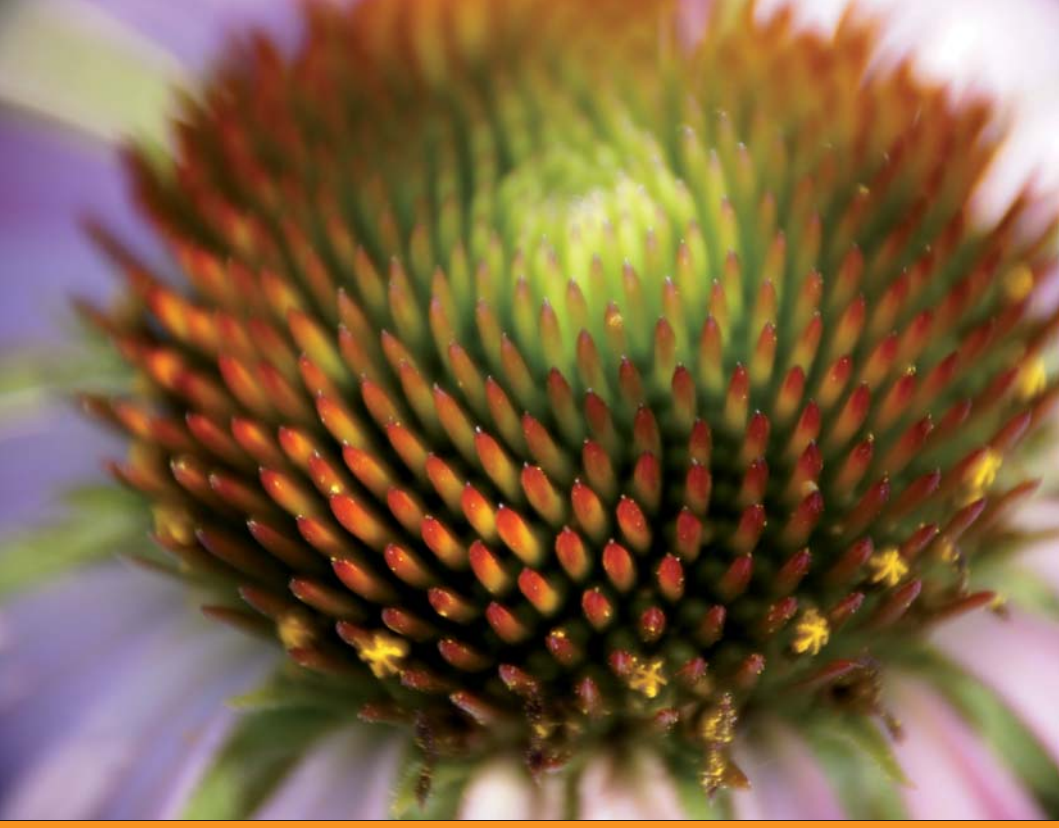


Администрирование сервера NGINX

Подробное руководство по настройке NGINX в любой ситуации, с многочисленными примерами и справочными таблицами для всех директив

Димитрий Айвалиотис

ДМК ПРЕСС издательство

УДК 004.738.5:004.42Nginx
ББК 32.973.202
А36

Айвалиотис Д.

А36 Администрирование сервера NGINX. – М.: ДМК Пресс, 2015. – 288 с.: ил.

ISBN 978-5-94074-162-4

NGINX – это высокопроизводительный сервер, который реализует функции прокси для веб-серверов и почтовых серверов и потребляет очень мало системных ресурсов. В Интернете хватает руководств по его настройке и примеров конфигураций, но при этом трудно понять, как правильно настроить NGINX для конкретных нужд.

Эта книга расчистит мутные воды конфигурирования NGINX и научит вас настраивать его для решения различных задач. Попутно вы узнаете, что означают некоторые покрытые мраком параметры, и поймете, как разработать конфигурацию, отвечающую именно вашим целям.

