Руководство по установке

Версия 1.0

Аннотация

Настоящий документ содержит пошаговую инструкцию по развертыванию Angie PRO.

Angie PRO — эффективный, мощный и масштабируемый веб-сервер.

Оглавление

1 Общие сведения

Angie PRO – единственный коммерческий веб-сервер разработка которого локализована в России.

Веб-сервер — это класс программного обеспечения, предоставляющего доступ к сетевым ресурсам по протоколу HTTP конечным пользователям.

Angie PRO, например, может использоваться для работы интернет-сайтов, мобильных приложений, киосков самообслуживания в метрополитене, мультимедиа систем в поездах дальнего следования.

Для начала работы с Angie PRO установите его в своем окружении руководствуясь данным документом.

2 Поддерживаемые дистрибутивы

Alpine

Версия	Платформа
3.17	x86_64, aarch64
3.16	x86_64, aarch64
3.15	x86_64, aarch64
3.14	x86_64, aarch64

Debian

Версия	Платформа
11 «Bullseye»	x86_64, aarch64
10 «Buster»	x86_64, aarch64

Oracle

Версия	Платформа
9	x86_64, aarch64
8	x86_64, aarch64

RED OS

Версия	Платформа
7.3 «MUROM»	x86_64

Rocky

Версия	Платформа
9	x86_64, aarch64
8	x86_64, aarch64

Ubuntu

Версия	Платформа
22.04 «Jammy Jellyfish»	x86_64, aarch64
20.04 «Focal Fossa»	x86_64, aarch64

3 Установка

Чтобы поставить Angie PRO на новой машине, необходимо подключить и настроить репозиторий пакетов Angie PRO. После этого можно будет установить и обновлять Angie PRO из этого репозитория средствами пакетного менеджера.

Примечание: Если у вас есть лицензия Angie PRO, но нет сертификата/ключа, обратитесь по адресу support@angie.software

3.1 Установка на сервер

Alpine

1. Переименуйте полученные с лицензией файлы:
   - angie-repo.crt в /etc/apk/cert.pem — сертификат
   - angie-repo.key в /etc/apk/cert.key — приватный ключ
2. Установите пакеты, необходимые для подключения репозитория Angie PRO:

```
$ sudo apk add curl ca-certificates
```

3. Скачайте публичный ключ репозитория Angie PRO, используемый apk для проверки подлинности пакетов:

```
$ sudo curl -o /etc/apk/keys/angie-signing.rsa \
       https://angie.software/keys/angie-signing.rsa
```

4. Добавьте репозиторий Angie PRO в файл /etc/apk/repositories:

```
$ echo "https://download.angie.software/angie-pro/alpine/v$(egrep -o \
     '[0-9]+\.[0-9]+' /etc/alpine-release)/main" \
     | sudo tee -a /etc/apk/repositories >/dev/null
```

5. Установите пакет Angie PRO:

```
$ sudo apk add angie-pro
```

6. Если необходимо, установите дополнительно модули:

```
$ sudo apk add <package-name>
```

7. Запустите сервис:

```
$ sudo service angie start
```

8. Чтобы сервис стартовал после перезагрузки сервера, выполните команду:

```
$ sudo rc-update add angie
```

Debian / Ubuntu

1. Создайте директорию /etc/ssl/angie:

```
$ sudo mkdir -p /etc/ssl/angie
```

2. Поместите полученные с лицензией файлы:

   - angie-repo.crt в /etc/ssl/angie/angie-repo.crt — сертификат
   - angie-repo.key в /etc/ssl/angie/angie-repo.key — приватный ключ

   Выполните команду:

```
$ sudo chown -R _apt:nogroup /etc/ssl/angie
```

3. Установите пакеты, необходимые для подключения репозитория Angie PRO:

```
$ sudo apt-get install -y apt-transport-https lsb-release \
      ca-certificates curl gnupg2
```

4. Скачайте публичный ключ репозитория Angie PRO, используемый apt для проверки подлинности пакетов:

```
$ sudo curl -o /etc/apt/trusted.gpg.d/angie-signing.gpg \
      https://angie.software/keys/angie-signing.gpg
```

5. Для подключения apt-репозитория Angie PRO, выполните следующую команду:

   Debian

```
$ echo "deb https://download.angie.software/angie-pro/debian/ `lsb_release -cs` main" \
      | sudo tee /etc/apt/sources.list.d/angie.list >/dev/null
```

   Ubuntu

```
$ echo "deb https://download.angie.software/angie-pro/ubuntu/ `lsb_release -cs` main" \
      | sudo tee /etc/apt/sources.list.d/angie.list >/dev/null
```

6. Создайте файл конфигурации apt репозитория Angie PRO в /etc/apt/apt.conf.d:

```
$ sudo vi /etc/apt/apt.conf.d/90download-angie
```

со следующим содержимым:

```
Acquire::https::download.angie.software::Verify-Peer "true";
Acquire::https::download.angie.software::Verify-Host "true";
Acquire::https::download.angie.software::SslCert     "/etc/ssl/angie/angie-repo.crt";
Acquire::https::download.angie.software::SslKey      "/etc/ssl/angie/angie-repo.key";
```

7. Установите пакет Angie PRO:

```
$ sudo apt-get update
$ sudo apt-get install -y angie-pro
```

8. Если необходимо, установите дополнительно пакеты модулей:

```
$ sudo apt-get install -y <package-name>
```

Oracle / RED OS / Rocky

1. Создайте директорию /etc/ssl/angie:

```
$ sudo mkdir -p /etc/ssl/angie
```

2. Поместите полученные с лицензией файлы:

   - angie-repo.crt в /etc/ssl/angie/angie-repo.crt — сертификат
   - angie-repo.key в /etc/ssl/angie/angie-repo.key — приватный ключ

3. Для подключения yum-репозитория создайте файл с именем /etc/yum.repos.d/angie.repo со следующим содержимым:

