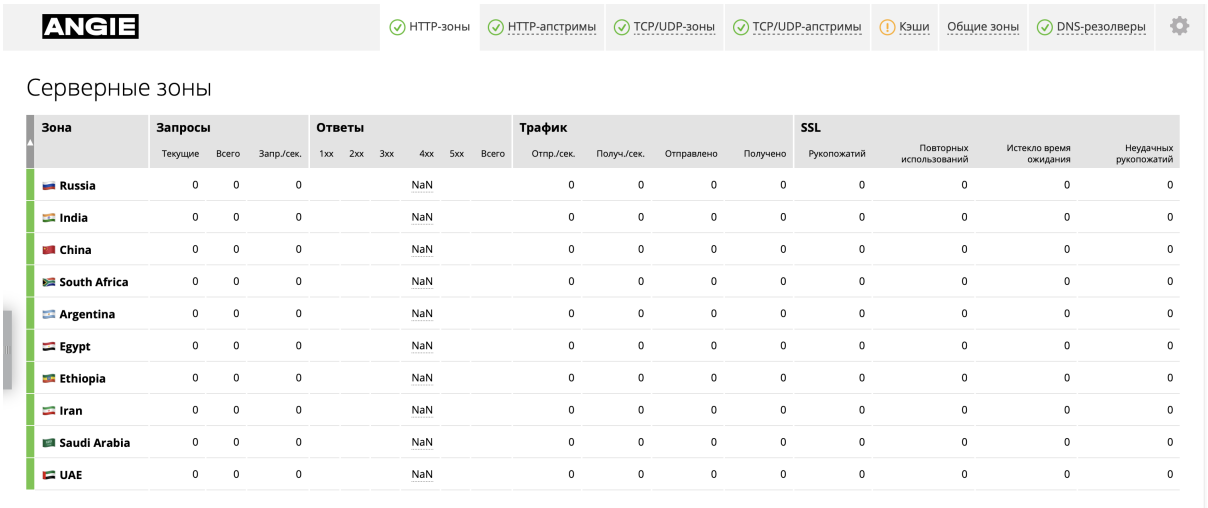
6.6.8 Виджет "TCP/UDP-апстримы"

Здесь представлена статистика апстримов контекста `stream`, формируемая на основе раздела API `/status/stream/upstreams/`:

`Всего`	Общее количество апстримов
`Проблем`	Количество апстримов с какими-либо проблемами
`Серверы`	Статистика серверов с разделением по состоянию

Вкладка "HTTP-зоны"

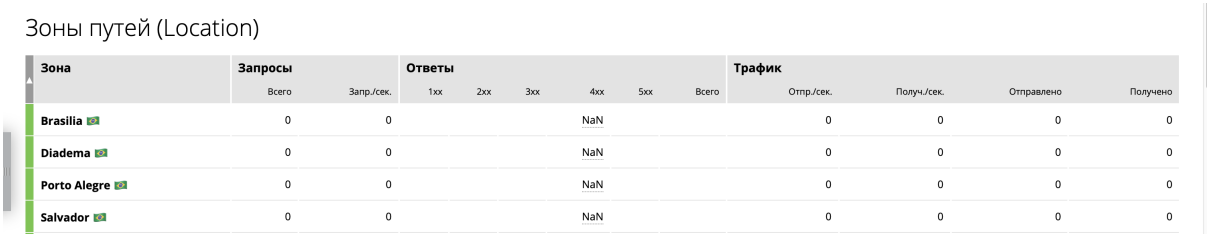
6.6.9 Раздел "Серверные зоны"



Здесь в сводном виде отображается статистика мониторинга зон разделяемой памяти для контекста `server` в `http`, формируемая на основе раздела API `/status/http/server_zones/`. Для каждой зоны представлены следующие данные:

`Зона`	Имя зоны **Подсказка** Щелкните стрелку рядом с пунктом `Зона`, чтобы отсортировать зоны по алфавиту или порядку в конфигурации.
`Запросы`	Общее количество запросов, а также число запросов в секунду
`Ответы`	Количество ответов с разбиением по кодам статуса, а также их общее количество
`Трафик`	Скорость исходящего и входящего трафика, а также общие объемы исходящего и входящего трафика
`SSL`	Суммарные показатели количества: успешных SSL-рукопожатий; повторных использований SSL-сессий; SSL-рукопожатий с истекшим таймаутом; неуспешных SSL-рукопожатий

6.6.10 Раздел "Зоны путей (Location)"



Здесь в сводном виде отображается статистика мониторинга зон разделяемой памяти для контекста `location` в `http`, формируемая на основе раздела API `/status/http/location_zones/`. Для каждой зоны представлены следующие данные:

`Зона`	Имя зоны **Подсказка** Щелкните стрелку рядом с пунктом `Зона`, чтобы отсортировать зоны по алфавиту или порядку в конфигурации.
`Запросы`	Общее количество запросов, а также число запросов в секунду
`Ответы`	Количество ответов с разбиением по кодам статуса, а также их общее количество
`Трафик`	Скорость исходящего и входящего трафика, а также общие объемы исходящего и входящего трафика

6.6.11 Раздел "Зоны ограничения соединений (Limit Conn)"

Здесь приведена статистика зон `limit_conn` в контексте `http`, формируемая на основе раздела API `/status/http/limit_conns/`. Для каждой зоны представлены следующие данные:

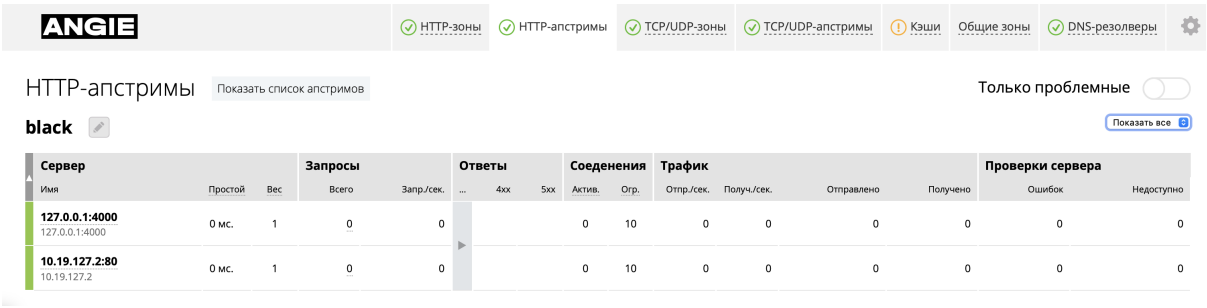
`Зона`	Имя зоны **Подсказка** Щелкните значок рядом с пунктом `Зона`, чтобы открыть или закрыть график со следующими показателями.
`Передано`	Общее количество проксированных соединений
`Отклонено`	Общее количество сброшенных соединений
`Сброшено`	Общее количество соединений, сброшенных из-за переполнения хранилища зоны
`Пропущено`	Общее количество соединений, переданных с нулевым или превосходящим 255 байт ключом

6.6.12 Раздел "Зоны ограничения запросов (Limit Req)"

Здесь приведена статистика зон `limit_reqs` в контексте `http`, формируемая на основе раздела API `/status/http/limit_reqs/`. Для каждой зоны представлены следующие данные:

`Зона`	Имя зоны **Подсказка** Щелкните значок рядом с пунктом `Зона`, чтобы открыть или закрыть график со следующими показателями.
`Передано`	Общее количество проксированных соединений
`Задержано`	Общее количество задержанных соединений
`Отклонено`	Общее количество сброшенных соединений
`Сброшено`	Общее количество соединений, сброшенных из-за переполнения хранилища зоны
`Пропущено`	Общее количество соединений, переданных с нулевым или превосходящим 255 байт ключом

Вкладка "HTTP-апстримы"



На этой вкладке в сводном виде отображается статистика мониторинга апстримов контекста `http`, формируемая на основе раздела API `/status/http/upstreams/`.

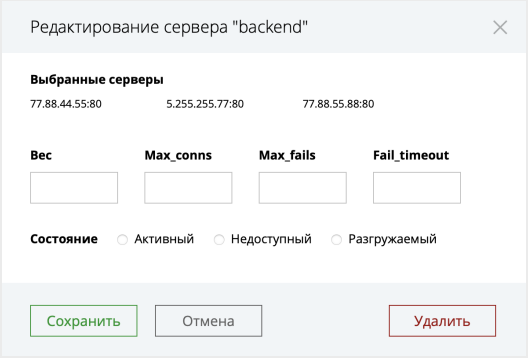
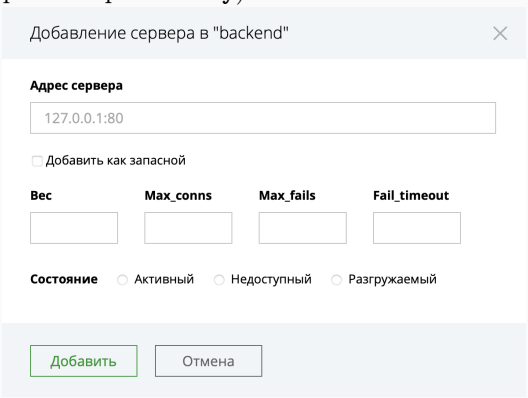
- Кнопка `Показать список апстримов` переключает краткий список апстримов с указанием числа проблемных апстримов и пиров.
- Переключатель `Только проблемные` переключает режим вывода статистики по проблемным апстримам.
- Кнопка редактирования переключает интерфейс `редактирования апстрима`.
- Раскрывающийся список с правой стороны таблицы каждого апстрима позволяет отфильтровать серверы в определенном состоянии (`Активные`, `Проблемные`, `На проверке`, `Недоступные`, `Занятые`).

