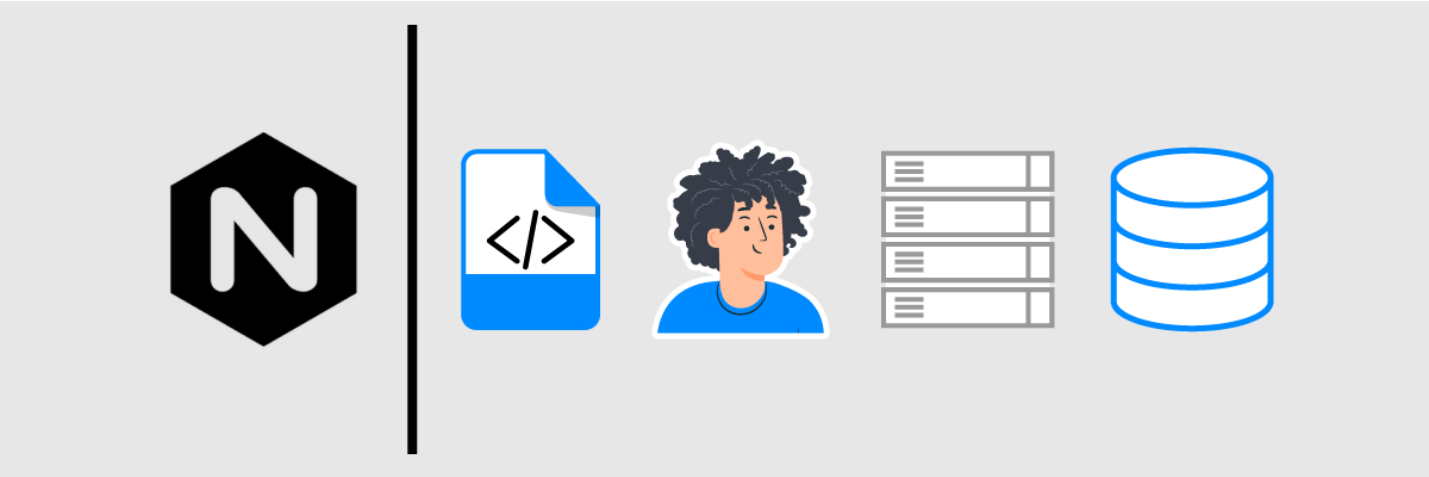
Nginx: принципы работы и настройка

06.06.2022

Nginx обслуживает более миллиона сайтов по всему миру и пользуется заслуженной любовью и почитанием. Однако несмотря на его популярность и большое количество материалов, вопросов по его использованию не убавляется.

В этой статье мы ответим на вопросы о том, как работает Nginx и как он читает конфигурационные файлы. А в следующем посте мы детально разберём 4 кейса его использования: от хостинга нескольких сайтов на одном сервере, до защиты от DDOS.

Коротко о Nginx



Nginx – это веб-сервер, прокси-сервер, обратный прокси-сервер, smtp-сервер и балансировщик нагрузки. Сразу раскроем все эти понятия. Магия перестает быть магией, когда понимаешь как устроен мир.

Веб-сервер – это программа, которая принимает и обрабатывает запросы от клиентов по протоколам HTTP и HTTPS и возвращает им ответ в виде HTML-страницы. Прокси-сервер принимает и обрабатывает запросы клиентов, а затем передает их дальше, другим программам. Обратный прокси-сервер – принимает результат работы других серверов и отдаёт его клиентам. Smtp-сервер – это сервер почтовой службы. Балансировщик нагрузки – программа, которая распределяет сетевые запросы между серверами, следуя настройкам балансировки.

NGINX сочетает в себе все перечисленные возможности, хотя изначально он задумывался только как web- и smtp-сервер. NGINX был разработан Игорем Сысоевым в 2004 году. Его не устраивала скорость работы Apache. Ему нужен был web-сервер, который мог бы держать 10,000 одновременных запросов, при этом расходовать минимум памяти, не теряя производительности.