   Oracle

```
[angie-pro]
name=Angie PRO repo
baseurl=https://download.angie.software/angie-pro/oracle/$releasever/
sslclientcert=/etc/ssl/angie/angie-repo.crt
sslclientkey=/etc/ssl/angie/angie-repo.key
gpgcheck=1
enabled=1
gpgkey=https://angie.software/keys/angie-signing.gpg
```

   RED OS 7.3

```
[angie-pro]
name=Angie PRO repo
baseurl=https://download.angie.software/angie-pro/redos/73/
sslclientcert=/etc/ssl/angie/angie-repo.crt
sslclientkey=/etc/ssl/angie/angie-repo.key
gpgcheck=1
enabled=1
gpgkey=https://angie.software/keys/angie-signing.gpg
```

   Rocky

```
[angie-pro]
name=Angie PRO repo
baseurl=https://download.angie.software/angie-pro/rocky/$releasever/
sslclientcert=/etc/ssl/angie/angie-repo.crt
sslclientkey=/etc/ssl/angie/angie-repo.key
gpgcheck=1
enabled=1
gpgkey=https://angie.software/keys/angie-signing.gpg
```

4. Чтобы установить Angie PRO, выполните следующую команду:

```
$ sudo yum install angie-pro
```

5. Если необходимо, установите дополнительно модули:

```
$ sudo yum install <package-name>
```

6. Запустите сервис:

```
$ sudo systemctl start angie
```

7. Чтобы сервис стартовал после перезагрузки сервера, выполните команду:

```
$ sudo systemctl enable angie
```

3.2 Динамические модули

В репозитории Angie PRO представлены следующие пакеты динамических модулей:

```
angie-pro-module-auth-spnego
angie-pro-module-brotli
angie-pro-module-dav-ext
angie-pro-module-geoip2
angie-pro-module-headers-more
angie-pro-module-image-filter
angie-pro-module-ndk
angie-pro-module-njs
angie-pro-module-perl
angie-pro-module-rtmp
angie-pro-module-set-misc
angie-pro-module-xslt
```

4 Управление Angie PRO

Процесс Angie PRO запускается как системный сервис следующей командой:

```
sudo service angie start
```

Рекомендуется перед стартом выполнить проверку синтаксиса конфигурационного файла, команда:

```
sudo angie -t && sudo service angie start
```

Команда для перезагрузки:

```
sudo angie -t && sudo service angie reload
```

Команда для остановки процесса Angie:

```
sudo service angie stop
```

После первичной установки можно проверить, что Angie успешно запустился:

```
curl localhost:80
```

У Angie PRO есть один главный и несколько рабочих процессов. Основная задача главного процесса — чтение и проверка конфигурации и управление рабочими процессами. Рабочие процессы выполняют фактическую обработку запросов. Angie PRO использует модель, основанную на событиях, и зависящие от операционной системы механизмы для эффективного распределения запросов между рабочими процессами. Количество рабочих процессов задаётся в конфигурационном файле и может быть фиксированным для данной конфигурации или автоматически устанавливаться равным числу доступных процессорных ядер.

Управлять Angie PRO можно также с помощью сигналов. Номер главного процесса по умолчанию записывается в файл /var/run/angie.pid. Изменить имя этого файла можно при конфигурации сборки или же в angie.conf директивой pid. Главный процесс поддерживает следующие сигналы:

TERM, INT	быстрое завершение
QUIT	плавное завершение
HUP	изменение конфигурации, обновление изменившейся временной зоны (только для FreeBSD и Linux), запуск новых рабочих процессов с новой конфигурацией, плавное завершение старых рабочих процессов
USR1	переоткрытие лог-файлов
USR2	обновление исполняемого файла
WINCH	плавное завершение рабочих процессов

Управлять рабочими процессами по отдельности не нужно. Тем не менее, они тоже поддерживают некоторые сигналы:

TERM, INT	быстрое завершение
QUIT	плавное завершение
USR1	переоткрытие лог-файлов
WINCH	аварийное завершение для отладки (требует включения debug_points)

4.1 Изменение конфигурации

Для того чтобы Angie PRO перечитал файл конфигурации, нужно послать главному процессу сигнал HUP. Главный процесс сначала проверяет синтаксическую правильность конфигурации, а затем пытается применить новую конфигурацию, то есть, открыть лог-файлы и новые слушающие сокеты. Если ему это не удаётся, то он откатывает изменения и продолжает работать со старой конфигурацией. Если же удаётся, то он запускает новые рабочие процессы, а старым шлёт сообщение о плавном выходе. Старые рабочие процессы закрывают слушающие сокеты и продолжают обслуживать старых клиентов. После обслуживания всех клиентов старые рабочие процессы завершаются.

4.2 Ротация лог-файлов

Лог-файлы нужно переименовать, а затем послать сигнал USR1 главному процессу. Он откроет заново все текущие открытые файлы и назначит им в качестве владельца непривилегированного пользователя, под которым работают рабочие процессы. После успешного открытия главный процесс закрывает все открытые файлы и посылает сообщение о переоткрытии файлов рабочим процессам. Они также открывают новые файлы и сразу же закрывают старые. В результате старые файлы практически сразу же готовы для дальнейшей обработки, например, их можно сжимать.

4.3 Обновление исполняемого файла на лету

Для обновления исполняемого файла сервера вначале нужно записать на место старого файла новый. Затем нужно послать сигнал USR2 главному процессу — он переименует свой файл с номером процесса в файл с суффиксом .oldbin, например, /usr/local/angie/logs/angie.pid.oldbin, после чего запустит новый исполняемый файл, а тот в свою очередь — свои рабочие процессы.