Для каждого апстрима, помимо имени и загрузки зоны разделяемой памяти, представлены следующие данные:

`Сервер`	Имена, время простоя и веса серверов апстрима **Подсказка** Щелкните стрелку рядом с пунктом `Сервер`, чтобы отсортировать серверы по их состоянию или порядку в конфигурации.
`Запросы`	Общее количество и скорость обработки запросов
`Ответы`	Количество ответов с разбиением по кодам статуса
`Соединения`	Количество активных соединений и их максимальный предел, если он задан
`Трафик`	Скорость исходящего и входящего трафика, а также общие объемы исходящего и входящего трафика
`Проверки сервера`	Количество неуспешных обращений к серверу и число раз, когда сервер считался недоступным (объект `health` в API)
`Проверки работоспособности`	Общее количество проверок сервера, количество неуспешных проверок, а также время последней проверки

6.6.13 Редактирование апстримов

Рядом с каждым апстримом есть кнопка редактирования; при нажатии она выводит еще две кнопки:

`Редактировать выбранные`	Редактирование выбранных серверов в составе апстрима. Позволяет одновременно задать для всех следующие параметры: `Вес`, максимальный предел соединений (`Max_conns`), максимальный предел сбоев, переводящий сервер в недоступные (`Max_fails`), временное окно подсчета сбоев для максимального предела сбоев (`Fail_timeout`), состояние (`активный` – включен, `недоступный` – выключен или `разгружаемый` – получает только запросы сессий, привязанных ранее через `sticky`). Также здесь можно удалить выбранные серверы. 
`Добавить сервер`	Добавление сервера в апстрим. Позволяет задать следующие параметры: адрес, запасной сервер или нет, `Вес`, максимальный предел соединений (`Max_conns`), максимальный предел сбоев, переводящий сервер в недоступные (`Max_fails`), временное окно подсчета сбоев (`Fail_timeout`), состояние (`активный` – включен, `недоступный` – выключен или `разгружаемый` – получает только запросы сессий, привязанных ранее через `sticky`). 

Вкладка "TCP/UDP-зоны"

6.6.14 Раздел "TCP/UDP-зоны"

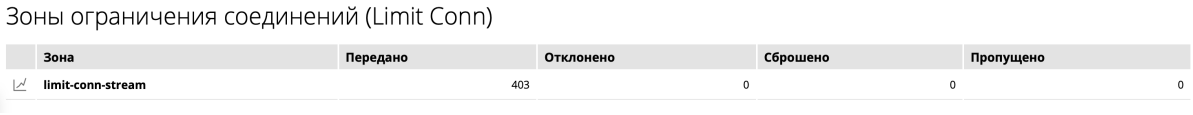
TCP/UDP-зоны

Зона	Соединения			Сессии				Трафик				SSL		
	Текущих	Всего	Соед./сек.	2xx	4xx	5xx	Всего	Отпр./сек.	Получ./сек.	Отправлено	Получено	Рукопожатий	Неудачных рукопожатий	Повторных использований
sing_chorus	3	403	0	372	0	28	400	0	0	4.41 ГБ	6.51 МБ	375	0	180

Здесь в сводном виде отображается статистика мониторинга зон разделяемой памяти контекста `server` в `stream`, формируемая на основе раздела API `/status/stream/server_zones/`. Для каждой зоны представлены следующие данные:

`Зона`	Имя зоны
`Соединения`	Текущее и общее количество соединений, а также число соединений в секунду
`Сессии`	Количество сессий с разбиением по кодам статуса, а также их общее число
`Трафик`	Скорость исходящего и входящего трафика, а также общие объемы исходящего и входящего трафика
`SSL`	Суммарные показатели количества: успешных SSL-рукопожатий; неуспешных SSL-рукопожатий; повторных использований SSL-сессий

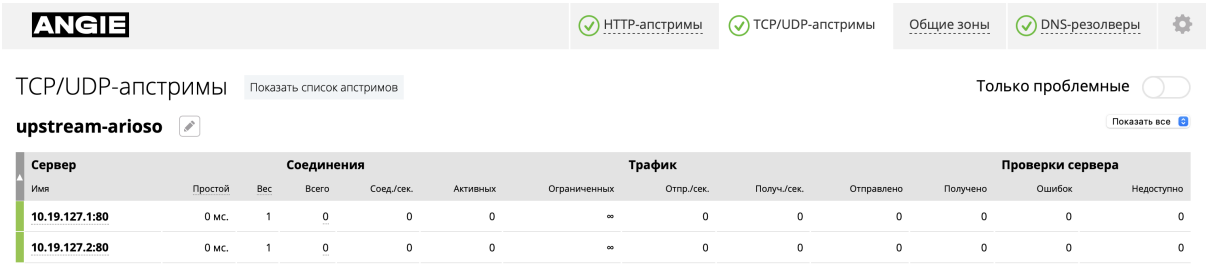
6.6.15 Раздел "Зоны ограничения соединений (Limit Conn)"



Здесь приведена статистика зон `limit_conn` в контексте `stream`, формируемая на основе раздела API `/status/stream/limit_conns/`. Для каждой зоны представлены следующие данные:

`Зона`	Имя зоны **Подсказка** Щелкните значок рядом с пунктом `Зона`, чтобы открыть или закрыть график со следующими показателями.
`Передано`	Общее количество проксированных соединений
`Отклонено`	Общее количество сброшенных соединений
`Сброшено`	Общее количество соединений, сброшенных из-за переполнения хранилища зоны
`Пропущено`	Общее количество соединений, переданных с нулевым или превосходящим 255 байт ключом

Вкладка "TCP/UDP-апстримы"



На этой вкладке в сводном виде отображается статистика мониторинга апстримов контекста `stream`, формируемая на основе раздела API `/status/stream/upstreams/`.

- Кнопка `Показать список апстримов` переключает отображение краткого списка апстримов с указанием числа проблемных апстримов и пиров.
- Переключатель `Только проблемные` включает и отключает режим вывода статистики по проблемным апстримам.

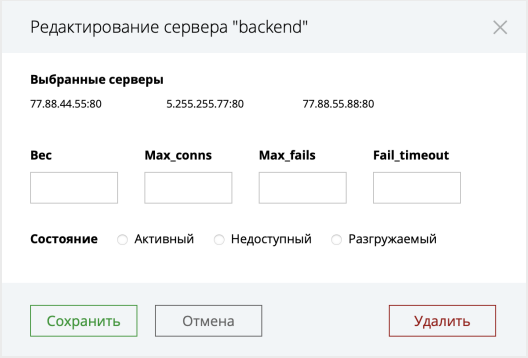
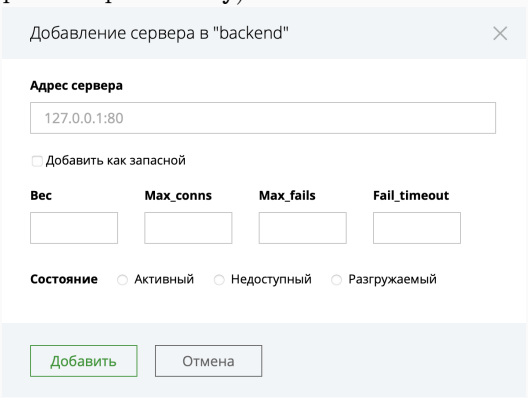
- Кнопка редактирования открывает виджет `редактирования апстрима`.
- Раскрывающийся список с правой стороны таблицы каждого апстрима позволяет отфильтровать серверы в определенном состоянии (`Активные`, `Проблемные`, `На проверке`, `Недоступные`, `Занятые`).

Для каждого апстрима представлены следующие данные:

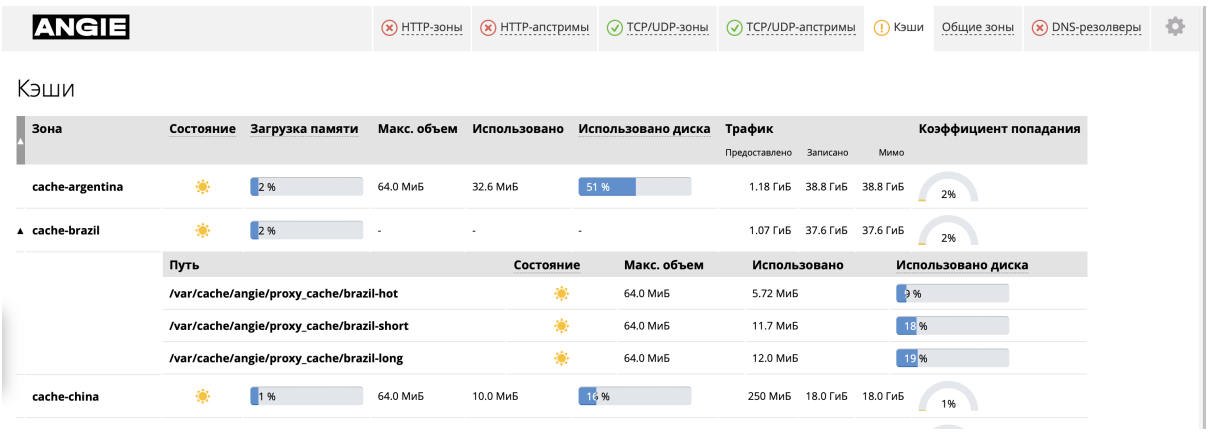
`Сервер`	Имена, время простоя и веса серверов апстрима **Подсказка** Щелкните стрелку рядом с пунктом `Сервер`, чтобы отсортировать серверы по их состоянию или порядку в конфигурации.
`Соединения`	Количество активных соединений и их максимальный предел, если он задан
`Трафик`	Скорость исходящего и входящего трафика, а также общие объемы исходящего и входящего трафика
`Проверки сервера`	Количество неуспешных обращений к серверу и число раз, когда сервер считался недоступным (объект `health` в API)

6.6.16 Редактирование апстримов

Рядом с каждым апстримом есть кнопка редактирования; при нажатии она выводит еще две кнопки:

`Редактировать выбранные`	Редактирование выбранных серверов в составе апстрима. Позволяет одновременно задать для всех следующие параметры: `Вес`, максимальный предел соединений (`Max_conns`), максимальный предел сбоев, переводящий сервер в недоступные (`Max_fails`), временное окно подсчета сбоев для максимального предела сбоев (`Fail_timeout`), состояние (`активный` – включен, `недоступный` – выключен или `разгружаемый` – получает только запросы сессий, привязанных ранее через `sticky`). Также здесь можно удалить выбранные серверы. 
Добавить сервер	Добавление сервера в апстрим. Позволяет задать следующие параметры: адрес, запасной сервер или нет, `Вес`, максимальный предел соединений (`Max_conns`), максимальный предел сбоев, переводящий сервер в недоступные (`Max_fails`), временное окно подсчета сбоев (`Fail_timeout`), состояние (`активный` – включен, `недоступный` – выключен или `разгружаемый` – получает только запросы сессий, привязанных ранее через `sticky`). 