Вначале дается краткий обзор процедуры компиляции NGINX и описывается формат конфигурационного файла. Затем автор переходит к модулям и рассказывает о многочисленных настройках, позволяющих использовать NGINX в качестве обратного прокси-сервера. Завершается книга обсуждением поиска и устранения неполадок.

Издание предназначено для системных администраторов или инженеров, имеющих опыт эксплуатации веб-серверов.

УДК 004.738.5:004.42Nginx
ББК 32.973.202

ISBN 978-1-84951-744-7 (анг.)
ISBN 978-5-94074-162-4 (рус.)

Содержание

Глава 2. Руководство по настройке

Формат конфигурационного файла NGINX прост и логичен. Понимание его устройства и назначения отдельных секций – одно из условий самостоятельного составления конфигурационных файлов. В этой главе мы постараемся достичь данной цели, рассмотрев следующие вопросы.

- Основы формата конфигурационного файла.
- Глобальные конфигурационные параметры NGINX.
- Включаемые файлы.
- Секция с описанием HTTP-сервера.
- Секция с описанием виртуального сервера.
- Местоположения – где, когда и как.
- Секция с описанием почтового сервера.
- Полный пример конфигурации.

Основы формата конфигурационного файла

Конфигурационный файл NGINX состоит из секций. Секции устроены следующим образом:

```
<секция> {

    <директива> <параметры>;

}
```

Обратите внимание, что строки, содержащие директивы, оканчиваются точкой с запятой (;). Это признак конца строки. Фигурные скобки вводят новый конфигурационный контекст, но мы будем называть такие контексты просто «секциями».

Глобальные конфигурационные параметры NGINX

В глобальной секции задаются параметры, оказывающие влияние на сервер в целом. Его формат отличается от описанного выше. Глобальная секция может включать как конфигурационные директивы, например `user` и `worker_processes`, так и секции, например `events`. Глобальная секция не заключается в фигурные скобки.

В таблице ниже приведены наиболее распространенные директивы, задаваемые в глобальном контексте.

Глобальные конфигурационные директивы

Директива	Описание
`user`	Пользователь и группа, от имени которых исполняются рабочие процессы. Если группа опущена, то подразумевается группа, имя которой совпадает с именем пользователя
`worker_processes`	Количество рабочих процессов, создаваемых сразу после запуска. Эти процессы обрабатывают запросы на соединения со стороны клиентов. Сколько процессов задать, зависит от сервера и в первую очередь от дисковой подсистемы и сетевой инфраструктуры. Если решаются в основном счетные задачи, то рекомендуется задавать этот параметр равным количеству процессорных ядер, а если задачи, требующие интенсивного ввода-вывода, – то умножать это количество на коэффициент от 1,5 до 2
`error_log`	Это файл, в который записываются сообщения об ошибках. Если ни в каком другом контексте директивы `error_log` нет, то в этом файле будут регистрироваться вообще все ошибки. Второй параметр директивы обозначает уровень сообщений, попадающих в журнал (`debug`, `info`, `notice`, `warn`, `error`, `crit`, `alert`, `emerg`). Сообщения уровня `debug` выводятся, только если программа была сконфигурирована с параметром `--with-debug`
`pid`	Файл, в котором хранится идентификатор главного процесса. Переопределяет значение, заданное на этапе конфигурирования и компиляции
`use`	Определяет метод обработки соединения. Переопределяет значение, заданное при компиляции, и если используется, то должна содержаться в контексте `events`. Обычно не нуждается в переопределении, разве что в случае, когда значение по умолчанию приводит к ошибкам

Директива	Описание
worker_connections	Эта директива задает максимальное число соединений, одновременно открытых в одном рабочем процессе. Сюда входят, в частности, соединения с клиентами и с проксируемыми серверами (но не только). Особенно важно это для обратных прокси-серверов – чтобы достичь указанного количества одновременно открытых соединений, может понадобиться настройка на уровне операционной системы