Старый процесс не закрывает свои слушающие сокеты и при необходимости ему можно сказать, чтобы он снова запустил свои рабочие процессы. Если работа нового исполняемого файла по каким-то причинам не устраивает, можно проделать одно из следующих действий:

- Послать старому главному процессу сигнал HUP. Старый главный процесс, не перечитывая конфигурации, запустит новые рабочие процессы. После этого можно плавно завершить все новые процессы, послав новому главному процессу сигнал QUIT.
- Послать новому главному процессу сигнал TERM. В ответ на это он пошлёт сообщение о немедленном выходе своим рабочим процессам, и все они практически сразу же завершатся. (Если новые процессы по каким-то причинам не завершаются, нужно послать им сигнал KILL, который заставит их завершиться.) По завершению нового главного процесса старый главный процесс автоматически запустит новые рабочие процессы.

Если новый главный процесс выходит, то старый главный процесс убирает суффикс .oldbin из имени файла с номером процесса.

Если же обновление прошло удачно, то старому процессу нужно послать сигнал QUIT, и останутся только новые процессы.

4.4 Управление из командной строки

-? | -h	вывод справки по параметрам командной строки
-c файл	использование альтернативного конфигурационного файла файл вместо файла по умолчанию.
-e файл	использование альтернативного лог-файла ошибок файл вместо файла по умолчанию. Специальное значение stderr выбирает стандартный файл ошибок
-g директивы	задание глобальных директив конфигурации, например: angie -g "pid /var/run/angie.pid; worker_processes `sysctl -n hw.ncpu`;"
-p префикс	задание префикса пути angie, т.е. каталога, в котором будут находиться файлы сервера (по умолчанию — каталог /usr/local/angie)
-q	вывод только сообщений об ошибках при тестировании конфигурации
-s сигнал	отправка сигнала главному процессу: stop, quit, reopen, reload
-t	тестирование конфигурационного файла: Angie проверяет синтаксическую правильность конфигурации, а затем пытается открыть файлы, описанные в конфигурации
-T	то же, что и -t, а также вывод конфигурационных файлов в стандартный поток вывода
-v	вывод версии Angie Pro
-V	вывод версии Angie Pro, версии компилятора и параметров конфигурации сборки

5 Как работает Angie PRO

5.1 Методы обработки соединений

Angie PRO поддерживает различные методы обработки соединений. Наличие того или иного метода зависит от используемой платформы. Если на платформе доступно сразу несколько методов, Angie PRO обычно сам выбирает наиболее эффективный метод. Однако при необходимости можно явно выбрать метод обработки соединений с помощью директивы[1] use.

[1] Подробное описание директив и переменных приведены в руководстве по эксплуатации Angie Pro

Поддерживаются следующие методы обработки соединений:

select

Стандартный метод. Модуль для поддержки этого метода собирается автоматически, если на платформе не обнаружено более эффективного метода. Можно принудительно разрешить или запретить сборку этого модуля с помощью параметров --with-select_module и --without-select_module.

poll

Стандартный метод. Модуль для поддержки этого метода собирается автоматически, если на платформе не обнаружено более эффективного метода. Можно принудительно разрешить или запретить сборку этого модуля с помощью параметров --with-poll_module и --without-poll_module.

kqueue

Эффективный метод, используемый во FreeBSD 4.1+, OpenBSD 2.9+, NetBSD 2.0 и macOS.

epoll

Эффективный метод, используемый в Linux 2.6+.

/dev/poll

Эффективный метод, используемый в Solaris 7 11/99+, HP/UX 11.22+ (eventport), IRIX 6.5.15+ и Tru64 UNIX 5.1A+.

eventport

event ports, метод, используемый в Solaris 10+ (из-за имеющихся проблем вместо него рекомендуется использовать метод /dev/poll).

5.2 Обработка TCP/UDP-сессий

Обработка клиентской TCP/UDP-сессии происходит последовательными фазами:

Post-accept	Первая фаза после принятия клиентского соединения. В этой фазе выполняется модуль stream_realip.
Pre-access	Предварительная проверка доступа. В этой фазе выполняются модули stream_limit_conn и stream_set.
Access	Ограничение доступа для клиента перед обработкой данных. В этой фазе выполняется модуль stream_access.
SSL	Терминирование TLS/SSL. В этой фазе выполняется модуль stream_ssl.
Preread	Чтение первых байт данных в буфер предварительного чтения для анализа,

	например модулем stream_ssl_preread, перед их обработкой.
Content	Обязательная фаза, в которой происходит обработка данных, как правило проксирование на группу серверов или отправка клиенту заданного значения.
Log	Заключительная фаза, в которой записывается результат обработки клиентской сессии. В этой фазе выполняется модуль stream_log.

5.3 Обработка запросов

Выбор виртуального сервера

Сначала соединение создаётся в контесте сервера по умолчанию. Затем имя сервера может быть определено на следующих стадиях обработки запроса, каждая из которых участвует в выборе конфигурации:

- предварительно во время операции SSL handshake согласно SNI
- после обработки строки запроса
- после обработки поля “Host“ заголовка запроса

Если после обработки строки запроса или поля “Host” заголовка запроса имя сервера не было выбрано, то Angie будет использовать пустое имя в качестве имени сервера.

На каждой из этих стадий могут применяться различные конфигурации сервера. То есть, некоторые директивы[2] следует указывать с осторожностью:

- в случае использования директивы ssl_protocols список протоколов задаётся библиотекой OpenSSL перед применением конфигурации сервера согласно имени, запрашиваемого через SNI. Таким образом, протоколы должны быть заданы только для сервера по умолчанию;
- директивы client_header_buffer_size и merge_slashes задействуются перед чтением строки запроса, они используют конфигурацию сервера по умолчанию или конфигурацию сервера, выбранного через SNI;
- в случае использования директив ignore_invalid_headers, large_client_header_buffers и underscores_in_headers, которые участвуют в обработке полей заголовка запроса, выбор сервера дополнительно зависит от того, была ли обновлена конфигурация сервера согласно строке запроса или полю заголовка “Host”;
- ошибочный ответ будет обработан с помощью директивы error_page в том сервере, который в настоящий момент выполняет запрос.