Вкладка "Кэши"



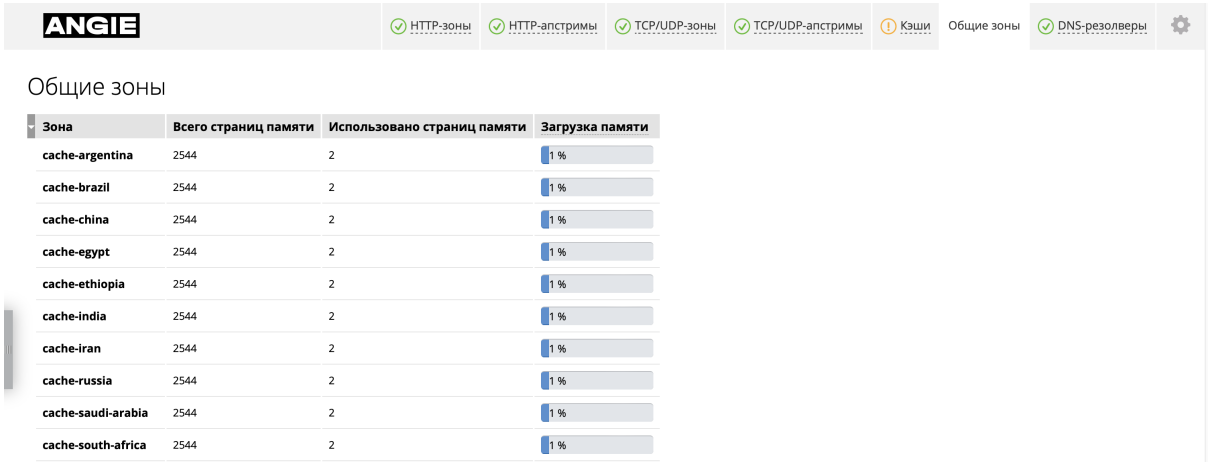
На этой вкладке в сводном виде отображается статистика мониторинга зон `proxy_cache` контекста `http`, формируемая на основе раздела API `/status/http/caches/`. Для каждой зоны представлены следующие данные:

`Зона`	Имя зоны **Подсказка** Щелкните значок рядом с пунктом `Зона`, чтобы открыть или закрыть списки `шардов` для всех зон, где они есть.
`Состояние`	Состояние кэша: холодный (метаданные загружаются в память) или горячий (метаданные загружены)
`Загрузка памяти`	Коэффициент использования памяти
`Макс. размер`	Максимальный объем памяти
`Использовано`	Использованный объем памяти
`Загрузка диска`	Коэффициент использования дисковой памяти
`Трафик`	Переданный из кэша, записанный в кэш и возвращенный в обход кэша трафик
`Коэффициент попадания`	Коэффициент попадания в кэш (отношение переданного из кэша трафика к общему объему)

Если для зоны включен *шардинг*, то она обозначена как раскрывающийся список, в котором перечислены отдельные шарды:

`Путь`	Путь шарда на диске
`Состояние`	Состояние шарда: холодный (метаданные загружаются в память) или горячий (метаданные загружены)
`Макс. размер`	Максимальный объем памяти
`Использовано`	Использованный объем памяти
`Загрузка диска`	Коэффициент использования дисковой памяти

Вкладка "Общие зоны"



На этой вкладке в сводном виде отображается статистика мониторинга **всех** зон разделяемой памяти для всех контекстов. Для каждой зоны представлены следующие данные:

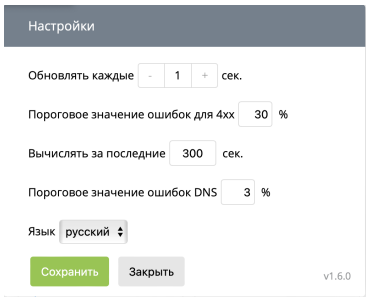
`Зона`	Имя зоны **Подсказка** Щелкните стрелку рядом с пунктом `Зона`, чтобы отсортировать зоны по размеру или порядку в конфигурации.
`Всего страниц памяти`	Общее количество страниц памяти
`Использовано страниц памяти`	Количество используемых страниц памяти
`Загрузка памяти`	Коэффициент использования памяти для зоны

Вкладка "DNS-резолверы"

На этой вкладке в сводном виде отображается статистика запросов в зонах разделяемой памяти DNS, формируемая на основе раздела API `/status/resolvers/`. Для каждой зоны представлены следующие данные:

`Зона`	Имя зоны **Подсказка** Щелкните стрелку рядом с пунктом `Зона`, чтобы отсортировать зоны по состоянию или порядку в конфигурации.
`Запросы`	Количество запросов типа A и AAAA, SRV, PTR
`Ответы`	Количество ответов с разделением по соответствующим кодам (`Успешные`, `Ошибок формата`, `Сервер не завершил запрос`, `Ошибок имени`, `Запрос не поддерживается`, `Запрос отклонен` и прочих)

Виджет "Настройки"



Позволяет настроить общие параметры консоли:

- Частоту обновления данных. Значение по умолчанию — 1 сек.

- Пороговое соотношение статусов `4xx`. По достижении порога в соответствующих разделах, посвященных ответам сервера, появляются "желтые" предупреждения. Значение по умолчанию — 7%.
- Временное окно для вычисления соотношения успешных попаданий в кэш. Значение по умолчанию — 300 сек.
- Порог учета ошибок для резолвера. По достижении указанного порога резолвер станет "красным". Значение по умолчанию — 3%.
- Язык интерфейса консоли. Доступные варианты: английский и русский. По умолчанию язык консоли выбирается на основе локали, установленной в браузере.

Панель управления консолью

На всех вкладках в середине левой части страницы есть выдвигающаяся панель с двумя кнопками  . Верхняя приостанавливает и вновь запускает обновление данных из API, а нижняя позволяет обновить данные вручную, когда обновление приостановлено.

ГЛАВА 7

Работа через Angie ADC CLI

Через командную строку Angie ADC CLI вы можете управлять разными функциями Angie ADC, например:

- *загружать сертификаты* для включения безопасного соединения;
- *настраивать маршрутизацию*.

Инструкцию по запуску Angie ADC CLI можно посмотреть *здесь*.

7.1 Запуск Angie ADC CLI

При запуске Angie ADC интерфейс CLI будет доступен по SSH-протоколу на порту 2022. Доступ к конфигурации осуществляется через интерфейс Angie ADC CLI с помощью утилиты `vtysh`.

Примечание

Если SSH-сервер запущен, то в журнале отобразится сообщение `starting ssh server on port <номер порта>`.

Чтобы запустить Angie ADC CLI по SSH-протоколу, выполните следующие действия:

1. Подключитесь к SSH-серверу. Например, для порта 2022 выполните следующую команду:

   ```
   ssh user@localhost -p 2022
   ```

2. После подключения сервер запросит пароль для авторизации. Введите пароль:

   ```
   user@localhost's password:
   #
   ```

   Если авторизация прошла успешно, откроется терминал Angie ADC CLI. Вам будет доступна справка по `?`.

3. Чтобы перейти в настройки SSH, выполните следующую команду:

```
vtysh
```

Команда переключит вас в конфигурационный режим, где можно вносить изменения в настройки SSH.

Примечание

Интерфейс может зависать при выходе по команде `exit`, в таком случае необходимо нажать `CTRL+C`, чтобы вернуться в рабочий режим.

ГЛАВА 8

Типовые задачи и примеры

В этом разделе собраны примеры настройки Angie ADC под разные задачи.

8.1 Настройка HTTPS

Вы можете загружать сертификаты (`crt`) и закрытые ключи (`key`) через Angie ADC CLI для аутентификации и шифрования данных.

После успешного добавления, загрузки и включения сертификата наряду с ключом в файле конфигурации Angie ADC `/etc/angie/http.d/default.conf` в блоке `http` добавится редирект:

```
listen       8080;
listen       [::]:8080;
server_name  localhost;


return 301 https://$host:8443$request_uri;
```

Также появится раздел конфигурации сервера с `https`.

8.1.1 Загрузка файлов TLS-сертификата

Чтобы загрузить файлы для TLS-сертификата, выполните следующие действия:

1. Подключитесь к Angie ADC CLI *по инструкции*.
2. Войдите в режим загрузки TLS-сертификатов, поочередно введя следующие команды:

```
cert-config
tls-upload
```

Отобразится приглашение:

```
(cert-config-tls-upload)#
```

3. Загрузите файл сертификата:

```
management crt
```

Будет выведено сообщение:

```
insert the certificate line by line at the end of the input, add an empty line␣
↪or enter end
```

В появившемся поле введите содержимое crt-файла.