Ниже приведен простой пример задания описанных директив.

```
# мы хотим, чтобы nginx работала от имени пользователя 'www'
user www;

# рабочая нагрузка счетная и имеется 12 процессорных ядер
worker_processes 12;

# явно задаем путь к обязательному журналу ошибок
error_log /var/log/nginx/error.log;

# явно задаем путь к pid-файлу
pid /var/run/nginx.pid;

# создаем конфигурационный контекст для модуля 'events'
events {

    # мы работаем в системе Solaris и обнаружили, что при использовании
    # подразумеваемого по умолчанию механизма обработки соединений nginx
    # со временем перестает отвечать на запросы, поэтому переходим на
    # следующий по качеству механизм
    use /dev/poll;

    # произведение этого числа и значения worker_processes
    # показывает, сколько может быть одновременно открыто соединений
    # для одной пары IP:порт
    worker_connections 2048;
}
```

Глобальная секция должна находиться в начале конфигурационного файла `nginx.conf`.

Включаемые файлы

Включать файлы можно в любое место конфигурационного файла. Их цель – сделать файл более удобным для восприятия и обеспечить повторное использование некоторых частей. Включаемые файлы

должны быть синтаксически корректны с точки зрения записи директив и блоков. Для включения файла нужно указать путь к нему:

```
include /opt/local/etc/nginx/mime.types;
```

Метасимволы в пути позволяют включить сразу несколько файлов:

```
include /opt/local/etc/nginx/vhost/*.conf;
```

Если указан неполный путь, то NGINX считает, что путь задан относительно главного местоположения конфигурационного файла.

Проверить правильность конфигурационного файла можно, запустив NGINX следующим образом:

```
nginx -t -c <path-to-nginx.conf>
```

При этом на наличие ошибок проверяются и все включаемые файлы.

Секция с описанием HTTP-сервера

Секция (или конфигурационный контекст), описывающая HTTP-сервер, доступна, только если при конфигурировании NGINX не был задан параметр `--without-http`, отключающий модуль HTTP. В этой секции описываются все аспекты работы с модулем HTTP – именно с ним вы чаще всего будете иметь дело.

Поскольку конфигурационных директив, относящихся к работе с HTTP-соединениями, много, мы разобьем их на несколько категорий и будем рассматривать каждую категорию в отдельности.

Клиентские директивы

Директивы из этой категории относятся к самому соединению с клиентом, а также описывают некоторые аспекты поведения для клиентов разных типов.

Клиентские директивы модуля HTTP

Директива	Описание
`chunked_transfer_encoding`	Позволяет отключить специфицированный в стандарте HTTP/1.1 механизм поблочной передачи данных (chunked transfer encoding) в ответе клиенту

Директива	Описание
client_body_buffer_size	Задает размер буфера для чтения тела запроса клиента. По умолчанию для буфера выделяются две страницы памяти. Увеличение размера позволяет предотвратить запись во временный файл на диске
client_body_in_file_only	Используется для отладки или последующей обработки тела запроса клиента. Если значение равно on, то тело запроса принудительно записывается в файл
client_body_in_single_buffer	Заставляет NGINX сохранить все тело запроса клиента в одном буфере, чтобы уменьшить количество операций копирования
client_body_temp_path	Определяет путь к каталогу для сохранения файлов с телами запросов клиентов
client_body_timeout	Задает время между последовательными операциями чтения тела запроса клиента
client_header_buffer_size	Задает размер буфера для чтения заголовка запроса клиента, если он превышает подразумеваемую по умолчанию величину 1 КБ
client_header_timeout	Время, отведенное на чтение всего заголовка запроса
client_max_body_size	Максимальный размер тела запроса клиента. В случае превышения отправляется ответ 413 (Request Entity Too Large)
keepalive_disable	Запрещает соединения типа keep-alive для некоторых браузеров
keepalive_requests	Определяет, сколько запросов можно принять по одному соединению типа keep-alive, прежде чем закрывать его
keepalive_timeout	Определяет, сколько времени соединение типа keep-alive может оставаться открытым. Можно задать второй параметр, используемый для формирования заголовка ответа «Keep-Alive»
large_client_header_buffers	Задает максимальное число и размер буферов для чтения большого заголовка запроса клиента
msie_padding	Разрешает или запрещает добавлять комментарии в ответы со статусом больше 400 для увеличения размера ответа до 512 байт при работе с MSIE
msie_refresh	Разрешает или запрещает отправлять MSIE-клиентам ответ Refresh вместо перенаправления

Директивы, относящиеся к вводу-выводу

Эти директивы управляют отправкой статических файлов и порядком работы с файловыми дескрипторами.