Определение виртуального сервера по имени

Angie PRO вначале решает, какой из серверов должен обработать запрос. Рассмотрим простую конфигурацию, где все три виртуальных сервера слушают на порту **:80*:

---

[2] Подробное описание директив и переменных приведены в руководстве по эксплуатации Angie Pro

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

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

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

В этой конфигурации, чтобы определить, какому серверу следует направить запрос, Angie PRO проверяет только поле “Host” заголовка запроса. Если его значение не соответствует ни одному из имён серверов или в заголовке запроса нет этого поля вовсе, Angie направит запрос в сервер по умолчанию для этого порта. В вышеприведённой конфигурации сервером по умолчанию будет первый сервер, что соответствует стандартному поведению Angie по умолчанию. Сервер по умолчанию можно задать явно с помощью параметра *default_server* в директиве[3] listen:

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

Примечание

Следует иметь в виду, что сервер по умолчанию является свойством слушающего сокета, а не имени сервера.

Как предотвратить обработку запросов без имени сервера

Если запросы без поля “Host” в заголовке не должны обрабатываться, можно определить сервер, который будет их отклонять:

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

[3] Подробное описание директив и переменных приведены в руководстве по эксплуатации Angie Pro

Здесь в качестве имени сервера указана пустая строка, которая соответствует запросам без поля “Host” в заголовке, и возвращается специальный код 444, который закрывает соединение.

Определение виртуального сервера по имени и IP-адресу

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

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

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

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

В этой конфигурации Angie вначале сопоставляет IP-адрес и порт запроса с директивами listen в блоках server. Затем он сопоставляет значение поля “Host” заголовка запроса с директивами[4] server_name в блоках server, которые соответствуют IP-адресу и порту. Если имя сервера не найдено, запрос будет обработан в сервере по умолчанию. Например, запрос www.example.com, пришедший на порт 192.168.1.1:80, будет обработан сервером по умолчанию для порта 192.168.1.1:80, т.е. первым сервером, т.к. для этого порта *www.example.com* не указан в списке имён серверов.

Сервер по умолчанию является свойством слушающего сокета, поэтому у разных сокетов могут быть определены свои серверы по умолчанию:

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

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

[4] Подробное описание директив и переменных приведены в руководстве по эксплуатации Angie Pro

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

Определение location

На примере простого PHP-сайта:

```
server {
  listen      80;
  server_name example.org www.example.org;
  root        /data/www;

  location / {
     index   index.html index.php;
  }

  location ~* \.(gif|jpg|png)$ {
     expires 30d;
  }

  location ~ \.php$ {
     fastcgi_pass  localhost:9000;
     fastcgi_param SCRIPT_FILENAME
                   $document_root$fastcgi_script_name;
     include       fastcgi_params;
  }
}
```

Angie PRO вначале ищет среди всех префиксных *location*, заданных строками, максимально совпадающий. В вышеприведённой конфигурации указан только один префиксный *location /*, и, поскольку он подходит под любой запрос, он и будет использован, если других совпадений не будет найдено. Затем Angie проверяет *location*, заданные регулярными выражениями, в порядке их следования в конфигурационном файле. При первом же совпадении поиск прекращается и Angie использует совпавший *location*. Если запросу не соответствует ни одно из регулярных выражений, Angie использует максимально совпавший префиксный *location*, найденный ранее.

Примечание

Следует иметь в виду, что *location* всех типов сопоставляются только с URI-частью строки запроса без аргументов. Так делается потому, что аргументы в строке запроса могут быть заданы различными способами, например:

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

Кроме того, в строке запроса можно запросить что угодно:

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

Теперь посмотрим, как бы обрабатывались запросы в вышеприведённой конфигурации:

Запросу /logo.gif во-первых соответствует префиксный *location /*, а во-вторых — регулярное выражение *.(gif|jpg|png)$*, поэтому он обрабатывается location'ом регулярного выражения. Согласно директиве[5] *root /data/www* запрос отображается в файл /data/www/logo.gif, который и посылается клиенту.

Запросу /index.php также во-первых соответствует префиксный *location /*, а во-вторых — регулярное выражение *.(php)$*. Следовательно, он обрабатывается *location*'ом регулярного выражения и запрос передаётся FastCGI-серверу, слушающему на *localhost:9000*. Директива[6] fastcgi_param устанавливает FastCGI-параметр *SCRIPT_FILENAME* в */data/www/index.php*, и сервер FastCGI выполняет указанный файл. Переменная $document_root равна значению директивы *root*, а переменная $fastcgi_script_name равна URI запроса, т.е. /index.php.

Запросу /about.html соответствует только префиксный *location /*, поэтому запрос обрабатывается в нём. Согласно директиве *root /data/www* запрос отображается в файл /data/www/about.html, который и посылается клиенту.

Обработка запроса / более сложная. Ему соответствует только префиксный *location /*, поэтому запрос обрабатывается в нём. Затем директива index проверяет существование индексных файлов согласно своих параметров и директиве *root /data/www*. Если файл /data/www/index.html не существует, а файл /data/www/index.php существует, то директива делает внутреннее перенаправление на /index.php и Angie снова сопоставляет его с *location*'ами, как если бы такой запрос был послан клиентом. Как мы видели ранее, перенаправленный запрос будет в конечном итоге обработан сервером FastCGI.

5.4 Проксирование и балансировка нагрузки

Одним из частых применений Angie является использование его в качестве прокси-сервера, то есть сервера, который принимает запросы, перенаправляет их на проксируемые сервера, получает ответы от них и отправляет их клиенту.

Простейшая конфигурация прокси-сервера:

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

Директива proxy_pass инструктирует Angie передавать клентские запросы на сервер бэкенда *backend:8080* (проксируемый сервер). Проксирование посредством Angie гибко конфигурируется множеством других директив.