Пример:

```
-----BEGIN CERTIFICATE-----
MIIDFDCCAfwCCQCq3AlavrbijjANBgkqhkiG9w0BAQsFADBMMQswCQYDVQQGEwJV␣
↪UzELMAkGA1UECAwCQ0ExFjAUBgNVBAcMDVNhbiBGcmFuY2lzY28xGDAWBgNVBAMM␣
↪D2FwcC5leGFtcGxlLmNvbTAeFw0yMDAzMjMyMzIwNDNaFw0yMzAzMjMyMzIwNDNa␣
↪MEwxCzAJBgNVBAYTAlVTMQswCQYDVQQIDAJDQTEWMBQGA1UEBwwNU2FuIEZyYW5j␣
↪aXNjbzEYMBYGA1UEAwwPYXBwLmV4YW1wbGUuY29tMIIBIjANBgkqhkiG9w0BAQEF␣
↪AAOCAQ8AMIIBCgKCAQEA2BExYGROFHh7eO1Yqx+VDs3G3+THrLFT/7DPQDBY3kzC /
↪hfZkX+w9mM6D5EOn+iiVSaQiP2mZ4P77oiGGfwrk62txD9pza8394/ZJ1yCGWgT␣
↪+cVPEYnLcBKsI4LrKIgmhYROR3sYdcGRdIrVPVe5ETBY5gU4DhaO6NPB+j+f+I+Z aXb/
↪2TAzBa4z31jHC86jePy1LvIFk1bcr6q+1DdyzIqqldD6/St8Ckwp9Nh1BPaa␣
↪KYgVUwMtQPbok5pQfmS+t884wR3GSLE8VLQo82bra5DwzhHjis99IDhsmKtSyV9s icImzytpgIyaM/
↪sXHQAOJmQInQmyh2zGuXXSCIdDkQIDAQABMA0GCSqGSIb3DQEB␣
↪CwUAA4IBAQCK1VHNgY9TvKioWoKovYBvsQ2f+raN8BpucCrtoFmy5G3pb3SiONwZ AvrxmJn/
↪GylkrJLpiAP5yCA4b6ciX2tFkzPFhITVJE5Ax9ihAvYfpMARuej3oG7e␣
↪wLpBMbRyFlrXWoMYEAOllWBnxtPAvXKf8IVFa4RH8sWRIHOx3hEv5mCuTf2SE84C␣
↪b6scKvw1oBANUXlWEVUa1fz/kYfAkYktvrWbTq6SXlhutIaf8XF3IC+/luox3e8L 0ApAPTNlgBp99/
↪qs+8oO1kaJd5NezNyIyxRuKI39CZLnBoNbi3vQXcSsD+XSiXOG pEgN3mr/
↪1Fk89VLHCaNy2+0j26tyjbrW
-----END CERTIFICATE-----
```

4. Загрузите ключ:

```
management key
```

Будет выведено сообщение:

```
insert the certificate line by line at the end of the input, add an empty line␣
↪or enter end
```

В появившемся поле введите содержимое key-файла.

Пример:

```
-----BEGIN PRIVATE KEY-----
MIIEvgIBADANBgkqhkiG9w0BAQEFAASCBKgwggSkAgEAAoIBAQDYETFgZE4UeHt47VirH5UOzcbf5MessVP/
↪sM9AMFjeTML+F9mRf7D2YzoPkQ6f6KJVJpCI/aZng/vuiIYZ/
↪CuTra3EP2nNrzf3j9knXIIZaBP5xU8RictwEqwjgusoiCaFhHRHexh1wZF0itU9V7kRMFjmBTgOFrTo08H6P5/
↪4j5lpdv/ZMDMFrjPfWMcLzqN4/
↪LUu8gWTVtyvqr7UN3LMiqqVOPr9K3wKTCnO2HUE9popiBVTAy1A9uiTmlB+ZL63zzjBHcZIsTxUtCjzZutrkPDOEeOKz3C
↪8+jWFc4gT8Ihe0FmGEIBYozXX/O0z0lWHxYx60ynYJ2qOoKQJOp9b9ljbQPdh/
↪f7a+EatG6MPYk4nqHF7kAsrclEz6He/
↪hLz6ddI2u7dLF0zBStB39X1Qg+gRsN+m9x5KQUMv12Koj7KshDzZ38naJ7yl82phAWYFg0NdysFPQnktZiMILNnQc7NyKt
↪a690A+jU8TRMmVrN/
↪FyXZLXkW9ocqV0Qm04L3+HLrlSBNTVK96SZUOm7V5bO216gxKgI+r0bndtNFMD/
↪1gp8lvRMqIHxfSyJ8HzlK5bOIgiJqDhs+pJY6f/+HW6uBFr7sFKyqmYHBP4H/
↪AtlOyKxEc0qWk1eOkB1cMHltX4pzkCgYEA2gp8CT5XZ3LIdP7c5HzCKV0s0XKXF6f2LsrXOVVZNbB7SHKSl90HSeVTlws7
↪rXV6hqtyGDi88i1YzKapAzuywoE7Nq1BiwfXIDPy9O5GF5qX5qTxCsIcHrj6gmW0FUAhhKYPp4qwRLw1LfBlwu5Vhn7r7z
↪+piz6KM2CR9aJESGfHpZwm1+5trEr0aX1j14ldqY9Jv0k3vqUk6khym8TRMfm8cxnFVGS31wIZLijf9igvIwJZBpEhI/
↪hO7zuAZbFahpGXLUBRPRrjSqCMHCU0pJVMkHeKB5XjqtO6nUQKBgCIPzTysbn8MoWAViHBxROutUJkPq6bYaE77BRBB0wH
↪/qrFNoUXrkqKixNN0qc+ywC9bOIZGqrTXnc8r3FGDVeeb9AikMM2kpda1Nhrgi/11ZaoYfTM/Bdk6/
↪HRAoGBAJqsNaYc16rUsc03PL2lWPcsFvqdb7HBrjERFHEw44VKv1Yq+FVbsM7QSAVuWYeplFAJCc7+HKub4lkXQ3TdwRj2
↪xxLdA76znZbXv9yqxgMUcMVpddnJLUfoPUVuu7KKeUU7M3HnTJQ+krWmmLkiIR
-----END PRIVATE KEY-----
```

5. Нажмите `Enter` или введите `end` и нажмите `Enter`, чтобы вернуться в `cert-config-tls-upload`.
6. Введите команду `exit`, чтобы перейти в `cert-config`.
7. Чтобы применить настройки `tls-management`, введите команду:

   ```
   enable management
   ```

   Файлы будут применены.

8.1.2 Загрузка файлов TLS-трафика

Для загрузки сертификатов и ключей TLS-трафика процесс аналогичен:

- Сертификат:

  ```
  traffic crt
  ```

- Ключ:

  ```
  traffic key
  ```

8.1.3 Проверка загруженных данных

После загрузки файлы будут сохранены по следующим путям:

- TLS-сертификат:

  ```
  /etc/angie/crt/cp.crt  # (crt)
  /etc/angie/crt/cp.key  # (key)
  ```

- TLS-трафик:

  ```
  /etc/angie-lb/crt/cp.crt # (crt)
  /etc/angie-lb/crt/cp.key # (key)
  ```

8.2 Настройка однорукого режима (one-armed mode)

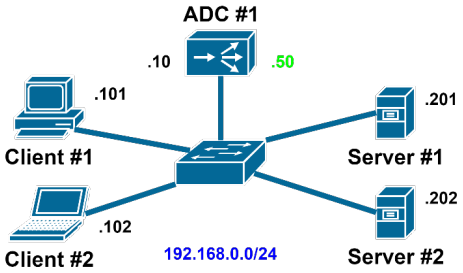
8.2.1 Введение

Однорукий режим (one-armed mode, одноинтерфейсный режим) — это способ балансировки нагрузки, при котором весь входящий и исходящий трафик (между клиентами и системой балансировки, между системой балансировки и upstream-серверами) проходит через один логический интерфейс. В отличие от режима работы с двумя интерфейсами (two-armed mode, двурукий режим), в одноруком режиме балансировщик работает как промежуточное звено, используя один интерфейс для передачи данных в обоих направлениях.

Этот режим подходит для использования в виртуализованных средах и облачных решениях, где может быть сложно разделить сетевые интерфейсы балансировщика: он проще настраивается, не требует существенных изменений в сети и может использоваться в уже существующей инфраструктуре. Также этот режим подходит для сценариев с SSL-терминацией или кэшированием. К минусам можно отнести повышенную нагрузку на интерфейс балансировщика и возможность проблем с асимметричной маршрутизацией (если отключить SNAT).

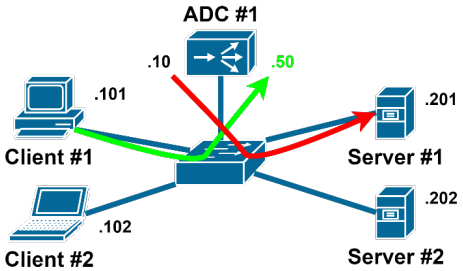
8.2.2 Схема работы

Типовая схема для однорукого режима представлена ниже:

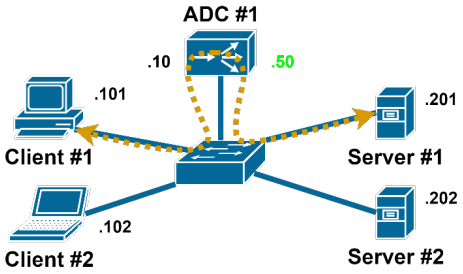


Здесь клиенты, серверы и сама система балансировки располагаются в одной подсети. IP-адреса виртуальных серверов, трафик которых балансируется, должны принадлежать этой же подсети. На этой схеме все устройства используют адреса из подсети `192.168.0.0/24`. Значение последнего октета IP-адреса устройства представлено на схеме. Зеленым указан адрес виртуального сервера. Для упрощения будем рассматривать только один виртуальный сервер с одним IP-адресом.

Клиент подключается к ресурсу по адресу виртуального сервера (в данном случае это `192.168.0.50`). Система балансировки с помощью настроенного метода балансировки выбирает сервер, к которому подключается. Для примера будем считать, что подключение инициирует устройство Client #1, а нагрузка была распределена на сервер Server #1. В итоге мы получаем две независимые TCP-сессии: первая сессия устанавливается между адресами `192.168.0.101` и `192.168.0.50`, а вторая – между адресами `192.168.0.10` и `192.168.0.201`. Система балансировки «перекладывает» данные из одной TCP-сессии в другую:



При такой схеме соединений сервер не может точно установить по IP-адресу отправителя пакета, какой клиент к нему подключается, то есть, по сути, выполняется SNAT, поэтому обратный трафик от сервера возвращается в систему балансировки, а не напрямую клиенту. Опция SNAT отключаемая, однако в данном случае ее отключение приведет к тому, что обратный трафик (от сервера к клиенту) будет передаваться напрямую, что приведет к невозможности установления соединения, так как Angie ADC не поддерживает DSR (Direct Server Response).

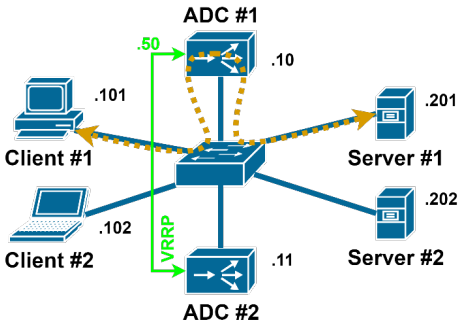


Angie ADC может передать серверу информацию об IP-адресе клиента двумя способами:

- с помощью XFF-заголовка (X-Forwarded-For) для протокола HTTP;
- с помощью протокола PROXY.