Сегодня с помощью NGINX решают следующие задачи:

- Кеширование и стриминг видео
- Отдача статического контента сайта
- Построение CDN-сетей
- Распределение нагрузки
- Обеспечение безопасности
- Работа почтовых служб

и многое другое.

Секрет успеха NGINX в его архитектуре, которая коренным образом отличается от других веб-серверов. Давайте разберёмся, как работает NGINX?

Как работает NGINX?

Многие веб-серверы построены на простой многопоточной модели, NGINX использует событийную архитектуру, которая позволяет ему масштабироваться до сотен тысяч параллельных соединений. Ключевое – это то, что NGINX обрабатывает множество соединений в одном процессе.

У NGINX есть один мастер-процесс и несколько вспомогательных процессов. Например, на нашем 2 ядерном сервере, NGINX запустил 2 дополнительных процесса, обозначенных как worker process:

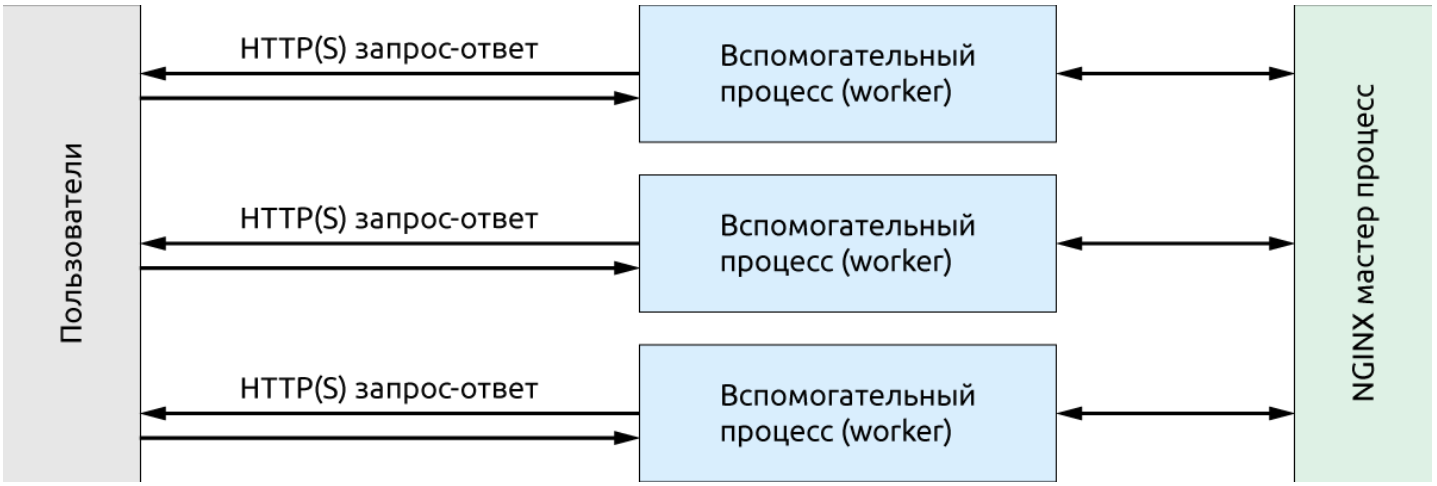
```
root@SRVN6KU737P3E:~# systemctl status nginx
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2022-05-04 11:34:14 UTC; 2 weeks 5 days ago
     Docs: man:nginx(8)
 Main PID: 126039 (nginx)
    Tasks: 3 (limit: 2287)
   CGroup: /system.slice/nginx.service
           ├─126039 nginx: master process /usr/sbin/nginx -g daemon on; master_process on;
           ├─126040 nginx: worker process
           └─126041 nginx: worker process
```

Мы не раз уже упомянули такие понятия как процесс и поток. Процесс или поток (с точки зрения ОС это одно и то же, разница лишь в разделении адресного пространства) – это самодостаточный набор инструкций, который операционная система может запланировать для выполнения на ядре процессора.

