GeekBrains

Введение в UNIX-системы

Практика. Запускаем свой веб-сервер

База данных MySQL, работа в консоли и через phpmyadmin. Установка и настройка веб-сервера Apache2 и работа с PHP. Веб-сервер Nginx, режимы работы, установка и настройка, модули php_fpm.

Оглавление

Введение

- Веб-серверы

Проверка сети

- IP-адреса, маски, MAC-адреса
- Клиент-серверное взаимодействие
- Минимум для работы TCP/IP

Основы Web

- Протокол HTTP
- Немного об Apache2 и Nginx

База данных MySQL

Знакомство с Apache2

- Конфигурационные файлы, conf-available/conf-enabled
- Модули Apache2, mods-available и mods-enabled
- Виртуальные хосты, sites-available и sites-enabled
- Создаем простейший конфигурационный файл
- Работа с модулями
  - Пример 1. Модуль активен
  - Пример 2. Модуль установлен, но не активен
  - Пример 3. Модуль не установлен
- Виртуальные хосты (много сайтов на одном сервере)
  - Многосайтовость по порту
  - Многосайтовость по IP-адресу
  - Многосайтовость по домену

Устанавливаем Nginx

- Конфигурация nginx
- Настроим работу с PHP
- Настраиваем наш сайт

Практическое задание

Дополнительные материалы

Введение

Для работы с сетевыми сервисами нужно понимать основы TCP/IP-стека, которые мы проходили на пятом уроке данного курса.

У вас уже есть минимум знаний о работе в сети, IP-адресе и его отличиях от MAC-адреса, TCP- и UDP-портах. Вы можете проверить открытые порты на сервере и уже изучили, как можно защититься, разрешив трафик только на те сервисы, которые вы планируете использовать для обслуживания пользователей извне.

Чтобы реализовать полноценный веб-сервер LAMP, надо научиться устанавливать и настраивать стек технологий: Linux, Apache2 и PHP. Дополнительно потребуется база данных MySQL, MariaDB или Postgres.

Кроме LAMP есть LEMP, или LNMP (Linux +[Enginx]* + MySQL + PHP, или Linux + Nginx + MySQL + PHP), Nginx (произносится как [Энджинкс]), а также LNAMP (Linux + Nginx + Apache + MySQL + PHP).

Сегодня научимся настраивать полноценный веб-сервер с базами данных.

Настраивать веб-сервер будем на подготовленном виртуальном сервере в облаке GCP. Но если у вас нет возможности запустить облако GCP, то можно установить виртуальную машину в VirtualBox или приобрести виртуальный сервер VDS (например, тут: https://firstbyte.ru/?from=624).

Веб-серверы

Веб-сервер — это служба или сервис, принимающий HTTP-запросы и выдающий ответы. Как правило, они содержат HTML-код, но могут присутствовать и двоичные файлы.

Apache2 — популярный и, пожалуй, самый известный веб-сервер, но есть и другие:

- lighthttpd — используется во встраиваемом оборудовании с Linux на борту;
- Nginx — еще один популярный веб-сервер, часто работающий в связке с Apache;
- Google Web Server — вариация Apache2 от Nginx;
- Microsoft IIS — используется в основном на Windows Server.

Технически веб-приложение не обязательно должно использовать отдельный веб-сервер. Возможна реализация протокола HTTP внутри самого приложения. Такие решения могут применяться для встраиваемых систем, и по большей части используется тот или иной веб-сервер.

LAMP — Linux + Apache2 + MySQL +PHP — самая популярная конфигурация сервера, ориентированного на выдачу веб-содержимого. Apache2 может использоваться и в Windows — как для сервера, так и для локальной разработки (такая конфигурация называется WAMP).

Проверка сети

IP-адреса, маски, MAC-адреса

Подробно говорили об этом в пятом уроке по сетям TCP/IP — вспомним немного.

Упрощенно можно считать, что IPv4-адрес — это 4 байта (октета), записанные в десятичной системе. Пример: 192.168.0.1 или 8.8.8.8. Они используются как адреса хостов, сетей и в качестве broadcast-адресов (широковещательных).

В классовой адресации первый адрес из диапазона будет адресом сети, а последний — broadcast-адресом. Маска позволяет определить по адресу, к какой сети он относится.

Пример: если для адреса 192.168.1.1 указана маска 255.255.255.0, это сеть 192.168.1.0, а широковещательный адрес — 192.168.1.255.

Для адреса 192.168.1.1 может быть задана и маска 255.255.0.0. Тогда адресом сети будет уже 192.168.0.0, а broadcast`ом — 192.168.255.255. При этом адреса 192.168.1.0. и 192.168.1.255 могут быть адресами хостов, как и 192.168.1.1. Существует еще более сложная система адресации — бесклассовая (разбирается в курсе TCP/IP).