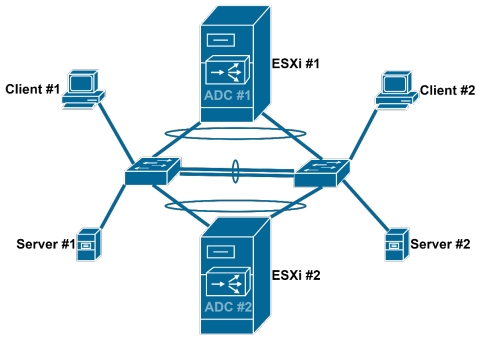
8.2.3 Обеспечение отказоустойчивости

Отказоустойчивость можно обеспечить несколькими стандартными способами: с помощью протокола VRRP, либо путем использования протоколов динамической маршрутизации OSPF или BGP. Для типовой схемы выше можно применять резервирование с помощью протокола VRRP. Адреса виртуальных серверов в этом случае назначаются VRRP-интерфейсу:



Обратный трафик (от сервера к клиенту) возвращается через ту же систему балансировки, через которую прошел трафик в прямом направлении. Это достигается за счет применения SNAT.

При применении Angie ADC в критически важных сегментах сети можно зарезервировать не только всю систему балансировки целиком, но и физические интерфейсы, которыми она подключается к сети. Такое резервирование позволит не только повысить доступность системы, но также и увеличит производительность сетевой инфраструктуры. однорукий режим работы подразумевает использование одного логического интерфейса в data-plane, тогда как физических интерфейсов, входящих в состав этого логического интерфейса, может быть несколько. Физические интерфейсы в этом случае объединяются в группу (Bundle, LAG, Port-Channel, EtherChannel). IP-адрес назначается на такой логический интерфейс:

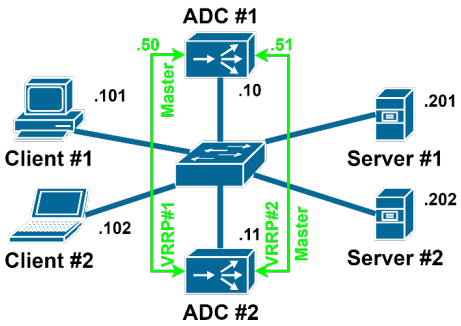


Для обеспечения максимальной доступности в такой сети следует также зарезервировать коммутатор по любой доступной технологии: стек, виртуальный стек, VPC, VSS, mLAG, VLT и т. п.

8.2.4 Масштабирование

Вы можете горизонтально масштабировать систему балансировки Angie ADC, работающую в одноруком режиме, организовав работу нескольких Angie ADC в режиме псевдо active-active с помощью применения multigroup VRRP. Суть такого подхода состоит в том, что между системами балансировки Angie ADC поднимается несколько VRRP-групп, каждая из которых имеет собственный уникальный набор адресов виртуальных серверов. Для одной группы роль мастера получает первая система балансировки, тогда как для второй группы роль мастера получает вторая система.

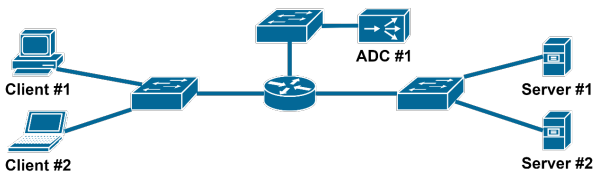
На схеме ниже представлены две VRRP-группы (1 и 2). Первой группе назначен виртуальный адрес `192.168.0.50`, тогда как второй – `192.168.0.51`.



8.2.5 Прочие схемы

Обсуждаемая схема — не единственная с точки зрения взаимного расположения клиентов, серверов и систем балансировки. Серверы и клиенты могут располагаться как в одном сетевом сегменте с системой балансировки, так и в разных.

На схеме ниже представлен пример сети, когда клиенты и серверы расположены в отдельных сегментах:



8.2.6 Заключение

При выборе режима работы системы балансировки (однорукий, one-armed или двурукий, two-armed) следует опираться на особенности дизайна сети, требования отдела информационной безопасности, а также прочие внутренние правила и регламенты. Angie ADC поддерживает оба режима.

8.3 Настройка IPv6

IPv6 в Angie ADC настраивается на трех уровнях:

- Настройка доступа к консоли Angie ADC (management plane).
- Настройка сетевых протоколов маршрутизации (control plane).
- Настройки для передачи и обработки клиентского трафика (data plane).

8.3.1 Доступ к консоли Angie ADC

Консоль Angie ADC одновременно поддерживает IPv4 и с IPv6. Для доступа к консоли с IPv6-адресом `2001:db8::10` введите в браузере:

- `http://[2001:db8::10]:8080`
- `https://[2001:db8::10]:8443` (если требуется безопасное подключение).

Если есть DNS-запись для адреса `2001:db8::10`, можно обращаться к консоли по имени.

8.3.2 Настройка сетевых протоколов маршрутизации

Для маршрутизации IPv6 подходят все стандартные и вспомогательные протоколы маршрутизации:

- BGPv4;
- OSPFv3 (для IPv6 только OSPFv3, в отличие от IPv4);
- BFD;
- VRRPv3.

Примечание

Сосед для протокола BGP и пир для BFD указываются в формате IPv6-адреса.

Пример настройки OSPFv3:

```
router ospf6
  ospf6 router-id 2.2.2.2
  log-adjacency-changes
  redistribute connected
exit
```

Также необходимо включить OSPFv3 на всех интерфейсах:

```
interface ens33
  description internal
  ipv6 address 2001:db8::2/64
```

```
  ipv6 ospf6 area 0
exit
```

8.3.3 Передача и обработка клиентского трафика

Для поддержки IPv6:

- Бэкенд-серверы должны быть настроены на прием подключений по протоколу IPv6.
- В конфигурации балансировщика нагрузки необходимо включить прием трафика по IPv6:

  ```
  server {
      listen 80;
      listen [::]:80;
      location / {
  ```

  После этого балансировщик сможет принимать подключения по IPv6.
- Также необходимо добавить AAAA-записи в DNS, чтобы клиенты могли находить балансировщик по IPv6-адресу.
- Для маршрутизации трафика к бэкенд-серверам в секции `upstream` должны быть указаны IPv6-адреса.

  В примере ниже один сервер задан адресом IPv4, а второй – IPv6:

  ```
  http {
      upstream myapp1 {
          server 192.168.0.128:80;
          server [2001:db8::4]:80;
      }
  }
  ```

8.3.4 Поддержка смешанных подключений IPv4 и IPv6 в Angie ADC

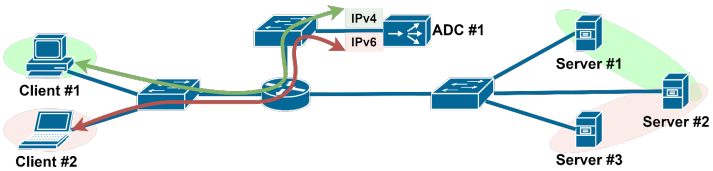
Angie ADC поддерживает гибкую маршрутизацию трафика между клиентами и бэкенд-серверами с разными конфигурациями.

Бэкенд-серверы могут быть трех типов:

- только IPv4;
- только IPv6;
- dual-stack (одновременно поддерживаются оба протокола).

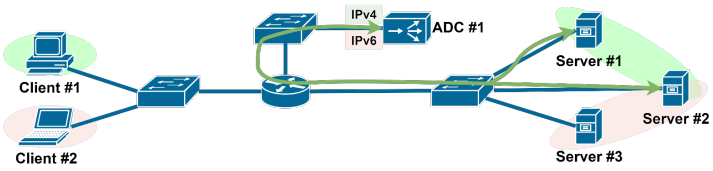
Пример:

К некоторому ресурсу подключаются IPv4-клиенты (отмечены зеленым цветом) и IPv6-клиенты (отмечены красным цветом).



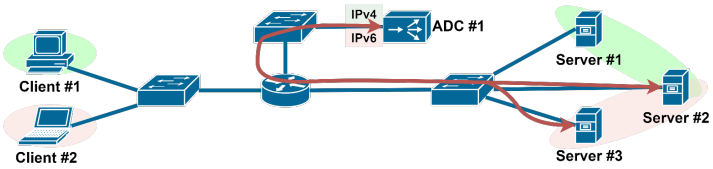
Только IPv4

IPv4-сессии установлены между Angie ADC и бэкенд-серверами с поддержкой IPv4:



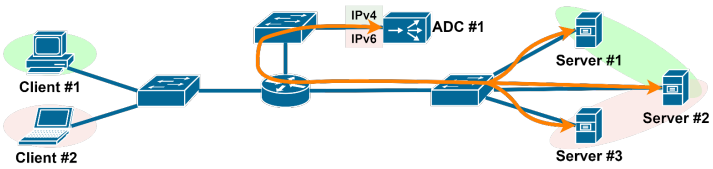
Только IPv6

IPv6-сессии установлены между Angie ADC и бэкенд-серверами с поддержкой IPv6:



Смешанные подключения

Angie ADC может проксировать входящие IPv4-подключения в любые сессии (как IPv4, так и IPv6) в сторону бэкенд-серверов. Аналогично и для IPv6-подключений.



Таким образом, Angie ADC позволяет гибко проксировать трафик между различными протоколами.

Преимущества такой схемы:

- Практически мгновенная публикация в сети IPv6-сервиса без поддержки IPv6 на бэкенд-серверах.
- Постепенный переход бэкенд-инфраструктуры на IPv6 без потери доступности сервисов снаружи по IPv4, т.е. возможна миграция с IPv4 на IPv6 с минимальными изменениями инфраструктуры.

8.3.5 Настройка ip6tables

Правила управления доступом для IPv6 похожи на правила для IPv4, но должны настраиваться отдельно вручную.