На одном ядре одновременно может находиться только один процесс или поток. Процессы на ядре постоянно подменяют друг друга, из-за чего сильно страдает производительность, когда процессов становится много. Это камень преткновения для многих современных web-приложений, которые открывают на каждое соединение свой поток.

Такая архитектура легка в реализации, однако плохо масштабируется, когда количество соединений сильно возрастает. NGINX использует модель с фиксированным числом процессов, это позволяет максимально эффективно использовать ресурсы сервера.

Графически схема работы NGINX выглядит следующим образом:



Структурно её можно описать так: все потоки управляются одним рабочим процессом, процесс содержит меньшие блоки – соединения. Соединения доставляют запросы к рабочему процессу, который также отправляет его в главный процесс. Наконец, основной процесс предоставляет результат этих запросов.

Если хотите больше узнать о работе NGINX изнутри – на хабре есть отличная статья: «NGINX изнутри: рожден для производительности и масштабирования» <https://habr.com/ru/post/260065/> . Она, правда, старая, но очень хорошая. Рекомендуем!

Благодаря такому принципу работы NGINX может обрабатывать до 1024 одновременных запросов, поэтому он отлично подходит для загруженных веб-сайтов, таких как интернет-магазины, поисковые системы и облачное хранилище, а ещё NGINX использует модульную систему, благодаря которой может расширять свои функции.

Модули настраиваются через конфигурационные файлы NGINX, о них и поговорим.

Работа с конфигурационными файлами NGINX

NGINX работает с двумя уровнями конфигурационных файлов. Первый уровень – глобальный, к нему относится конфигурационный файл etc/nginx/nginx.conf. Второй уровень – локальный, к нему относятся конфигурационные файлы конкретных сайтов, расположенных, как правило, в /etc/nginx/site-available или в /etc/nginx/conf.d/

nginx.conf – это корневой конфигурационный файл NGINX. В нём содержатся его глобальные настройки:

- пользователь, от чьего имени будет запускаться Nginx;
- расположение pid-файла;
- сколько рабочих процессов будет запущено;
- ограничения на количество соединений;
- условия сжатия контента

и многое другое.

С nginx.conf NGINX начинает парсить конфигурационные файлы, которые состоят из директив. Директивы могут быть простыми – однострочными, а могут быть блочными. Если блочная директива содержит другую вложенную блочную директиву, то такая блочная директива называется контекстом.

Вот пример простой однострочной директивы: access_log /var/nginx/access.log main; Параметры директивы разделяются пробелами, а в конце строки обязательно ставится точка с запятой.

Пример простой блочной директивы может выглядеть так:

```
events {
  worker_connections 1024;
}
```

Здесь контекст events содержит одну директиву – worker_connections, которая указывает на максимальное число одновременных соединений. Контекст может содержать вложенные контексты.

Вот пример стандартного контекста:

```
server {
    location / {
        root /data/www;
    }

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

Здесь блочная директива server содержит несколько блочных директив location, организуя тем самым контекст server. Директивы вне любого контекста относятся к main.

Список основных контекстов и их значений:

- main – корень конфигурации, по сути это /etc/nginx/nginx.conf;
- events – контекст работы с соединениями;
- http – контекст работы с http-протоколом;
- server – контекст работы с сервером;
- location – контекст работы с маршрутизацией запросов.

Контекст может быть указан не явно, а через директиву include с указанием директории расположения. Это делается для экономии места и возможности отдельного изменения параметров директив server. Вот пример, как это может быть исполнено:

```
user www www;
worker_processes 5;
```

```
error_log logs/error.log;
pid logs/nginx.pid;
worker_rlimit_nofile 8192;

events {
  worker_connections  4096;
}

http {
  include conf/mime.types;
  include /etc/nginx/sites-enabled/*;
  index index.html index.htm index.php;

  default_type application/octet-stream;
  access_log logs/access.log  main;
  sendfile on;
  tcp_nopush on;
  }
```

Здесь указаны сразу две директивы include: первая директива ссылается на мимотипы файлов для отдачи, а вторая на все конфигурационные файлы расположенные в директории /etc/nginx/sites-enabled/ . То есть, при чтении конфигурационного файла, include разворачивается в то, на что ссылается.

В нашем случае вместо include /etc/nginx/sites-enabled/* будут подставлены все конфигурационные файлы в директории /etc/nginx/sites-enabled/ , выглядящие следующим образом:

```
server {
    listen 80;
    server_name  domain2.com www.domain2.com;

    location /  {
      root    /var/www/virtual/big.server.com/htdocs;
      index index.html;
      try file $uri $uri/index.html;
    }
}
```

Благодаря такой древовидной модели сборки конфигурационных файлов удаётся достичь гибкости управления сайтами. Например, если на одном сервере располагаются несколько сайтов и один из них нужно отключить или изменить у него настройки – достаточно выполнить нужные работы и перечитать конфигурационный файл командой `nginx -s reload`.

Структуру конфигурационных файлов разобрали, осталось ответить ещё на один вопрос – как Nginx понимает, что нужно отдавать клиенту и по какому запросу?

Nginx анализирует HTTP-запрос от клиента и ищет совпадения со значениями условий директив server_name и linsten в контексте server. Когда совпадения найдены начинает работать контекст location, который отвечает за маршрутизацию запросов.

В обработке контекстов location Nginx тоже следует определённой иерархии обработки. Так совпадения с условиями обработки location будут искаться сначала среди префиксных location (location, не содержащий регулярных выражений), затем среди location с регулярными выражениями в порядке их следования в конфигурационном файле.

Если среди location с регулярными выражениями Nginx не найдёт совпадений, он вернёт первый префиксный location – корень сайта.

Итак, мы разобрались с принципами работы NGINX, поняли как устроены его конфигурационные файлы и как NGINX их читает. Давайте соберем все новые знания в единое целое.

Расставляем всё по полочкам

NGINX – это легковесный и мощный web-сервер, который способен держать до 10 тысяч одновременных соединений, расходуя при этом минимальное количество ресурсов. Чаще всего его используют в качестве прокси-сервера и обратного прокси-сервера.

NGINX обладает модульной архитектурой. Модули настраиваются через конфигурационные файлы, которые имеют древовидную структуру и собираются в единое целое при их чтении. Корневой конфигурационный файл NGINX – /etc/nginx/nginx.conf . Его NGINX читает в первую очередь, затем он читает другие конфиги, указанные в nginx.conf явным образом или с помощью директивы include.