IP-адреса на машине можно посмотреть с помощью команды **ifconfig**:

```
user@ubuntu:~$ ifconfig
ens33     Link encap:Ethernet  HWaddr 00:0c:29:29:80:b7
          inet addr:192.168.116.129  Bcast:192.168.116.255  Mask:255.255.255.0
          inet6 addr: fe80::da89:c6b5:3098:4d72/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:127500 errors:0 dropped:0 overruns:0 frame:0
          TX packets:342871 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:147726823 (147.7 MB)  TX bytes:302244447 (302.2 MB)
          Interrupt:19 Base address:0x2000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:1999 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1999 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:665887 (665.8 KB)  TX bytes:665887 (665.8 KB)
```

Но в современных версиях ОС Linux лучше использовать команду **ip addr:**

```
user@ubuntu:~$ ip addr
9: ens33: <BROADCAST,MULTICAST,UP> mtu 1500 group default qlen 1
    link/ether 00:0c:29:29:80:b7
    inet 192.168.116.129/24 brd 192.168.116.255 scope global dynamic
       valid_lft forever preferred_lft forever
    inet6 fe80::da89:c6b5:3098:4d72/64 scope link dynamic
       valid_lft forever preferred_lft forever
1: lo: <LOOPBACK,UP> mtu 1500 group default qlen 1
    link/loopback 00:00:00:00:00:00
    inet 127.0.0.1/8 brd 127.255.255.255 scope global dynamic
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host dynamic
       valid_lft forever preferred_lft forever
```

Здесь видим два сетевых интерфейса: **ens33** — Ethernet, с MAC-адресом 00:0c:29:29:80:b7, IP-адресом 192.168.116.129 и маской 255.255.255.0. Отсюда можем узнать, что хост принадлежит к сети 192.168.116.0. Есть статистика по сетевому интерфейсу, информация о локальном IPv6-адресе, MTU. Это размер данных кадра — фактически IP-пакета — в который могут быть упакованы данные TCP-сегмента или UDP-дейтаграммы. Он не может быть превышен.

Видим интерфейс локальной петли. Все адреса в сети 127.0.0.1 могут идентифицировать только конкретную машину и не уходят в сеть. Любая посылка, отправленная на адрес 127.0.0.1, не покинет компьютер (виртуальную машину). Такую адресацию используют для взаимодействия между локальными приложениями (Nginx обращается к Apache2, который слушает TCP-порт 80 по адресу 127.0.0.1; PHP обращается к MySQL, который слушает TCP-порт 3306 по адресу 127.0.0.1 и так далее).

Неформально адреса делят на «белые», которые могут маршрутизироваться в интернете (например, 8.8.8.8, 5.255.255.55), и «серые», которые не обладают такой способностью и используются в локальных сетях. Две разные локальные сети могут использовать одни и те же диапазоны «серых» адресов: большинство DSL-модемов предоставляет по DHCP адреса в сети 192.168.1.1/255.255.255.0, а многие Wi-Fi-роутеры — в сети 192.168.0.1/255.255.255.0.

Тем не менее даже компьютеры с «серыми» адресами могут выходить в интернет:

- если используется прокси-сервер — тогда будет доступен только интернет, так как все остальные TCP/IP-службы будут ограничены локальной сетью. Даже если веб-адрес будет доступен по IP 5.255.255.55, команда **ping** 5.255.255.55 покажет, что связи нет;
- если используется NAT — точнее, NAPT, которая будет подменять «серый» адрес, предоставленный провайдером, на «белый» адрес провайдера.

Команда **ping** позволяет проверить доступность узла:

```
user@ubuntu:~$ ping -c 4 5.255.255.55
PING 5.255.255.55 (5.255.255.55) 56(84) bytes of data.
64 bytes from 5.255.255.55: icmp_seq=1 ttl=128 time=41.1 ms
64 bytes from 5.255.255.55: icmp_seq=2 ttl=128 time=40.7 ms
64 bytes from 5.255.255.55: icmp_seq=3 ttl=128 time=41.3 ms
64 bytes from 5.255.255.55: icmp_seq=4 ttl=128 time=41.3 ms
--- 5.255.255.55 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3006ms
rtt min/avg/max/mdev = 40.798/41.168/41.375/0.222 ms
user@ubuntu:~$
```

Ключ **-c** указывает количество отправок. Если запустить без него, **ping** будет осуществляться до нажатия Ctrl+C.

ping с ключом **-b** позволяет сделать **ping** всех узлов в сети, используя broadcast-адрес:

```
# ping -b 192.168.116.255
```

Ответят все компьютеры, на которых не заблокирован ICMP-протокол, или используемые ICMP UDP-порты.

Клиент-серверное взаимодействие

По большей части в сетевых подключениях фигурирует клиент-серверное взаимодействие. Программа-клиент, используя стек TCP/IP, обращается к программе-демону и получает (а иногда отправляет) контент.