Пример типовой настройки `iptables`:

```
[root@angie-va angie-va]# iptables -S
-P INPUT DROP
-P FORWARD DROP
-P OUTPUT ACCEPT
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 80 -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 8080 -j ACCEPT
```

```
-A INPUT -p tcp -m state --state NEW -m tcp --dport 443 -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 53 -j ACCEPT
-A INPUT -p udp -m state --state NEW -m udp --dport 53 -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 2022 -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 179 -j ACCEPT
-A INPUT -p ospf -j ACCEPT
-A INPUT -p udp -m state --state NEW -m udp --dport 520 -j ACCEPT
-A INPUT -p eigrp -j ACCEPT
-A INPUT -j REJECT --reject-with icmp-host-prohibited
```

Для IPv6 соответствующая настройка производится с помощью `ip6tables`:

```
[root@angie-va angie-va]# ip6tables -S
-P INPUT DROP
-P FORWARD DROP
-P OUTPUT ACCEPT
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 80 -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 8080 -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 443 -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 53 -j ACCEPT
-A INPUT -p udp -m state --state NEW -m udp --dport 53 -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 2022 -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 179 -j ACCEPT
-A INPUT -p ipv6-icmp -j ACCEPT
-A INPUT -p ospf -j ACCEPT
-A INPUT -p udp -m state --state NEW -m udp --dport 521 -j ACCEPT
-A INPUT -j REJECT --reject-with icmp6-adm-prohibited
```

8.3.6 GSLB

К сервису GSLB можно обращаться по протоколу IPv6, но получить AAAA-записи пока невозможно. Функциональность в разработке.

```
C:>nslookup www.example.com 2001:db8::11
Server:  gslb.example.com
Address:  2001:db8::11

Name:    www.example.com
Address:  172.20.0.10
```

8.4 Настройка ECMP

В этой статье рассматривается:

- ECMP-распределение трафика со стороны сетевого оборудования между несколькими узлами Angie ADC;
- распределение трафика самой системой балансировки Angie ADC между несколькими путями в сторону клиентов или серверов.

В конце также будет рассмотрена возможность UCMP-балансировки.

ECMP (Equal-Cost Multi-Path) – это механизм, который позволяет равномерно распределять трафик между несколькими путями, если они имеют одинаковую стоимость (метрику). В контексте

Angie ADC это означает, что входящий трафик может приходить на разные узлы Angie ADC, а также сама система балансировки Angie ADC может отправлять трафик через разные пути с одинаковой стоимостью.

Такой подход может использоваться:

- Если необходимо распределять трафик в сторону серверов или клиентов между несколькими доступными путями.
- При горизонтальном масштабировании Angie ADC, когда трафик распределяется между несколькими узлами Angie ADC с помощью сетевого оборудования, а затем узлы Angie ADC распределяют пользовательские сессии между конечными серверами.

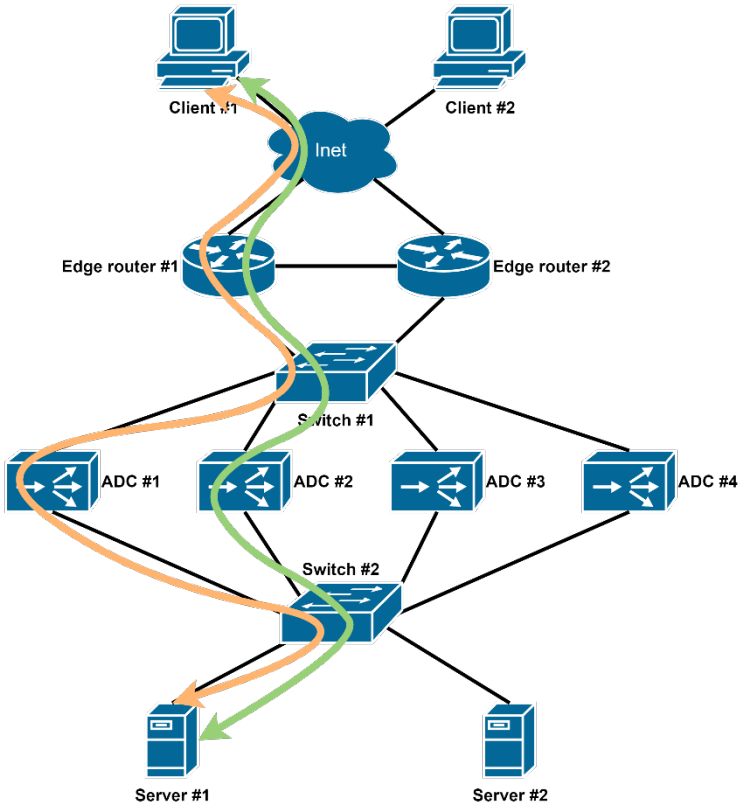
Angie ADC поддерживает ECMP-балансировку между 64 путями.

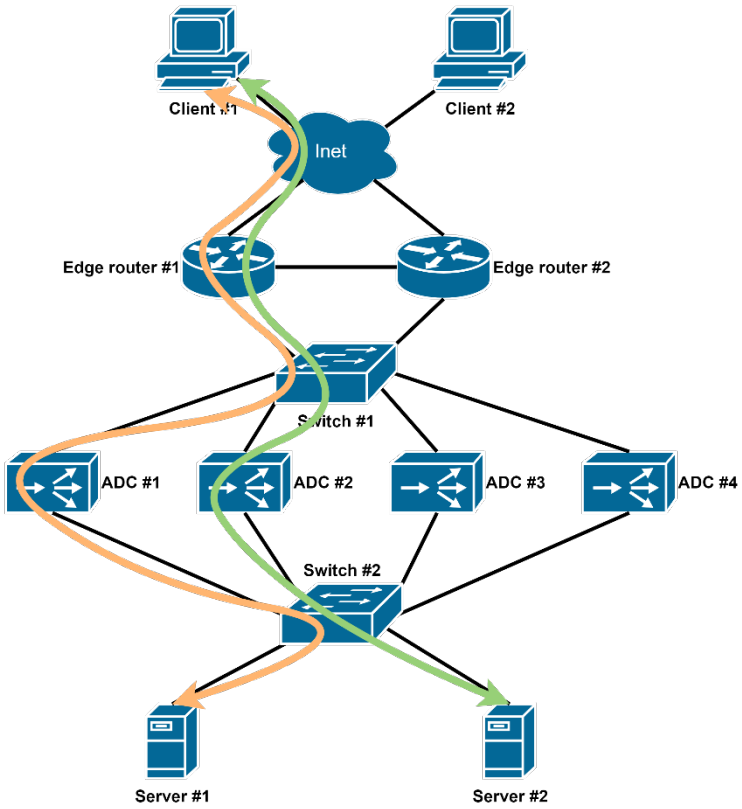
8.4.1 Распределение трафика между несколькими Angie ADC

Чтобы трафик успешно передавался в Angie ADC по нескольким путям с одинаковой стоимостью, узлы Angie ADC должны «видеть» всю сессию полностью. То есть, весь прямой трафик в рамках одной сессии (первый поток) должен попадать на одну систему балансировки. Обратный трафик также должен возвращаться на нее, благодаря использованию адресов или пулов SNAT.

Трафик второй сессии (второй поток) может распределиться на другую систему балансировки, но будет полностью обрабатываться в ней.

Пример:





Первая сессия показана оранжевой стрелкой, вторая – зеленой. Второй Angie ADC может выбрать тот же или другой бэкенд-сервер для обработки пользовательского запроса.

8.4.2 Внешнее хранилище sticky

Когда критичным является распределение всех сессий от одного клиента на один бэкенд-сервер, используется механизм sticky. Angie ADC запоминает, куда было отправлено первое соединение от клиента, и все последующие подключения отправляет на этот же сервер. Если вторая сессия обрабатывается другим Angie ADC, который не обладает информацией о sticky, то для sticky можно использовать внешнее хранилище информации. В качестве такого хранилища может выступать используемый или выделенный узел Angie ADC.

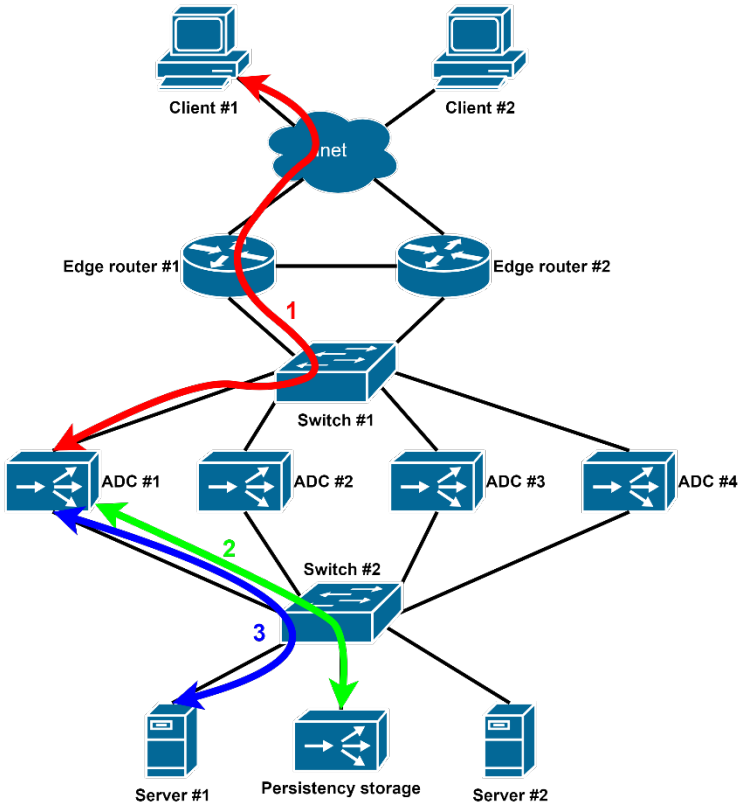
Алгоритм использования внешнего хранилища для sticky будет таким:

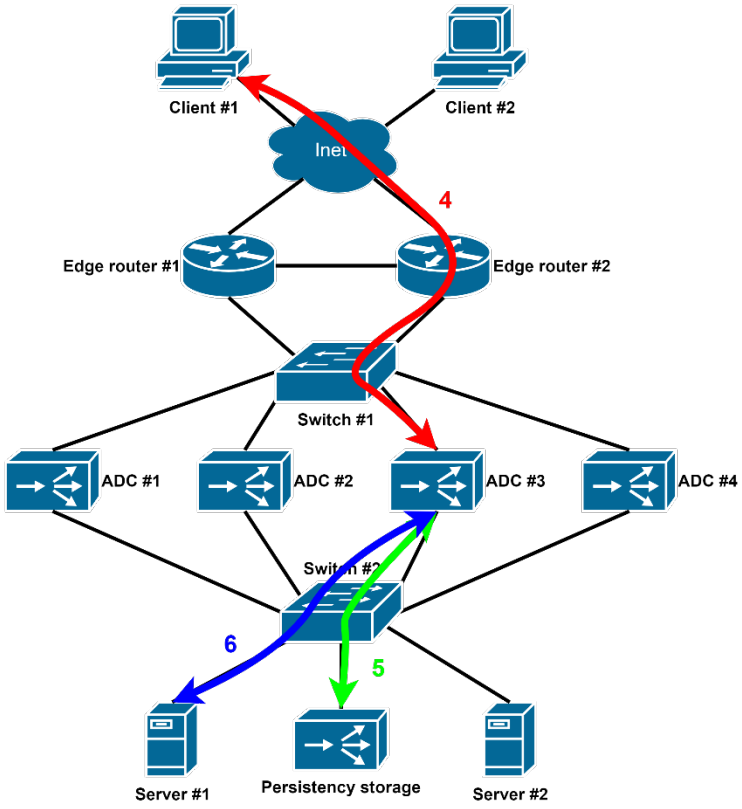
1. Пограничный маршрутизатор распределяет входящее подключение на один из узлов Angie ADC.
2. Получив пользовательский запрос, Angie ADC #1 выбирает сервер для перенаправления запроса (в соответствии с настроенным методом балансировки) и отправляет сообщение в хранилище sticky. Так как это первое подключение со стороны клиента, Angie ADC #1 сохраняет sticky в хранилище и подтверждает правильность распределения.
3. Angie ADC #1 устанавливает вторую часть подключения (до бэкенд-сервера). Теперь пользовательское соединение полностью установлено и готово к работе.
4. Клиент устанавливает вторую сессию, которая попадает на Angie ADC #3.
5. Angie ADC #3 выбирает сервер для перенаправления запроса (в соответствии с настроенным методом балансировки) и отправляет сообщение в хранилище sticky. Если сервер, на который предлагается распределить нагрузку, совпадает с тем, который записан в хранилище, то хранилище подтверждает выбор системы балансировки. Если предложенный сервер не совпадает с записанным, то хранилище возвращает хэш-сумму «правильного» сервера.

6. Система балансировки Angie ADC #3 установит новую сессию в соответствии с ответом, полученным от хранилища.

Таким образом, клиент всегда будет устанавливать соединения с одним и тем же бэкенд-сервером.

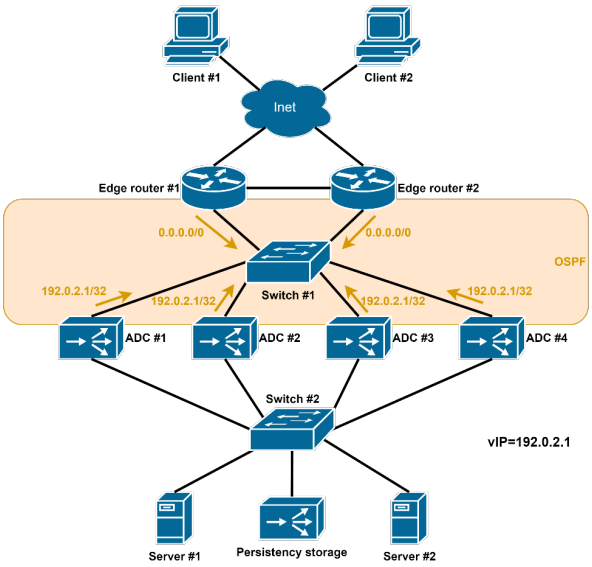
Пример (стрелки пронумерованы в соответствии с шагами выше):





8.4.3 Распределение трафика между несколькими маршрутизаторами

Ниже приведен пример небольшой корпоративной сети, где из соображений отказоустойчивости или для преодоления ограничений производительности граничных маршрутизаторов реализована схема с двумя независимыми каналами в интернет, которые терминируются на двух независимых маршрутизаторах.



Каждая из систем балансировки Angie ADC получает маршрут по умолчанию от каждого из граничных маршрутизаторов. Если параметры этих маршрутов одинаковы (для OSPF это типы маршрутов и метрики), то Angie ADC будет равномерно распределять трафик между граничными маршрутизаторами, то есть будет выполнять ECMP-балансировку.

Ниже приведен пример двух маршрутов по умолчанию, полученных с помощью протокола OSPF:

```
angie-adc1# sho ip ro os`
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`
O[1]>* 0.0.0.0/0 [110/1] via 192.168.0.201, ens33, weight 1, 00:00:32`
* via 192.168.0.202, ens33, weight 1, 00:00:32`
```

В операционной системе этот же маршрут виден так:

```
[root@angie-adc1 angie-adc1]# ip r
default nhid 23 proto ospf metric 20
nexthop via 192.168.0.202 dev ens33 weight 1
nexthop via 192.168.0.201 dev ens33 weight 1
```

8.4.4 UCMP-балансировка

Некоторые протоколы динамической маршрутизации (например, EIGRP, BGP, IS-IS) поддерживают балансировку по путям с разной стоимостью – UCMP (Unequal Cost Multi-Path). В примере ниже приведен сценарий, реализованный с помощью BGP с использованием Extended Community Bandwidth:

```
angie-adc1# sho ip bgp
BGP table version is 23, local router ID is 192.168.0.153, 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
*> 192.0.2.1/32 192.168.0.201 0 0 65002 i
*=   192.168.0.202 0 0 65002 i
Displayed 1 routes and 2 total paths
angie-adc1# sho ip bg 192.0.2.1/32
BGP routing table entry for 192.0.2.1/32, version 23
Paths: (2 available, best #1, table default)
Advertised to non peer-group peers:
192.168.0.201 192.168.0.201
2
192.168.0.202 from 192.168.0.202 (192.168.0.202)
Origin IGP, metric 0, valid, external, multipath, bestpath-from-AS 2, best (Older␣
↪Path)
Extended Community: LB:1:12500000 (100.000 Mbps)
Last update: Fri Feb 28 17:59:16 2025
2
192.168.0.201 from 192.168.0.201 (192.168.0.201)
Origin IGP, metric 0, valid, external, multipath
Extended Community: LB:1:6250000 (50.000 Mbps)
Last update: Fri Feb 28 17:59:16 2025
```

Пример тестировался на RedOS версии 7.3. На данный момент UCMP не поддерживается этой операционной системой, то есть со стороны data-plane могут быть два исхода:

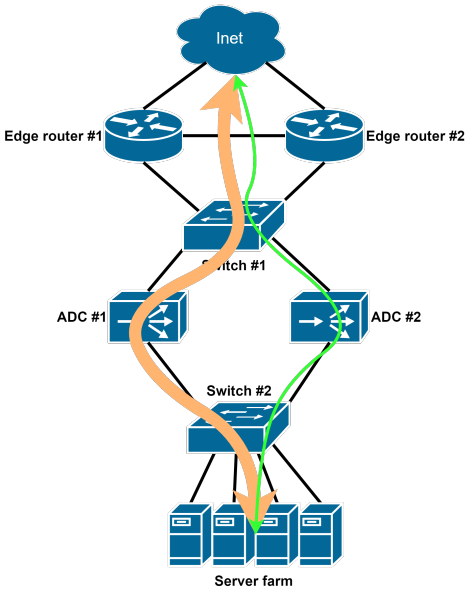
- маршрут будет установлен в RIB с балансировкой ECMP;
- маршрут не будет установлен в RIB, что зависит от других настроек протокола BGP.

```
angie-adc1# 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
K>* 0.0.0.0/0 [0/100] via 192.168.0.2, ens33, src 192.168.0.153, 01:30:36
B>* 192.0.2.1/32 [20/0] via 192.168.0.201, ens33, weight 1, 00:42:48
*                       via 192.168.0.202, ens33, weight 1, 00:42:48
C>* 192.168.0.0/24 is directly connected, ens33, 01:30:36
L>* 192.168.0.153/32 is directly connected, ens33, 01:30:36
```

Если маршрут устанавливается в RIB, то со стороны операционной системы таблица маршрутизации будет выглядеть так:

```
[root@angie-adc1 etc]# ip r
default via 192.168.0.2 dev ens33 proto dhcp src 192.168.0.153 metric 100
192.0.2.1 nhid 37 proto bgp metric 20
nexthop via 192.168.0.202 dev ens33 weight 1
nexthop via 192.168.0.201 dev ens33 weight 1
192.168.0.0/24 dev ens33 proto kernel scope link src 192.168.0.153 metric 100
```

На данный момент Angie ADC не поддерживает UCMP-балансировку на уровне передачи данных (data-plane), только ECMP. Однако, если сетевая инфраструктура поддерживает UCMP, то такая схема распределения трафика может использоваться. UCMP-балансировка в таком случае может быть полезной, например, когда есть значительный перекос в мощностях систем балансировки, участвующих в обработке трафика:



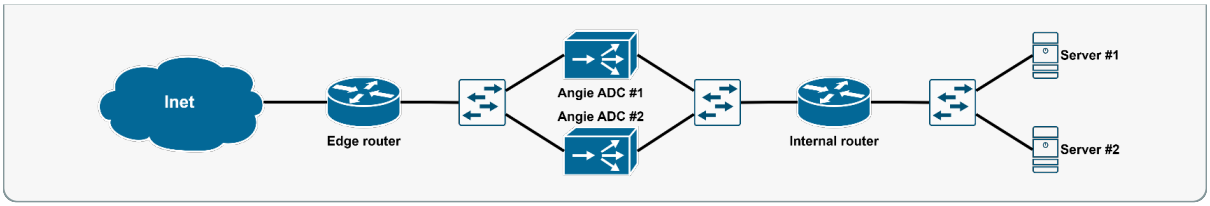
8.5 Настройка пула SNAT (SNAT Pool)

8.5.1 Введение

Пул SNAT (SNAT Pool, Source Network Address Translation Pool) — это набор IP-адресов, которые Angie ADC использует для подмены исходных IP-адресов клиентов при передаче трафика на серверы. SNAT-пулы применяются в высоконагруженных средах с большим количеством одновременно открытых сессий. Они позволяют обойти ограничение на количество одновременных подключений (около 65000 на один IP-адрес). Ограничение связано с количеством доступных клиентских сокетов (TCP/UDP-портов), используемых для установки соединений. SNAT-пулы могут использоваться как для L4-, так и для L7-балансировки.