---

[5] Подробное описание директив и переменных приведены в руководстве по эксплуатации Angie Pro
[6] Подробное описание директив и переменных приведены в руководстве по эксплуатации Angie Pro

Проксирование FastCGI

Angie можно использовать для перенаправления запросов на FastCGI-серверы. На них могут исполняться приложения, созданные с использованием разнообразных фреймворков и языков программирования, например, PHP.

Базовая конфигурация Angie для работы с проксируемым FastCGI-сервером включает в себя использование директивы fastcgi_pass вместо директивы proxy_pass, и директив fastcgi_param для настройки параметров, передаваемых FastCGI-серверу. Представьте, что FastCGI-сервер доступен по адресу *localhost:9000*. В PHP параметр SCRIPT_FILENAME используется для определения имени скрипта, а в параметре QUERY_STRING передаются параметры запроса. Получится следующая конфигурация:

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

    location ~ \.(gif|jpg|png)$ {
        root /data/images;
    }
}
```

Таким образом будет настроен сервер, который будет перенаправлять все запросы, кроме запросов статических изображений, на проксируемый сервер, работающий по адресу localhost:9000, по протоколу FastCGI.

Проксирование WebSocket

Для превращения соединения между клиентом и сервером из HTTP/1.1 в WebSocket используется доступный в HTTP/1.1 механизм смены протокола.

Но есть сложность: поскольку “Upgrade” является hop-by-hop заголовком, то он не передаётся от клиента к проксируемому серверу. При прямом проксировании клиенты могут использовать метод CONNECT, чтобы обойти эту проблему. Однако при обратном проксировании такой подход не работает, так как клиент ничего о проксирующем сервере не знает, и требуется специальная обработка на проксирующем сервере.

В Angie предусмотрен особый режим работы, который позволяет установить туннель между клиентом и проксируемым сервером, если проксируемый сервер вернул ответ с кодом 101 (Switching Protocols), и клиент попросил сменить протокол с помощью заголовка “Upgrade” в запросе.

Как уже отмечалось выше, hop-by-hop заголовки, включая “Upgrade” и “Connection”, не передаются от клиента к проксируемому серверу, поэтому, для того чтобы проксируемый сервер узнал о намерении клиента сменить протокол на WebSocket, эти заголовки следует передать явно:

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

Более сложный пример, в котором значение поля “Connection” в заголовке запроса к проксируемому серверу зависит от наличия поля “Upgrade” в заголовке запроса клиента:

```
http {
    map $http_upgrade $connection_upgrade {
        default upgrade;
        ''      close;
    }

    server {
        ...

        location /chat/ {
            proxy_pass http://backend;
            proxy_http_version 1.1;
            proxy_set_header Upgrade $http_upgrade;
            proxy_set_header Connection $connection_upgrade;
        }
    }
```

По умолчанию соединение будет закрыто, если с проксируемого сервера данные не передавались в течение 60 секунд. Этот таймаут можно увеличить при помощи директивы[7] proxy_read_timeout. Кроме того, на проксируемом сервере можно настроить периодическую отправку WebSocket ping-фреймов для сброса таймаута и проверки работоспособности соединения.

Балансировка нагрузки

Для оптимизации использования ресурсов, повышения устойчивости и скорости обработки запросов широко используется распределение нагрузки клиентского трафика между несколькими серверами приложений. Angie эффективно решает эту задачу.

Простейшая конфигурация для распределения клиентского трафика по трём бэкендам выглядит так:

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

    server {
```

[7] Подробное описание директив и переменных приведены в руководстве по эксплуатации Angie Pro

```
        listen 80;

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

В примере три сервера srv1-srv3 одного приложения объединены в группу *upstream* и при проксировании Angie PRO распределит между ними клиентские запросы. По умолчанию используется алгоритм распределения round-robin. Возможно использование других алгоритмов: weight, least_conn, ip_hash. Для группы серверов upstream по умолчанию работает механизм проверки работоспособности каждого сервера, настраиваемый в контексте upstream параметрами max_fails и fail_timeout директивы[8] server.

5.5 Логирование

Запись в syslog

Директивы error_log и access_log поддерживают запись в *syslog*. Запись в *syslog* настраивается при помощи следующих параметров:

server= *адрес*	Задаёт адрес сервера *syslog*. 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 UDP-порт 514. Если доменному имени соответствует несколько IP-адресов, используется только первый адрес.
facility= *строка*	Задаёт категорию сообщений syslog в соответствии с RFC 3164. В качестве категории может быть указано одно из следующих значений: *“kern”*, *“user”*, *“mail”*, *“daemon”*, *“auth”*, *“intern”*, *“lpr”*, *“news”*, *“uucp”*, *“clock”*, *“authpriv”*, *“ftp”*, *“ntp”*, *“audit”*, *“alert”*, *“cron”*, *“local0”..“local7”*. По умолчанию используется *“local7”*.
severity= *строка*	Задаёт важность сообщений *syslog* для access_log в соответствии с RFC 3164. Возможны те же самые значения, что и у второго параметра (уровень) директивы error_log. По умолчанию используется *“info”*. Важность сообщений об ошибках определяется самим Angie Pro, поэтому в директиве error_log параметр игнорируется.
tag= *строка*	Задаёт метку сообщений syslog. По умолчанию используется *“angie”*.
nohostname	Запрещает добавление поля “hostname” в заголовок сообщения syslog

Пример конфигурации syslog:

```
error_log syslog:server=192.168.1.1 debug;
```

[8] Подробное описание директив и переменных приведены в руководстве по эксплуатации Angie Pro

```
access_log syslog:server=unix:/var/log/angie.sock,nohostname;
access_log syslog:server=[2001:db8::1]:12345,facility=local7,tag=angie,severity=info
combined;
```

Отладочный лог

Для работы отладочного лога Angie PRO должен быть сконфигурирован с поддержкой отладки на этапе сборки:

./configure –with-debug …

Затем нужно задать уровень debug с помощью директивы error_log:

```
error_log /path/to/log debug;
```

Чтобы убедиться, что поддержка отладки сконфигурирована, необходимо выполнить команду *angie -V*:

configure arguments: –with-debug …

Готовые пакеты для Linux по умолчанию предоставляют поддержку отладочного лога при помощи бинарного файла angie-debug. Из пакета бинарные файлы Angie располагаются:

```
$ ls -l /usr/sbin/ | grep angie
   lrwxrwxrwx 1 root root      13 Sep 21 18:58 angie -> angie-nodebug
   -rwxr-xr-x 1 root root 1561224 Sep 21 18:58 angie-debug
   -rwxr-xr-x 1 root root 1426056 Sep 21 18:58 angie-nodebug
```

systemd service Angie сконфигурирован:

```
DAEMON=${DAEMON:-/usr/sbin/angie}
```

Поэтому для того, чтобы запустить бинарный файл Angie с поддержкой отладочного лога, нужно переназначить symlink:

```
$ ln -fs /usr/sbin/angie-debug /usr/sbin/angie
```

и выполнить команду

```
$ sudo service angie upgrade
```

которая запустит процедуру обновления исполняемого файла на лету, и затем задать уровень *debug*.

Обратите внимание, что переопределение лога без одновременного указания уровня *debug* отключит отладочный лог. В примере ниже, переопределение лога на уровне server отключает отладочный лог для этого сервера:

```
error_log /path/to/log debug;

http {
   server {
```

```
    error_log /path/to/log;
   # ...
```

Чтобы избежать этого, следует либо закомментировать строку, переопределяющую лог, либо добавить определение уровня *debug*:

```
error_log /path/to/log debug;

http {
   server {
     error_log /path/to/log debug;
   #  ...
```

Отладочный лог для определённых клиентов

Можно включить отладочный лог только для определённых клиентских адресов:

```
error_log /path/to/log;

events {
  debug_connection 192.168.1.1;
  debug_connection 192.168.10.0/24;
}
```

Запись в кольцевой буфер в памяти

Отладочный лог можно записывать в кольцевой буфер в памяти:

```
error_log memory:32m debug;
```

Запись в буфер в памяти на уровне *debug* не оказывает существенного влияния на производительность даже при высоких нагрузках. В этом случае лог может быть извлечён при помощи gdb-скрипта, подобного следующему:

```
set $log = ngx_cycle->log

while $log->writer != ngx_log_memory_writer
  set $log = $log->next
end

set $buf = (ngx_log_memory_buf_t *) $log->wdata
dump binary memory debug_log.txt $buf->start $buf->end
```

6 Конфигурация

Angie работает с текстовым конфигурационным файлом. Файл конфигурации *angie.conf* находится по пути conf-path, указанном при компиляции, по умолчанию в */etc/angie*.

Файл конфигурации состоит из контекстов:

- *events* – обработка соединений
- *http* – трафик HTTP

- *mail* – Mail трафик
- *stream* – TCP и UDP трафик

Директивы, расположенные вне этих контекстов, считаются директивами контекста *main*.

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

events {
    # конфигурация обработки соединений
}

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

    server {
        # конфигурация виртуального HTTP сервера 1
        location /one {
            # конфигурация обработки HTTP запросов с URI, начинающимися с '/one'
        }
        location /two {
            # конфигурация обработки HTTP запросов с URI, начинающимися с '/two'
        }
    }

    server {
        # конфигурация виртуального HTTP сервера 2
    }
}

stream {
    # Конфигурация трафика TCP/UDP, для всех вложенных виртуальных серверов
    server {
        # конфигурация виртуального TCP сервера 1
    }
}
```

Чтобы облегчить управление конфигурацией, рекомендуется использование директивы include в основном файле конфигурации angie.conf для включения в него специализированных файлов.

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

6.1 Наследование

Вложенный контекст — включённый внутри родительского, — наследует директивы из родительского контекста тогда и только тогда, когда такие директивы не описаны в нём самом. При наличии, директива вложенного контекста переопределяет директиву родительского.

6.2 Перезагрузка конфигурации

При любом изменении конфигурации, для применения изменений процесс Angie необходимо:

либо перезапустить полностью, предварительно проверив конфигурацию синтаксически:

```
$ sudo angie -t && sudo service angie restart
```

либо перезагрузить, что позволит не прерывать обработку текущих соединений.

```
$ sudo angie -t && sudo service angie reload
```

6.3 Единицы измерения

Размеры в конфигурационном файле можно указывать в байтах, килобайтах (суффиксы *k* и *K*) или мегабайтах (суффиксы *m* и *M*), например, *“1024”*, *“8k”*, *“1m”*.

Интервалы времени можно задавать в миллисекундах, секундах, минутах, часах, днях и т.д., используя следующие суффиксы:

ms	миллисекунды
s	секунды
m	минуты
h	часы
d	дни
w	недели
M	месяцы, 30 дней
y	годы, 365 дней

В одном значении можно комбинировать различные единицы, указывая их в порядке от более к менее значащим, и по желанию отделяя их пробелами. Например, *“1h 30m”* задаёт то же время, что и *“90m”* или *“5400s”*. Значение без суффикса задаёт секунды. Рекомендуется всегда указывать суффикс.

Некоторые интервалы времени можно задать лишь с точностью до секунд.

6.4 Настройка хэшей

Для быстрой обработки статических наборов данных, таких как имена серверов, значения директивы map, MIME-типы, имена полей заголовков запросов, Angie использует хэш-таблицы. Во время старта и при каждой переконфигурации Angie подбирает минимально возможный размер хэш-таблиц с учётом того, чтобы размер корзины, куда попадают ключи с совпадающими хэш-значениями, не превышал заданного параметра (*hash bucket size*).