При этом серверы — это программы-демоны. Они постоянно находятся в памяти и при запуске, которым управляет (в нашем случае) **systemd**, занимают TCP- или UDP-порт и слушают. В случае с UDP — ожидают приема сообщений, с TCP-портом — сессии. Для этого сервер открывает соответствующие сокеты (STREAM для TCP или DGRAM для UDP) и слушает.

Перечень прослушиваемых TCP-портов на машине можно посмотреть с помощью команды **netstat**:

```
# netstat -tl
```

Просмотреть список прослушиваемых UDP-портов:

```
# netstat -ul
```

Чтобы узнать номера портов без расшифровки (только номер порта), добавляем **n**. Опции можно комбинировать:

```
# netstat -ntul
```

Если хотим посмотреть список соединений во всех состояниях на текущий момент, добавляем **a**:

```
# netstat -ntua
```

Просмотреть список прослушиваемых UNIX-сокетов:

```
# netstat -x
```

Обратите внимание, что **netstat** — это не сканирование портов. Он показывает непосредственно информацию от ядра операционной системы. Для скана портов понадобится утилита **nmap**. Сначала ее надо установить:

```
# apt install nmap
```

Теперь можно посмотреть, какие порты открыты на общедоступном сервере:

```
# nmap vk.com
```

Скорее всего, увидим порты 80 и 443 для веб-сервера. И это правильно, ведь лишний открытый порт — это риск проблем с безопасностью сервера (особенно если это конфигурация по умолчанию).

Доменное имя преобразуется с помощью службы имен DNS:

```
# dig vk.com
root@ubuntu:/etc/openvpn/ccd# dig vk.com
; <<>> DiG 9.10.3-P4-Ubuntu <<>> vk.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 39370
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; MBZ: 0005 , udp: 512
;; QUESTION SECTION:
;vk.com.                                IN      A
;; ANSWER SECTION:
vk.com.                 5       IN      A       87.240.165.82
vk.com.                 5       IN      A       95.213.11.180
;; Query time: 43 msec
;; SERVER: 127.0.1.1#53(127.0.1.1)
;; WHEN: Fri Apr 14 23:59:46 MSK 2017
;; MSG SIZE  rcvd: 67
```

Каждый раз при обращении к браузеру из URL берется следующая информация: http://site.ru/path/file?param=something.

Первая часть — протокол:

- **http** — нешифрованый http, порт по умолчанию — 80;
- **https** — шифрованый http (http, упакованный в tls), порт 443;
- **ftp** — нешифрованый ftp, порт по умолчанию — 21.

Других протоколов браузеры обычно не знают.

Вторая часть — адрес сайта:

- **доменное имя** — преобразуется в IP-адрес. Именно на него будет установлена TCP-сессия;
- может быть сразу **IP-адрес** — например, http://192.168.0.1/admin.

Третья часть — путь (как правило, ресурса):

- **он будет передан на сервер** — это может быть действительно путь в файловой системе;
- **может включать параметры** — например, **?param=something**, которые обработает php;
- **может быть полностью произвольным** — и с помощью настроек Apache2 или Nginx переадресовываться скрипту (например, **index.php**), который разберет путь по собственным правилам.

Иногда (очень редко) используемый TCP-порт может быть указан явно.

Пример: http://site.ru:8080/music.mp3 означает, что, несмотря на использование http, будет подключен порт не 80, а 8080. Это может быть сервер радиостанции (например, icecast2). Соответственно, если TCP-сегменты приходят на порт 80, их получает приложение Apache2, если на 8080 — Icecast. Сами приложения используют динамические порты в верхнем диапазоне.

IANA зарезервировала для использования в качестве динамических порты 49152–65535. На практике этот диапазон сильно разнится в зависимости от того, в какой операционной системе используется. В Linux крайние значения диапазона можно посмотреть с помощью этой команды:

```
root@ubuntu:/etc/openvpn/ccd #cat /proc/sys/net/ipv4/ip_local_port_range
32768   60999
```

Отличие динамических портов от используемых серверами в том, что прослушиваемый порт может принимать много соединений, а браузер, подгружая несколько картинок одной страницы, использует несколько TCP-соединений — и на каждое будет отдельный динамический порт. Это легко посмотреть с помощью **tcpdump** или **wireshark**.