Примечание

Также возможен подход, основанный на горизонтальном масштабировании за счет установки нескольких Angie ADC и распределения трафика между ними. Этот подход позволяет решить проблему не только с ограничением количества сессий, но и снять ограничение на производительность системы, например, когда при больших объемах трафика необходимо обрабатывать данные на скоростях более 10 Гбит/с.

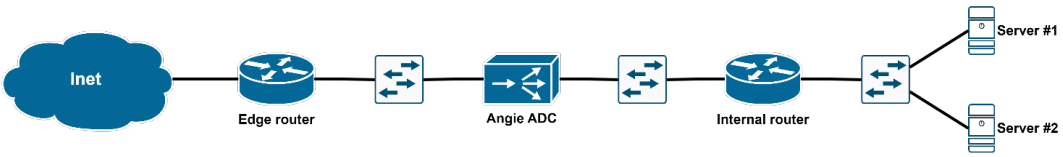


8.5.2 Ручная настройка SNAT-пулов

Настройка вручную создаваемых SNAT-пулов происходит в три этапа:

1. Выбор пула адресов и их настройка.
2. Настройка маршрутизации.
3. Настройка правил балансировки.

Пример простой схемы с одним Angie ADC представлен на рисунке ниже:



1. Выбор пула адресов и их настройка

Размер пула адресов напрямую зависит от количества сессий, которые требуется одновременно поддерживать. Чем больше сессий должно работать одновременно, тем больше IP-адресов должно входить в пул.

Выбор адресов условно свободный, можно использовать адреса из разных подсетей. Мы не рекомендуем использовать чужие валидные адреса, чтобы избежать случайных пересечений с адресами клиентов. Также лучше выбирать адреса не из произвольного диапазона, а выделить конкретную подсеть, которая будет полностью использоваться в пуле. Это упростит дальнейшую настройку маршрутизации.

В примере ниже мы будем использовать подсеть `192.168.12.20/30`, включающую в себя четыре адреса. Так как все четыре адреса будут использоваться в качестве адресов отправителя пакетов на участке сети между Angie ADC и апстримами, то необходимо все эти четыре адреса назначить на сетевой интерфейс системы, например, на loopback.

Пример конфигурации представлен ниже. Адреса назначаются с маской `/32`. Это нужно, чтобы система обрабатывала возвращающийся от серверов трафик и устанавливала полноценные соединения.

```
angie-va#
angie-va# conf t
angie-va(config)# int lo
angie-va(config-if)# ip add 192.168.12.20/32
angie-va(config-if)# ip add 192.168.12.21/32
angie-va(config-if)# ip add 192.168.12.22/32
angie-va(config-if)# ip add 192.168.12.23/32
```

Примечание

В этом примере между апстримами и Angie ADC размещен маршрутизатор. Ситуация, когда апстримы и один из интерфейсов Angie ADC размещены в одной подсети, будет рассмотрена отдельно (см. Заключение ниже).

2. Настройка маршрутизации

В этом примере мы будем использовать протокол BGP для динамической маршрутизации. Настройка других протоколов динамической маршрутизации выполняется аналогично.

В BGP можно анонсировать все четыре адреса из пула независимо (четыре префикса с маской `/32`). Однако BGP позволяет производить суммаризацию (агрегирование) анонсируемых префиксов, поэтому существует более оптимальный способ — выполнить анонс всего пула целиком в виде одного префикса с маской `/30`.

Создадим агрегированный маршрут для всей подсети, используемой в качестве пула SNAT:

```
angie-va# conf t
angie-va(config)# ip route 192.168.12.20/30 Null0
angie-va(config)# exi
angie-va# sho ip ro sta
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
S>* 192.168.12.20/30 [1/0] unreachable (blackhole), weight 1, 00:01:21
```

Получившийся маршрут необходимо анонсировать по протоколу BGP:

```
angie-va# conf t
angie-va(config)# router bg 1
angie-va(config-router)# add ipv4 un
angie-va(config-router-af)# red sta
angie-va(config-router-af)# end
angie-va#
```

Настройку BGP-соседей опустим для краткости (подробнее о настройке BGP см. *HA-решение в режиме резервирования с помощью протокола BGP*).

Примечание

На представленной выше схеме BGP-соседом должен стать маршрутизатор Internal router, так как трафик от серверов будет возвращаться через него.

Убедимся, что BGP-соседи получают корректный маршрут:

```
internal_router#sho ip bg
BGP table version is 6, local router ID is 192.168.0.150
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 192.168.12.20/30 192.168.0.147 0 100 0 ?
internal_router#
```

3. Настройка правил балансировки

Для распределения клиентов по адресам из пула можно использовать различные модули, например, Geo, Geo IP, Map, Split Clients. Рассмотрим несколько типовых сценариев с описанием используемых модулей.

Сценарий A: использование модуля Split Clients

Стандартная ситуация, когда на Angie ADC попадают подключения пользователей из интернета напрямую. Предполагается, что пользователи распределяются равномерно по всему пространству адресов. В этом случае для настройки можно использовать модуль Split Clients (применяется в контексте `http`). `Split_clients` работает как при L7-, так и при L4-балансировке.

Пример конфигурации:

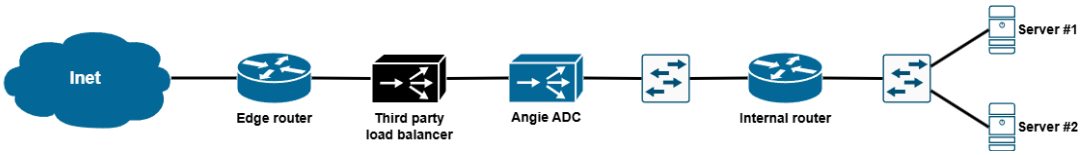
```
split_clients "${remote_addr}AAA" $variant {
25%               192.168.12.20;
25%               192.168.12.21;
25%               192.168.12.22;
25%               192.168.12.23;
}
```

Примечание

К адресу клиента добавлен текст `AAA`. Это нужно для изменения значения, выдаваемого хэш-функцией. Если потребуется поменять распределение клиентов по адресам из пула, можно поменять добавляемый текст.

Сценарий B: использование модуля Map

Возможна ситуация, когда перед Angie ADC с точки зрения трафика, идущего от клиентов, расположен другой балансировщик:



Если этот балансировщик использует SNAT-пул, то переменная `remote_addr` может не обеспечить достаточного разнообразия. Это может привести к коллизиям — перекосу в распределении клиентов по адресам из пула SNAT. Решить указанную проблему можно, например, используя модуль Map, с помощью которого можно в явном виде задать соответствие «старого» и «нового» адресов отправителя. Метод `map` работает как при L7-, так и при L4-балансировке.

Пример конфигурации (восемь адресов из предыдущего пула привязываются к четырем адресам пула SNAT):

```
map $remote_addr $variant {
"10.10.10.0"       "192.168.12.20";
"10.10.10.1"       "192.168.12.20";
"10.10.10.2"       "192.168.12.21";
"10.10.10.3"       "192.168.12.21";
```

```
"10.10.10.4"       "192.168.12.22";
"10.10.10.5"       "192.168.12.22";
"10.10.10.6"       "192.168.12.23";
"10.10.10.7"       "192.168.12.23";
}
```

Сценарий C: модуль Split Clients и хеш переменной $request_id

Для балансировки HTTP-трафика можно использовать подход, при котором не важно, используется ли какой-либо NAT/PAT или прокси между реальным клиентом и Angie ADC. Для распределения клиентов будет использоваться модуль Split Clients и хеш от переменной `$request_id`. Значение переменной генерируется автоматически и, по сути, является случайным числом. Этот метод имеет ограниченную область применения — L7-балансировка для протокола HTTP.

Пример конфигурации:

```
split_clients "${request_id}" $variant {
25% 192.168.12.20;
25% 192.168.12.21;
25% 192.168.12.22;
25% 192.168.12.23;
}
```

Примечание

При использовании этого метода разные сессии одного и того же клиента могут привязываться к различным адресам из пула. Это не должно быть большой проблемой, так как современные веб-системы используют заголовок X-FORWARDED-FOR для определения адреса клиента и не смотрят на адрес отправителя в IP-пакете.

Теперь с помощью директивы proxy_bind укажем, где и как использовать указанное распределение клиентов. Директиву можно задать в контекстах `http`, `server` или `location`.

Пример:

```
location / {
proxy_pass http://backend ;
proxy_bind ${variant};
}
```

8.5.3 Заключение

Предложенные выше способы ручного создания SNAT-пула позволят уверенно перешагнуть ограничение в 65000 одновременных сессий как для L4-, так и для L7-балансировки. Однако следует все же понимать, что ограничение количества одновременно поддерживаемых сессий зависит не только от количества сокетов, но и от объемов использования целого ряда других системных ресурсов.

Чтобы адаптировать используемый подход к ситуации, когда один из интерфейсов Angie ADC расположен в одной подсети с апстрим-серверами, потребуется следующее:

- не использовать динамическую маршрутизацию;
- настроить адреса из пула на физическом интерфейсе, а не на loopback (адреса должны быть из той же IP-подсети, в которой расположены серверы);

- для обеспечения отказоустойчивости можно использовать несколько Angie ADC, а IP-адреса из пула указать как виртуальные secondary-адреса для VRRP, вместо того чтобы назначать их непосредственно на физический интерфейс.

ГЛАВА 9

Права на интеллектуальную собственность

Документация на программный продукт Angie ADC является интеллектуальной собственностью ООО «Веб-Сервер».

Алфавитный указатель