Директивы модуля HTTP, относящиеся к вводу-выводу

Директива	Описание
`aio`	Разрешает использование асинхронного файлового ввода-вывода. Это возможно во всех современных версиях FreeBSD и дистрибутивах Linux. В FreeBSD директиву `aio` можно использовать для предварительной загрузки данных для `sendfile`. В Linux требуется директива `directio`, которая автоматически отключает `sendfile`
`directio`	Разрешает использовать зависящий от операционной системы флаг при чтении файлов, размер которых больше или равен указанному. Обязательна при использовании директивы `aio` в Linux
`directio_alignment`	Устанавливает выравнивание для `directio`. Обычно подразумеваемого по умолчанию значения `512` достаточно, но при использовании XFS в Linux рекомендуется увеличить до 4 K
`open_file_cache`	Настраивает кэш, в котором могут храниться дескрипторы открытых файлов, информация о существовании каталогов и информация об ошибках поиска файлов
`open_file_cache_errors`	Разрешает или запрещает кэширование ошибок поиска файлов в кэше `open_file_cache`
`open_file_cache_min_uses`	Задаёт минимальное число обращений к файлу в течение времени, заданного параметром `inactive` директивы `open_file_cache`, необходимое для того, чтобы дескриптор файла оставался в кэше открытых дескрипторов
`open_file_cache_valid`	Задает время между последовательными проверками актуальности данных, хранящихся в кэше `open_file_cache`
`postpone_output`	Задает минимальный размер порции данных, отправляемых клиенту. Если возможно, данные не будут отправляться, пока не накопится указанное количество
`read_ahead`	Если возможно, ядро будет сразу считывать из файла столько байтов, сколько указано в параметре `size`. Поддерживается в текущих версиях FreeBSD и Linux (в Linux параметр `size` игнорируется)

Директива	Описание
sendfile	Разрешает использовать системный вызов sendfile (2) для прямого копирования из одного файлового дескриптора в другой
sendfile_max_chunk	Задает максимальный размер данных, который можно скопировать за один вызов sendfile (2). Без этого ограничения одно быстрое соединение может целиком захватить рабочий процесс

Директивы, относящиеся к хэш-таблицам

Директивы из этой категории управляют выделением статической памяти для определенных переменных. NGINX вычисляет минимально необходимый размер при запуске и после изменения конфигурации. Как правило, достаточно настроить только один из параметров *_hash_max_size с помощью соответствующей директивы. NGINX выдает предупреждение при попытке задать сразу несколько таких параметров. Переменным вида *_hash_bucket_size по умолчанию присваивается значение, кратное размеру строки кэша процессора, чтобы минимизировать количество обращений к кэшу, необходимое для чтения записи. Поэтому изменять их не рекомендуется. Дополнительные сведения см. на странице http://nginx.org/en/docs/hash.html.

Директивы модуля HTTP, относящиеся к хэш-таблицам

Директива	Описание
server_names_hash_bucket_size	Задает размер кластера в хэш-таблицах имен серверов
server_names_hash_max_size	Задает максимальный размер хэш-таблиц имен серверов
types_hash_bucket_size	Задает размер кластера в хэш-таблицах типов
types_names_hash_max_size	Задает максимальный размер хэш-таблиц типов
variables_hash_bucket_size	Задает размер кластера в хэш-таблицах прочих переменных
variables_names_hash_max_size	Задает максимальный размер хэш-таблиц прочих переменных

Директивы, относящиеся к сокетам

Эти директивы описывают установку различных параметров TCP-сокетов, создаваемых NGINX.