	Уровень	**Адреса**	**Что адресуем**
4	Прикладной уровень	URL, email-адреса (как правило, на основе доменных имен), длина переменная в символьном виде	Пользователей, машины, ресурсы, приложения и т .д.
3	Транспортный уровень	TCP-порты и UDP-порты, длина — два байта	Приложения (процессы)
2	Сетевой уровень	IP-адреса. Для версии 4 —длина 4 байта	Хосты
1	Уровень сетевых интерфейсов	MAC-адреса. Длина — 6 байт	Сетевые интерфейсы в локальной сети

Минимум для работы TCP/IP

Для работы в TCP/IP в целом (и в интернете в частности) у машины должны быть следующие настройки:

- MAC-адрес у сетевой карты (присутствует по умолчанию);
- IP-адрес (даже «серый», если шлюз использует NAT);
- маска сети;
- маршрут по умолчанию (IP-адрес шлюза);
- IP-адреса одного или двух DNS-серверов для преобразования доменных имен в IP-адреса (часто DNS-сервером выступает шлюз, но могут быть указаны публичные кеширующие DNS-сервера от Google: 8.8.8.8 и 8.8.4.4).

Все настройки (кроме первого пункта) могут быть получены по DHCP при подключении компьютера к сети (или загрузке ОС, если к сети уже подключен).

Основы Web

Протокол HTTP

Для работы с запросами и ответами в веб-браузерах используется простой текстовый протокол, позволяющий запросить у веб-сервера информацию. HTTP-протокол позволяет получать информацию из интернета, в том числе служебную: например, user agent, в роли которого, как правило, выступает сам браузер. Но не всегда: программы **curl** и **wget** тоже **user agent**, так что не каждый HTTP-клиент — браузер. Веб-сервер также сообщает информацию о себе, метаинформацию о документе, кодировку и тип содержимого — MIME-type. Это позволит клиенту корректно раскодировать информацию.

Существует несколько типов запросов, но наиболее часто применяются GET и POST.

GET служит для запроса страницы. Как правило, в качестве адреса ресурса в GET передается путь к файлу относительно веб-директории. Например, **/articles/article.html**.

Если запрашивается веб-приложение, ему могут быть переданы параметры **/articles.php?page=12**.

Возможно настроить веб-сервер так, чтобы веб-приложение было замаскировано, а параметры передавались через имена «папок» и «файлов». То есть **/articles/pages/12.html** может означать как файл **12.html**, хранящийся в директории **pages**, которая находится в директории **articles**, так и скрипт, у которого запрашивается статья с номером 12.

Такие запросы, благодаря параметрам или пути к ресурсу, могут передавать информацию на сервер. Но эти значения кешируются и сохраняются в истории браузера, поэтому для передачи информации на сервер используются запросы POST.

Как правило, с помощью POST отправляются новые файлы на веб-сервер (upload), записи на форумах и блогах, логин и пароль для входа в ту или иную систему. Запрос POST, как правило, передается скрипту на PHP (или другом языке), и тот его обрабатывает.

В ответ сервер может вернуть HTML-код запрошенного ресурса или двоичный файл (например, картинку). В ответе веб-сервер сообщает код, например:

- 200 Ok — все хорошо, ресурс найден и отправлен пользователю;
- 201 Created — новая страница на сайте создана (например, скриптом PHP, который получил запрос POST);
- 301 — ресурс переехал;
- 404 — страница не найдена (либо в файловой системе, либо скрипт не сумел найти соответствующую запись в БД);
- 403 — доступ запрещен (например, при HTTP-аутентификации пользователь не сумел ввести правильный пароль);
- 500 — ошибка сервера (ошибка в конфигурационном файле).

Немного об Apache2 и Nginx

Протокол HTTP, как правило, работает через TCP-сессию. Поэтому для приема нескольких одновременных соединений веб-сервер должен использовать процессы или потоки.

Чтобы определить, как осуществлять распараллеливание обработки, в Apache2 используется **mpm** — Apache Multi-Processing Module, то есть модуль мультипроцессовой обработки. Еще есть такие варианты:

- **mpm-prefork** — используются заранее созданные процессы, стандартный вариант работы;
- **mpm-itk** — основан на **prefork**, но позволяет также запускать процессы от имени разных пользователей. Используется хостерами.
- **mpm-worker** — использует заранее созданные потоки (**p_threads**). Логично, что такой способ будет нестабильным, учитывая специфику работы Apache2: PHP, запущенный в качестве модуля, будет выполнен внутри потока, а не процесса.

В этом принципиальная разница в работе Apache2 только со статическим содержимым (**mpm-worker**, используются потоки) или со скриптами (**mpm-prefork**, используются предварительно созданные процессы). Разница потоков и процессов заключается в том, что потоки работают с одними и теми же данными. Данные процессов изолированы от других процессов (даже дочерних). Это небезопасно при выполнении в потоках разного кода. Поэтому для исполнения **php** и других скриптов применяется **mpm-prefork**.

Nginx использует другой механизм: он создает один мастер-процесс и один или более процессов-воркеров (**worker**). Как правило, на одном CPU используется один воркер, но может применяться и больше, например при интенсивном вводе-выводе. Все входящие подключения распределяются по воркерам. Каждый воркер использует одну очередь и более низкоуровневые механизмы, чем потоки: например, **kqueue** во FreeBSD или **epoll** в Linux. Каждый воркер может обработать довольно значительное число соединений. Nginx самостоятельно реализует обработку одновременных соединений, используя низкоуровневые возможности операционной системы. Именно поэтому Nginx может работать только со статическим содержимым. Но таким образом достигаются высокое быстродействие и отказоустойчивость.