Корневой конфиг содержит глобальные настройки NGINX: пользователь от чьего имени он будет запускаться, расположение pid-файла, количество рабочих процессов, ограничения на количество соединений и многое другое.

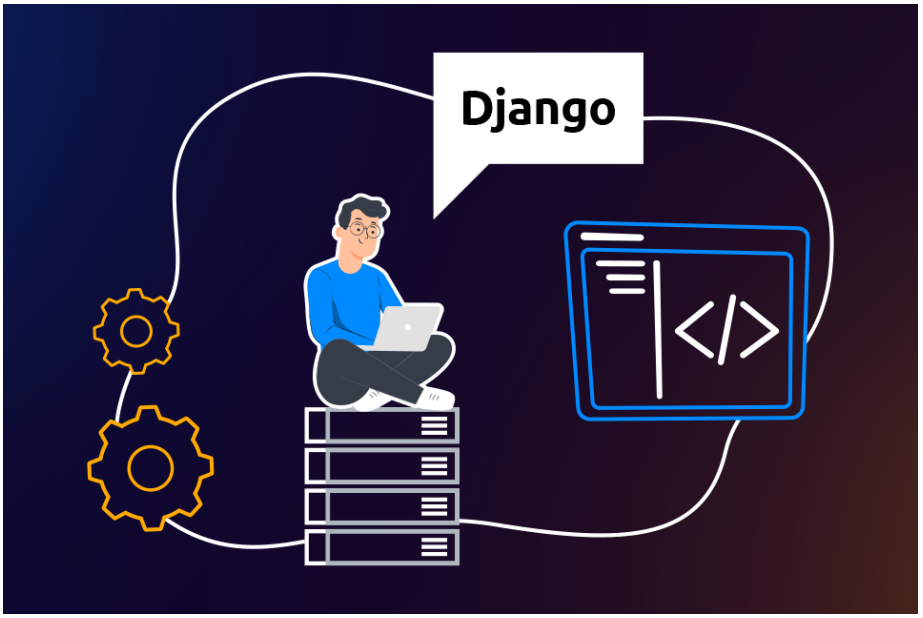
Дочерние конфигурационные файлы, как правило, называются по имени сайтов, которые обслуживает NGINX и содержат контекст работы с сервером и контекст работы с маршрутизацией запросов.

Модульное устройство и древовидная система загрузки конфигурационных файлов сделали NGINX одним из самых популярных web-серверов в мире, благодаря этому мы можем брать и использовать готовые решения и лучшие практики.

В следующей статье мы приведем такие кейсы использования NGINX:

- Как разместить несколько сайтов на одном сервере?
- Связка NGINX и Python – современный сайт на Django
- Как защитить сайт от DDoS?
- Балансировка нагрузки между бекенд-серверами.

Пока мы готовим новый материал, вы можете познакомиться поближе с NGINX. На сайте https://nginx.org/ru/ <https://nginx.org/ru/> доступным языком подана официальная документация и есть хорошие примеры конфигурационных файлов. Также для более глубокого понимания следующей статьи пригодятся знания по Django, Linux и Docker. Их можно взять из наших статей:



<https://1cloud.ru/blog/django_one_server>

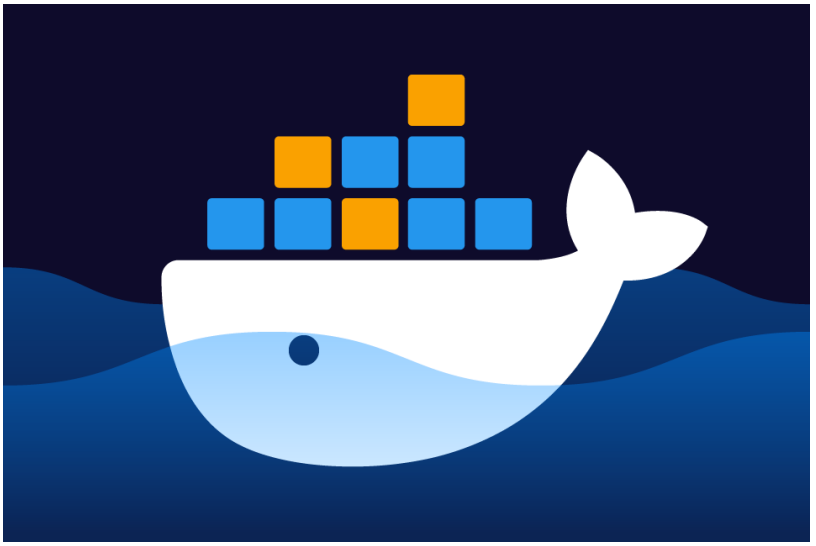
Django-стек <https://1cloud.ru/blog/django_one_server>

Разберёмся как работает Django, что такое Application-сервер и как связать Nginx и Gunicorn.

<https://1cloud.ru/blog/linux_in_windows_10>

Linux внутри Windows <https://1cloud.ru/blog/linux_in_windows_10>

Покажем как легко установить Linux внутрь Windows и научим работать в WSL2.



<https://1cloud.ru/blog/docker_start>

Введение в Docker <https://1cloud.ru/blog/docker_start>

Расскажем, что такое Docker, из каких элементов он состоит и как работает.

ЗАРЕГИСТРИРОВАТЬСЯ <HTTPS://AUTH.1CLOUD.RU/REGISTER>