Директивы модуля HTTP, относящиеся к сокетам

Директива	Описание
`lingering_close`	Определяет, следует ли оставлять соединение открытым в ожидании дополнительных данных от клиента
`lingering_time`	Связана с директивой `lingering_close` и определяет, сколько времени держать сокет открытым для обработки дополнительных данных
`lingering_timeout`	Также связана с директивой `lingering_close` и определяет, сколько времени держать соединение открытым в ожидании дополнительных данных
`reset_timedout_connection`	Если значение этой директивы равно `on`, то сокеты, для которых истек тайм-аут, сбрасываются немедленно, в результате чего освобождается выделенная для них память. По умолчанию сокет остается в состоянии `FIN_WAIT1`. На соединения типа keep-alive эта директива не распространяется, они всегда закрываются обычным образом
`send_lowat`	Если значение отлично от нуля, то NGINX пытается минимизировать количество операций отправки данных через клиентские сокеты. В Linux, Solaris и Windows эта директива игнорируется
`send_timeout`	Задает тайм-аут между двумя последовательными операциями записи при передаче ответа клиенту
`tcp_nodelay`	Разрешает или запрещает использование параметра `TCP_NODELAY` для соединений типа keep-alive
`tcp_nopush`	Учитывается только при использовании директивы `sendfile`. Разрешает NGINX отправлять заголовки ответа одним пакетом, а также передавать файл полными пакетами

Пример конфигурации

Ниже приведен пример конфигурационной секции модуля HTTP:

```
http {
  include       /opt/local/etc/nginx/mime.types;
  default_type  application/octet-stream;
```

```
  sendfile on;
  tcp_nopush on;
  tcp_nodelay on;
  keepalive_timeout 65;
  server_names_hash_max_size 1024;
}
```

Этот контекстный блок должен располагаться после всех глобальных директив в файле `nginx.conf`.

Секция с описанием виртуального сервера

По соглашению в контексте, начинающемся ключевым словом `server`, находится описание «виртуального сервера». Так называется логический набор ресурсов, сопоставленный со значением директивы `server_name`. Виртуальные серверы отвечают на запросы по протоколу HTTP и потому входят в состав секции `http`.

Виртуальный сервер определяется сочетанием директив `listen` и `server_name`. Директива `listen` задает комбинацию IP-адреса и номера порта либо путь к сокету в домене UNIX:

```
listen address[:port];
listen port;
listen unix:path;
```

Директива `listen` однозначно описывает привязку совета в NGINX. Дополнительно в ней могут присутствовать следующие параметры.

Параметры директивы `listen`

Параметр	Описание	Примечание
`default_server`	Означает, что данный сервер будет сервером по умолчанию для указанной пары `адрес:порт`	
`setfib`	Задает таблицу маршрутизации FIB для прослушивающего сокета	Поддерживается только в ОС FreeBSD. Игнорируется для сокетов в домене UNIX
`backlog`	Задает параметр `backlog` в системном вызове `listen()`	По умолчанию равен –1 в FreeBSD и 511 на всех остальных платформах
`rcvbuf`	Задает параметр `SO_RCVBUF` для прослушивающего сокета	

Параметр	Описание	Примечание
sndbuf	Задает параметр SO_SNDBUF для прослушивающего сокета	
accept_filter	Задает имя фильтра приема: dataready или httpready	Поддерживается только в ОС FreeBSD
deferred	Задает параметр TCP_DEFER_ACCEPT, означающий, что требуется отложить вызов accept()	Поддерживается только в ОС Linux
bind	Означает, что для данной пары адрес:порт нужен отдельный вызов bind()	Отдельный вызов bind() производится без специального указания, если для сокета заданы какие-то другие специальные параметры
ipv6only	Устанавливает значение параметра IPV6_V6ONLY	Можно установить только один раз при запуске. Игнорируется для сокетов в домене UNIX
ssl	Означает, что этот порт предназначен только для соединений по протоколу HTTPS	Позволяет построить более компактную конфигурацию
so_keepalive	Для прослушивающего сокета задается режим TCP keepalive	

Несмотря на свою простоту, директива server_name позволяет решить целый ряд задач конфигурирования. По умолчанию ее значение равно "", то есть секция server без директивы server_name сопоставляется с запросом, в котором отсутствует заголовок Host. Этим можно воспользоваться, например, для отбрасывания запросов без этого заголовка:

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

Нестандартный код ответа HTTP 444, использованный в этом примере, заставляет NGINX немедленно закрыть соединение.