Размер таблицы считается в корзинах. Подбор ведётся до тех пор, пока размер таблицы не превысит параметр *hash max size*. Для большинства хэшей есть директивы, которые позволяют менять эти параметры, например, для хэшей имён серверов директивы называются server_names_hash_max_size и server_names_hash_bucket_size.

Параметр *hash bucket size* всегда выравнивается до размера, кратного размеру строки кэша процессора. Это позволяет ускорить поиск ключа в хэше на современных процессорах, уменьшив число обращений к памяти. Если *hash bucket size* равен размеру одной строки кэша процессора, то во время поиска ключа число обращений к памяти в худшем случае будет равно двум — первый раз для определения адреса корзины, а второй — при поиске ключа внутри корзины. Соответственно, если Angie выдал сообщение о необходимости увеличить *hash max size* или *hash bucket size*, то сначала нужно увеличивать первый параметр.

6.5 Настройка HTTPS-серверов

Чтобы настроить HTTPS-сервер, необходимо включить параметр ssl на слушающих сокетах в блоке server, а также указать местоположение файлов с сертификатом сервера и секретным ключом:

```
server {
    listen              443 ssl;
    server_name         www.example.com;
    ssl_certificate     www.example.com.crt;
    ssl_certificate_key www.example.com.key;
    ssl_protocols       TLSv1 TLSv1.1 TLSv1.2;
    ssl_ciphers         HIGH:!aNULL:!MD5;
#...
}
```

Сертификат сервера является публичным. Он посылается каждому клиенту, соединяющемуся с сервером. Секретный ключ следует хранить в файле с ограниченным доступом (права доступа должны позволять главному процессу Angie PRO читать этот файл). Секретный ключ можно также хранить в одном файле с сертификатом:

```
ssl_certificate     www.example.com.cert;
ssl_certificate_key www.example.com.cert;
```

при этом права доступа к файлу следует также ограничить. Несмотря на то, что и сертификат, и ключ хранятся в одном файле, клиенту посылается только сертификат.

С помощью директив ssl_protocols и ssl_ciphers можно ограничить соединения использованием только “сильных” версий и шифров SSL/TLS. По умолчанию Angie PRO использует:

```
ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
ssl_ciphers HIGH:!aNULL:!MD5;
```

поэтому их явная настройка в общем случае не требуется.

Оптимизация HTTPS-сервера

SSL-операции потребляют дополнительные ресурсы процессора. На мультипроцессорных системах следует запускать несколько рабочих процессов, не меньше числа доступных процессорных ядер. Наиболее ресурсоёмкой для процессора является операция SSL handshake, в рамках которой формируются криптографические параметры сессии. Существует два способа уменьшения числа этих операций, производимых для каждого клиента: использование постоянных (keepalive) соединений, позволяющих в рамках одного соединения обрабатывать сразу несколько запросов, и повторное использование параметров SSL-сессии для предотвращения необходимости выполнения SSL handshake для параллельных и последующих соединений. Сессии хранятся в кэше SSL-сессий, разделяемом между рабочими процессами и настраиваемом директивой ssl_session_cache. В 1 мегабайт кэша помещается около 4000 сессий. Таймаут кэша по умолчанию равен 5 минутам. Он может быть увеличен с помощью директивы ssl_session_timeout. Вот пример конфигурации, оптимизированной под многоядерную систему с 10-мегабайтным разделяемым кэшем сессий:

```
worker_processes auto;

http {
    ssl_session_cache   shared:SSL:10m;
    ssl_session_timeout 10m;

    server {
        listen              443 ssl;
        server_name         www.example.com;
        keepalive_timeout   70;

        ssl_certificate     www.example.com.crt;
        ssl_certificate_key www.example.com.key;
        ssl_protocols       TLSv1 TLSv1.1 TLSv1.2;
        ssl_ciphers         HIGH:!aNULL:!MD5;
    #...
```

Цепочки SSL-сертификатов

Некоторые браузеры могут выдавать предупреждение о сертификате, подписанном общеизвестным центром сертификации, в то время как другие браузеры без проблем принимают этот же сертификат. Так происходит потому, что центр, выдавший сертификат, подписал его промежуточным сертификатом, которого нет в базе данных сертификатов общеизвестных доверенных центров сертификации, распространяемой вместе с браузером. В подобном случае центр сертификации предоставляет “связку” сертификатов, которую следует присоединить к сертификату сервера. Сертификат сервера следует разместить перед связкой сертификатов в скомбинированном файле:

```
$ cat www.example.com.crt bundle.crt > www.example.com.chained.crt
```

Полученный файл следует указать в директиве ssl_certificate:

```
server {
    listen              443 ssl;
    server_name         www.example.com;
    ssl_certificate     www.example.com.chained.crt;
    ssl_certificate_key www.example.com.key;
#...
}
```

Если сертификат сервера и связка сертификатов были соединены в неправильном порядке, Angie не запустится и выдаст сообщение об ошибке:

```
SSL_CTX_use_PrivateKey_file(» … /www.example.com.key») failed
```

(SSL: error:0B080074:x509 certificate routines: X509_check_private_key:key values mismatch)

поскольку Angie PRO попытается использовать секретный ключ с первым сертификатом из связки вместо сертификата сервера.