Чтобы использовать динамическое содержимое, генерируемое скриптами, Nginx может получать результаты выполнения скриптов от Apache2 или непосредственно PHP, работающего через модуль **php-fpm**. Так как в таком случае скрипты выполняются в другом процессе, работа Nginx безопасна. Nginx будет хорошо работать, если файлы, к которым обращаются HTTP-запросы, не блокируются

другими приложениями. В случае появления «медленного запроса» при использования механизма пула потоков такой запрос отдается отдельному потоку.

Разница между Apache2 и Nginx в том, что модули Apache2 выполняются в рамках одного приложения. Это работает быстрее, чем если бы модуль находился в отдельном приложении. Но так как потоки в данном случае использовать небезопасно (один скрипт может повредить данные другого, если они запускаются в потоках), Apache2 использует по умолчанию процессы. Каждый запрос выполняется в отдельном процессе, созданном заранее (такой механизм называется **prefork**).

Таким образом, Apache2 оказывается более медленным в выдаче статического содержимого, но удобным для работы с динамическим контентом. Именно поэтому часто используется связка Nginx + Apache2.

База данных MySQL

Чтобы реализовать полноценный веб-сервер LAMP, надо научиться устанавливать и настраивать стек технологий: Linux, Apache2 и PHP. Дополнительно потребуется база данных MySQL, MariaDB или Postgres.

Установим базу данных MySQL (проверяйте актуальные версии пакетов):

```
sudo apt-get install mysql-server mysql-client mysql-common php7.0-mysql
```

При установке будет предложено создать пароль для **root** (не для Ubuntu, а для самого MySQL).

Правим **/etc/mysql/my.cnf**:

```
bind-address=другой машины либо 0.0.0.0
```

Например, **bind-address=192.168.131.1**, если хотите с хостовой машины под Windows обращаться к MySQL. Либо оставьте так:

```
bind-address=127.0.0.1
```

Рекомендуется:

```
[mysqld]

skip-character-set-client-handshake

character-set-server=utf8

init-connect='SET NAMES utf8'

collation-server=utf8_general_ci

...

[client]
```

```
default-character-set=utf8

..

[mysqldump]

default-character-set=utf8

..
```

После установки желательно очистить MySQL от мусора: удалить тестовые учетные записи и базы данных, а также задать пароль для пользователя **root** и указать, разрешено ли ему заходить удаленно (по умолчанию — запрещено). Для этого нужно запустить команду:

```
mysql_secure_installation
```

Зайти пользователем **root**:

```
mysql -u root -p
```

Создаем базу **mydatabase**:

```
CREATE DATABASE mydatabase;
```

Создаём пользователя **user** с паролем 1234 и даем ему все права на эту базу. Не забываем обновить таблицу прав пользователей MySQL в памяти:

```
CREATE USER 'user'@'localhost' IDENTIFIED BY '1234';

GRANT ALL PRIVILEGES ON mydatabase.* TO user@localhost;

FLUSH PRIVILEGES;

Exit
```

Заходим пользователем **user** и вводим пароль:

```
mysql -u user -p

1234
```

Указываем, какую базу будем использовать:

```
USE mydatabase;

Создаем таблицу

create table mytable (

id int not null auto_increment,

txt varchar(100),

n int,

primary key (id));
```

Добавляем значения:

```
insert into mytable (txt,n) values('Wall Street',7);

insert into mytable (txt,n) values('5th Avenu',1);

...

select * from mytable;
```

Получаем на выход содержимое таблицы.

Можно подготовить команды в отдельном файле, например **cmds.sql**:

```
USE mydatabase;

insert into mytable (txt,n) values('Gorki park',7);

insert into mytable (txt,n) values('Kremlin',NULL);
```

Выполняем данные из файла:

```
mysql -u user -p mydatabase <cmds.sql
```

Можно зайти юзером и, сделав USE и SELECT, полюбоваться результатом.

Делаем бэкап:

```
mysqldump -u user -p mydatabase >backup.sql
```

С помощью скриптов и крона можно сделать это регулярным. Восстанавливаем — воспользуемся один раз **mysqladmin**, а не **mysql**.

Создаем рутом базу **newbase**:

```
mysqladmin -u root -p create newbase
```

Импортируем в **newbase** бэкап:

```
mysql -u root -p --default-character-set=utf8 newbase<backup.sql

mysql -u root -p

USE newbase;

select * from mytable;

exit
```

Если захочется работать в веб-интерфейсе, можно также установить **phpMyAdmin**.