Помимо обычной строки, NGINX допускает использование в директиве server_name метасимвола *:

- метасимвол можно указывать вместо поддомена: `*.example.com`;
- метасимвол можно указывать вместо домена верхнего уровня: `www.example.*`;
- существует особая форма, которая соответствует поддомену или самому домену: `.example.com` (соответствует как `*.example.com`, так и `example.com`).

Параметр директивы `server_name` может быть и регулярным выражением, для этого нужно лишь предпослать имени знак тильды (~):

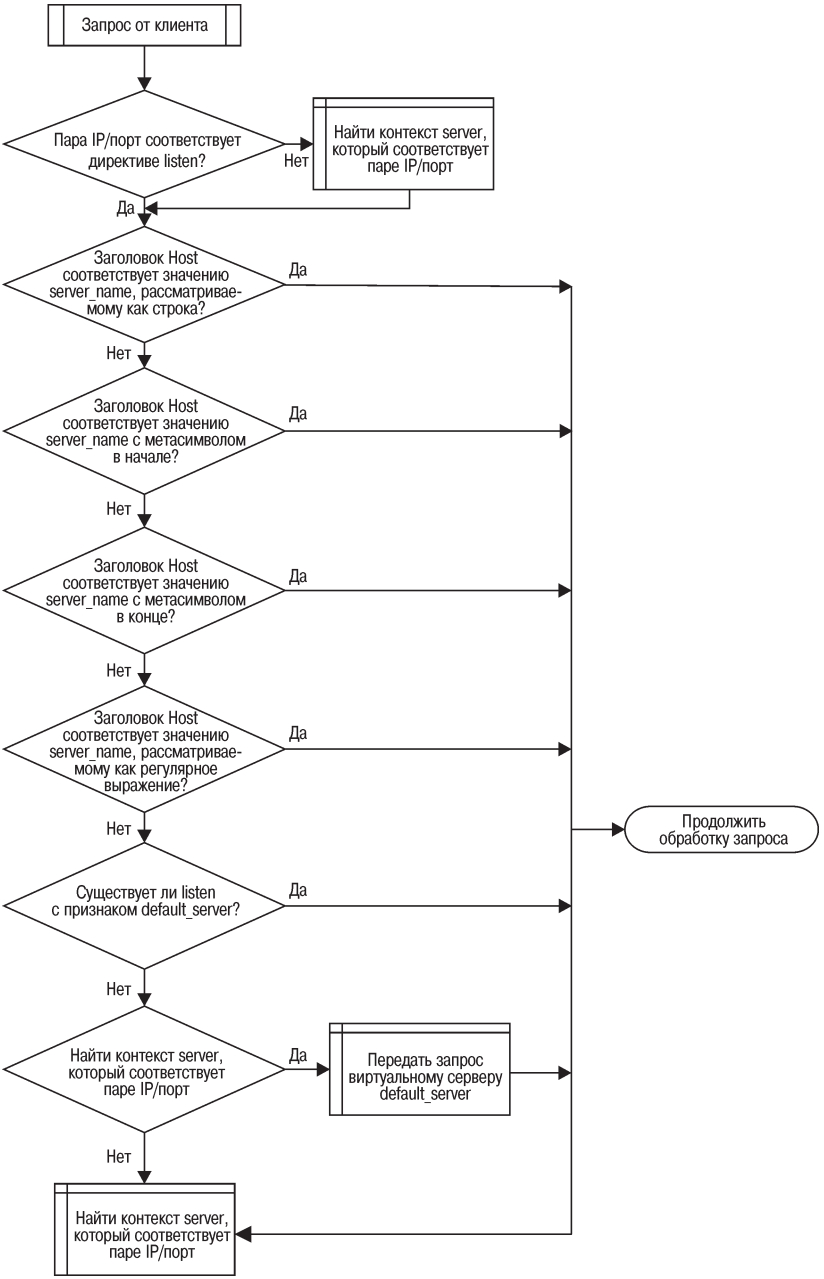
```
server_name ~^www\.example\.com$;
server_name ~^www(\d+).example\.(com)$;
```

Вторая форма применяется для запоминания подвыражений, на которые затем можно ссылаться (по номеру `$1`, `$2` и т. д.) в последующих директивах.