Браузеры обычно сохраняют полученные промежуточные сертификаты, подписанные доверенными центрами сертификации, поэтому активно используемые браузеры уже могут иметь требуемые промежуточные сертификаты и не выдать предупреждение о сертификате, присланном без связанной с ним цепочки сертификатов. Убедиться в том, что сервер присылает полную цепочку сертификатов, можно при помощи утилиты командной строки openssl, например:

```
$ openssl s_client -connect www.godaddy.com:443

Certificate chain
 0 s:/C=US/ST=Arizona/L=Scottsdale/1.3.6.1.4.1.311.60.2.1.3=US
     /1.3.6.1.4.1.311.60.2.1.2=AZ/O=GoDaddy.com, Inc
     /OU=MIS Department/CN=www.GoDaddy.com
     /serialNumber=0796928-7/2.5.4.15=V1.0, Clause 5.(b)
   i:/C=US/ST=Arizona/L=Scottsdale/O=GoDaddy.com, Inc.
     /OU=http://certificates.godaddy.com/repository
     /CN=Go Daddy Secure Certification Authority
     /serialNumber=07969287
 1 s:/C=US/ST=Arizona/L=Scottsdale/O=GoDaddy.com, Inc.
     /OU=http://certificates.godaddy.com/repository
     /CN=Go Daddy Secure Certification Authority
     /serialNumber=07969287
   i:/C=US/O=The Go Daddy Group, Inc.
     /OU=Go Daddy Class 2 Certification Authority
 2 s:/C=US/O=The Go Daddy Group, Inc.
     /OU=Go Daddy Class 2 Certification Authority
   i:/L=ValiCert Validation Network/O=ValiCert, Inc.
     /OU=ValiCert Class 2 Policy Validation Authority
     /CN=http://www.valicert.com//emailAddress=info@valicert.com
```

Примечание

При тестировании конфигураций с SNI необходимо указывать опцию *-servername*, так как openssl по умолчанию не использует SNI.

В этом примере субъект (“s”) сертификата №0 сервера www.GoDaddy.com подписан издателем (“i”), который в свою очередь является субъектом сертификата №1, подписанного издателем, который в свою очередь является субъектом сертификата №2, подписанного общеизвестным издателем ValiCert, Inc., чей сертификат хранится во встроенной в браузеры базе данных сертификатов.

Если связку сертификатов не добавили, будет показан только сертификат сервера №0.

Единый HTTP/HTTPS сервер

Можно настроить единый сервер, который обслуживает как HTTP-, так и HTTPS-запросы:

```
server {
    listen              80;
    listen              443 ssl;
    server_name         www.example.com;
    ssl_certificate     www.example.com.crt;
    ssl_certificate_key www.example.com.key;
#...
}
```

Выбор HTTPS-сервера по имени

Типичная проблема возникает при настройке двух и более серверов HTTPS, слушающих на одном и том же IP-адресе:

```
server {
    listen          443 ssl;
    server_name     www.example.com;
    ssl_certificate www.example.com.crt;
#...
}

server {
    listen          443 ssl;
    server_name     www.example.org;
    ssl_certificate www.example.org.crt;
#...
}
```

В такой конфигурации браузер получит сертификат сервера по умолчанию, т.е. www.example.com, независимо от запрашиваемого имени сервера. Это связано с поведением протокола SSL. SSL-соединение устанавливается до того, как браузер посылает HTTP-запрос, и Angie не знает имени запрашиваемого сервера. Следовательно, он лишь может предложить сертификат сервера по умолчанию.

Наиболее старым и надёжным способом решения этой проблемы является назначение каждому HTTPS-серверу своего IP-адреса:

```
server {
    listen          192.168.1.1:443 ssl;
    server_name     www.example.com;
    ssl_certificate www.example.com.crt;
#...
}

server {
    listen          192.168.1.2:443 ssl;
    server_name     www.example.org;
    ssl_certificate www.example.org.crt;
#...
}
```

SSL-сертификат с несколькими именами

Существуют и другие способы, которые позволяют использовать один и тот же IP-адрес сразу для нескольких HTTPS-серверов. Все они, однако, имеют свои недостатки. Одним из таких способов является использование сертификата с несколькими именами в поле SubjectAltName сертификата, например www.example.com и www.example.org. Однако, длина поля SubjectAltName ограничена.

Другим способом является использование wildcard-сертификата, например *.example.org. Такой сертификат защищает все поддомены указанного домена, но только на заданном уровне. Под такой сертификат подходит www.example.org, но не подходят example.org и www.sub.example.org. Два вышеуказанных способа можно комбинировать. Сертификат может одновременно содержать и точное, и wildcard имена в поле SubjectAltName, например example.org и *.example.org.

Лучше поместить сведения о файле сертификата с несколькими именами и файле с его секретным ключом на уровне конфигурации http, чтобы все серверы унаследовали их единственную копию в памяти:

```
ssl_certificate     common.crt;
ssl_certificate_key common.key;

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

server {
    listen          443 ssl;
    server_name     www.example.org;
#...
}
```

Указание имени сервера

Более общее решение для работы нескольких HTTPS-серверов на одном IP-адресе — расширение Server Name Indication протокола TLS (SNI, RFC 6066), которое позволяет браузеру передать запрашиваемое имя сервера во время SSL handshake, а значит сервер будет знать, какой сертификат ему следует использовать для соединения. Сейчас SNI поддерживается большинством современных браузеров, однако может не использоваться некоторыми старыми или специализированными клиентами.

Примечание

В SNI можно передавать только доменные имена, однако некоторые браузеры могут ошибочно передавать IP-адрес сервера в качестве его имени, если в запросе указан IP-адрес. Полагаться на это не следует.

Чтобы использовать SNI в Angie PRO, соответствующая поддержка должна присутствовать как в библиотеке OpenSSL, использованной при сборке бинарного файла Angie Pro, так и в библиотеке, подгружаемой в момент работы. OpenSSL поддерживает SNI начиная с версии 0.9.8f, если она была собрана с опцией конфигурации *--enable-tlsext*. Начиная с OpenSSL 0.9.8j эта опция включена по умолчанию. Если Angie был собран с поддержкой SNI, то при запуске Angie с ключом “-V” об этом сообщается:

```
$ angie -V
...
TLS SNI support enabled
...
```

Однако если Angie PRO, собранный с поддержкой SNI, в процессе работы подгружает библиотеку OpenSSL, в которой нет поддержки SNI, Angie PRO выдаёт предупреждение:

```
Angie was built with SNI support, however, now it is linked
dynamically to an OpenSSL library which has no tlsext support,
therefore SNI is not available.
```