Знакомство с Apache2

Познакомимся с Apache2. Он существовал раньше Nginx, и хотя сейчас последний его активно вытесняет, Apache2 стоит знать. Многие программы (Zabbix, Asterisk) и CMS (WordPress, Drupal и т. д.) заточены под Apache2. Чтобы настроить их работу на Nginx, нужно представлять его архитектуру.

Для установки Apache2 выполним команду:

```
$ sudo apt-get -y install apache2
```

Легко проверить, что Apache2 установился. Узнайте IP-адрес вашей машины и введите его в браузере. Вы должны увидеть стартовую страницу-заглушку веб-сервера Apache2.

Изучите структуру конфигурации в директории **/etc/apache2**.

Обратите внимание, что вы не найдете **httpd.conf**. Если вы где-то нашли информацию о данном конфигурационном файле, она уже устарела. Вместо него применяется набор из **apache2.conf** и конфигурационных файлов, расположенных в соответствующих директориях.

В Debian и Ubuntu используется удобный подход на основе символических ссылок. Директория ***-available** хранит доступные конфигурационные файлы, а ***-enabled** — символические ссылки на активные настройки в данный момент.

Конфигурационные файлы, conf-available/conf-enabled

Большинство настроек хранятся в директориях **conf-available** и **conf-enabled**. Папка **conf-available** содержит файлы ***.conf** — все доступные конфигурационные файлы. Папка **conf-enabled** — символические ссылки на конфиги из **conf-available**. Именно эти конфигурационные файлы активны, остальные «выключены». Утилиты **a2enconf** и **a2disconf** включают и выключают соответствующий конфиг. Фактически они создают и удаляют символическую ссылку через **ln -s** и предлагают послать серверу Apache2 сигнал **restart** или **reload**.

Модули Apache2, mods-available и mods-enabled

Директории **mods-available** и **mods-enabled** хранят модули. Если модуль не был установлен, его следует установить через **apt-get install**.

Активация и деактивация модулей выполняется аналогично конфигурационным файлам через утилиты **a2enmod/a2dismod**.

Виртуальные хосты, sites-available и sites-enabled

Конфигурационные файлы для виртуальных хостов содержатся в директориях **sites-available** и **sites-enabled**. Изначально файлы находятся в **sites-available** и они не активны. При активации сайта создается символическая ссылка в директории **sites-enabled** на соответствующий конфигурационный файл из директории **sites-available**. При деактивации символическая ссылка из директории **sites-enabled** удаляется, а сам конфигурационный файл в директории **sites-available** остается. Это очень удобно, так как можно подключать и отключать сайты.

Активация или деактивация хоста (если клиент не оплатил хостинг) идет через команды **a2ensite** и **a2dissite**.

Обратите внимание на два файла, присутствующих по умолчанию: **000-default.conf** и **default-ssl.conf**.

Это пример сайта — отвечает для всех доменных имен по IP-адресу вашего сервера. Первый — для протокола HTTP (порт 80), второй — для HTTPS (порт 443).

По умолчанию включен только HTTP.

Создаем простейший конфигурационный файл

Обратите внимание, что при попытке запуска веб-сервер выдает предупреждение:

```
Could not reliably determine the server's fully qualified domain name, using
127.0.1.1 for ServerName
```

Можно сделать соответствующую запись в файле **/etc/apache2.conf**, но мы пойдем другим путем: создадим **fqdn.conf** в директории **conf-available** и активируем конфиг.

```
$ echo "ServerName localhost" | sudo tee /etc/apache2/conf-available/fqdn.conf
$ sudo a2enconf fqdn
```

Работа с модулями

Пример 1. Модуль активен

Предположим, IP-адрес вашей виртуальной машины — 192.168.131.136. Тогда с помощью модуля **Apache2 status** мы можем посмотреть состояние сервера. Но если вы попытаетесь просмотреть в браузере http://192.168.131.136/server-status, увидите ошибку 403. Доступ запрещен.

Что это означает? В **mods-enable** увидим, что модуль **status** там присутствует — значит, он активен. Получается, дело в настройках прав доступа. Можем посмотреть данный ресурс локально, например в текстовом браузере **lynx**:

```
$ lynx  http://192.168.131.136/server-status
```

Или перейдем через Alt-F7 в X-Windows и запустим этот адрес в Firefox.

Работает. Но можно настроить так, чтобы из сети вашей хостовой машины тоже можно было посмотреть данный файл. Откройте файл **/etc/apache2/mods-available/**status.conf и после строки:

```
 Require local
```

… добавьте:

```
Require ip 192.168.131.0/24
```

Исправьте на адрес вашей сети.

Необходимо, чтобы сервер перечитал конфигурацию:

```
$ sudo systemctl reload apache2
```

Пример 2. Модуль установлен, но не активен

Кроме **status** есть модуль **info**.

Откройте **http://192.168.131.136/server-info**

Независимо от того, с хоста или гостевой машины вы смотрите, будет ошибка 404 (not found).