Чтобы определить, какой виртуальный сервер должен обслужить данный запрос, NGINX применяется следующий алгоритм.

1. Сопоставить IP-адрес и порт с указанными в директиве `listen`.
2. Сопоставить заголовок Host со значением директивы `server_name`, рассматриваемым как строка.
3. Сопоставить заголовок Host со значением директивы `server_name`, рассматриваемым как строка, считая, что в начале находится метасимвол *.
4. Сопоставить заголовок Host со значением директивы `server_name`, рассматриваемым как строка, считая, что в конце находится метасимвол *.
5. Сопоставить заголовок Host со значением директивы `server_name`, рассматриваемым как регулярное выражение.
6. Если все попытки сопоставления заголовка Host закончились неудачей, использовать ту директиву `listen`, в которой имеется признак `default_server`.
7. Если все попытки сопоставления заголовка Host закончились неудачей и директивы `listen` с признаком `default_server` не существует, использовать первый сервер, в котором директива `listen` удовлетворяет условию шага 1.

Эта логика изображена на следующей блок-схеме:


Запрос от клиента
Пара IP/порт соответствует директиве listen?
Нет
Найти контекст server, который соответствует паре IP/порт
Да
Заголовок Host соответствует значению server_name, рассматриваемому как строка?
Да
Нет
Заголовок Host соответствует значению server_name с метасимволом в начале?
Да
Нет
Заголовок Host соответствует значению server_name с метасимволом в конце?
Да
Нет
Заголовок Host соответствует значению server_name, рассматриваемому как регулярное выражение?
Да
Нет
Существует ли listen с признаком default_server?
Да
Нет
Найти контекст server, который соответствует паре IP/порт
Да
Передать запрос виртуальному серверу default_server
Нет
Найти контекст server, который соответствует паре IP/порт
Продолжить обработку запроса

Признак `default_server` позволяет обработать запросы, которые иначе остались бы необработанными. Поэтому рекомендуется всегда задавать этот признак явно, чтобы не гадать потом, почему запросы обрабатываются странным образом.

Кроме того, признак `default_server` полезен, когда требуется сконфигурировать несколько виртуальных серверов с одной и той же директивой `listen`. Все описанные выше директивы будут одинаковы для всех подходящих блоков `server`.

Местоположения – где, когда и как

Директива `location` может встречаться в секции с описанием виртуального сервера и содержит в качестве параметра URI-адрес, поступивший от клиента или в результате внутренней переадресации. Местоположения могут быть вложенными (с несколькими исключениями). Их назначение – определить максимально специализированную конфигурацию для обработки запроса.

Местоположение задается следующим образом:

```
location [модификатор] uri {...}
```

Можно задавать также именованные местоположения:

```
location @name {…}
```

Именованное местоположение доступно только при внутренней переадресации и может быть задано лишь на уровне контекста сервера. При этом сохраняется тот URI, который был перед входом в блок `location`.

Модификаторы изменяют обработку местоположения следующим образом:

Модификаторы местоположения

Модификатор	Обработка
=	Сравнить буквально и завершить поиск
~	Сопоставление с регулярным выражением с учетом регистра
~*	Сопоставление с регулярным выражением без учета регистра
^~	Прекратить обработку до сопоставления этого местоположения с регулярным выражением, если это совпадение с самым длинным префиксом. Отметим, что это не сопоставление с регулярным выражением, задача данного модификатора – как раз предотвратить такое сопоставление