Если посмотрим в **mods-enabled**, то не найдем файл **info**, а в **mods-available** он присутствует — установлен, но не активен.

Поэтому выполняем:

```
$ sudo a2enmod info
$ sudo systemctl restart apache2
```

Проверяем.

Также можете поправить IP-адреса, чтобы **Info** можно было видеть с машины-хоста (аналогично проделанным операциям).

Пример 3. Модуль не установлен

Поработаем с **php7**. Установим его и некоторые дополнения: для работы с MySQL, Curl — http-клиент для работы с удаленными файлами, **json** — часто применяемый транспорт между бэкендом и фронтендом):

```
$sudo apt-get -y install php7.0 libapache2-mod-php7.0 php7.0-mysql php7.0-curl
```

```
php7.0-json
```

Перезапускаем веб-сервер:

```
$ sudo systemctl reload apache2
```

Проверяем:

```
$ php -v
```

Создадим **phpinfo**:

```
# cat >/var/www/html/info.php
<?php
phpinfo();
?>
Ctrl-D
```

Проверяем: http://192.168.131.136/info.php.

Чтобы каждый раз не писать IP-адрес, можно в файле (если ваша машина-хост — Windows) **c:\windows\system32\drivers\etc\hosts** прописать его с помощью редактора FAR или Notepad++.

```
192.168.131.136 test.tst
```

Виртуальные хосты (много сайтов на одном сервере)

Если пропишем в файле **c:\windows\system32\drivers\etc\hosts** (если ваша машина-хост — Windows) или в **\etc\hosts** (если Linux или Mac OS):

```
192.168.131.136 test.tst
192.168.131.136 test.a
192.168.131.136 test.b
```

…все равно каждый раз будет выдаваться одна и та же страница-заглушка.

Как поддерживать несколько сайтов?

- **Использовать разделение по портам. test.tst** ведет на один сайт, а **test.tst:8080** — на другой. Так иногда осуществляют доступ к админке, но чаще всего так поступают, если на сервере работает несколько приложений-серверов HTTP. Например, на **test.tst** на 80 порте по умолчанию работает Apache2, а на **test.tst:8080** — сервер онлайн-радиостанции **icecast2**. В последнее время так поступать не принято, так как с помощью **Nginx** можно пробрасывать трафик (например, **test.tst/audio/** будет передавать трафик **icescat2**, слушающему трафик на **127.0.0.1:8080**, а остальные пути — на **127.0.0.1:80**, который слушает **Apache2**).

- **Использовать несколько IP-адресов.** В этом случае многосайтовость будет осуществляться благодаря механизму **DNS**. Веб-сервер будет определять сайт по IP-адресу.
- **Использовать механизм виртуальных хостов.** В этом случае в протоколе **http** передается заголовок **host**, который указывает доменное имя. В этом случае сайты будут различаться по хосту.

Многосайтовость по порту

Скопируйте **0000-default.conf** в **8080.conf**. Отредактируйте. Исправьте, чтобы получилось так:

```
<VirtualHost *:8080>
```

```
DocumentRoot /var/www/8080
```

Создайте соответствующую директорию и положите туда файл, указав в **title** и **h1**, что это именно сайт с порта 8080.

В **ports.conf** добавьте:

```
Listen 8080
```

Активируйте сайт и перезагрузите сервер:

```
# a2ensite 8080
# systemctl reload apache2
```

Проверьте. Деактивируйте сайт.

Многосайтовость по IP-адресу

Поднимите новый IP-адрес на вашей машине:

```
# ifconfig ens33:second 172.16.0.1 netmask 255.255.0.0 up
```

В Windows (**Win+X > Командная строка > Административная**) пропишите маршрут на этот адрес через IP-адрес вашей виртуальной машины:

```
c:\>route add 172.16.0.1 mask 255.255.0.0 192.168.131.128
```

192.168.131.128 — адрес гостевой **Ubuntu**.

Добавьте адрес **172.16.0.1** в **hosts** в Windows (исправьте):

```
172.16.0.1 test.b
```

Скопируйте **000-default.conf** в **test.b**. Отредактируйте **test.b**:

```
<VirtualHost 172.16.0.1:80>
```

Не забудьте создать директорию, указать ее в **test.b** и создать индекс. Активируйте сайт, попросите веб-сервер перечитать конфигурацию, проверьте. Сравните в браузере сайты **test.a** и **test.b**.

Теперь деактивируйте сайт, удалите конфигурационный файл **test.b** и отключите сетевой интерфейс **ens33:second**.

Не забудьте удалить маршрут в Windows:

```
c:\>route delete 172.16.0.1 mask 255.255.0.0 192.168.131.128
```

Многосайтовость по домену

Скопируйте **000-default.conf** в **test.a**. Раскомментируйте в файле **ServerName** и впишите туда **test.a**:

```
ServerName test.a
```

Не забудьте создать директорию, указать ее в **test.a** и создать индекс. Активируйте сайт, попросите веб-сервер перечитать конфигурацию, проверьте.

Сравните. Удалите сделанные изменения или настройте сайты.

Устанавливаем Nginx

```
apt install nginx
```

Если хочется совсем новую версию, нужно воспользоваться репозиторием производителя, но надо понимать, что конфигурация будет не в debian-стиле. Поэтому лучше сохранить конфигурационные файлы: они удачно настроены для работы с Ubuntu — возможно, придется в них подглядеть.

На официальном сайте Nginx в wiki-сообществе на странице https://www.nginx.com/resources/wiki/start/topics/tutorials/install/ подробно указано, что делать (для Ubuntu 16.0).

Сначала надо создать файл **/etc/apt/sources.list.d/nginx.list** и добавить туда строки:

```
deb http://nginx.org/packages/ubuntu/ xenial nginx
deb-src http://nginx.org/packages/ubuntu/ xenial nginx
```

После этого:

```
sudo apt-get update
```

Мы получим ошибку вида:

```
Reading package lists... Done
W: GPG error: http://nginx.org/packages/ubuntu xenial InRelease: The following signatures couldn't be verified because the public key is not available: NO_PUBKEY ABF5BD827BD9BF62
W: The repository 'http://nginx.org/packages/ubuntu xenial InRelease' is not signed.
N: Data from such a repository can't be authenticated and is therefore potentially dangerous to use.
N: See apt-secure(8) manpage for repository creation and user configuration details.
```

Вот эта фраза после NO_PUBKEY нам понадобится. Ее нужно вставить после **sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys**:

```
sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys   ABF5BD827BD9BF62
```

Теперь:

```
sudo apt-get update
sudo apt-get install nginx
```

Но этого, конечно, мало. Заглянем в директорию **/etc/nginx**.

Здесь есть файл **nginx.conf** (в котором есть подключение всех файлов в **conf.d**), и тут же будут подключения. А могут и не быть, но можно сделать на **sites-enabled**.

В директории **conf.d** находится файл **default.conf**, который и обеспечивает работу нужной страницы.

Если мы хотим сделать новый файл, легко его скопировать и настроить.

Какие-то вещи здесь похожи на Apache2, а другие сильно отличаются.

Для начала давайте разберем пример **default.conf** по умолчанию.

```
server {
    listen       80;
    server_name  localhost;
    #charset koi8-r;
    #access_log  /var/log/nginx/log/host.access.log  main;
    location / {
        root   /usr/share/nginx/html;
        index  index.html index.htm;
    }
    #error_page  404              /404.html;
    # redirect server error pages to the static page /50x.html
    #
    error_page   500 502 503 504  /50x.html;
    location = /50x.html {
        root   /usr/share/nginx/html;
    }
    # proxy the PHP scripts to Apache listening on 127.0.0.1:80
    #
    #location ~ \.php$ {
    #    proxy_pass   http://127.0.0.1;
    #}
    # pass the PHP scripts to FastCGI server listening on 127.0.0.1:9000
    #
    #location ~ \.php$ {
    #    root           html;
    #    fastcgi_pass   127.0.0.1:9000;
    #    fastcgi_index  index.php;
    #    fastcgi_param  SCRIPT_FILENAME  /scripts$fastcgi_script_name;
    #    include        fastcgi_params;
    #}
    # deny access to .htaccess files, if Apache's document root
    # concurs with nginx's one
    #
    #location ~ /\.ht {
    #    deny  all;
    #}
}
```

Бросаются в глаза **listen** и **servername**.

Есть и **root**, но добавляется незнакомая сущность **location**. После работы с **.htaccess** довольно сложно привыкнуть к работе с Nginx, ведь в нем нет отдельных правил для директорий в том виде, в каком они были в Apache. Apache проверял текущую директорию на этот скрытый файл, затем родительскую и так далее. Поэтому все фокусы с **rewrite** здесь выглядят по-другому.

Как вы заметили, **location** может быть несколько. В **location** допустимы регулярные выражения, если они относятся к данному серверу. Но если мы перенаправляем трафик на другой сервер Apache2,

придется обойтись без **regexp**. Разница в том, что Apache2 смотрит в директории, а Nginx анализирует запросы. **location** — набор конфигураций, который подпадает под тот или иной запрос.

Пока мы видим немного — все запросы, которые идут от /, нужно рассматривать как запрос статических файлов из директории **/usr/share/nginx/html**.

Также закомментированы несколько примеров.

Это отправка запросов, содержащих **.php**, веб-серверу Apache (слушающему 80 порт, но по адресу 127.0.0.1), либо через **Fast-CGI** на **php**, слушающем 127.0.0.1 по TCP-порту 9000.

В любом случае,**.htaccess**-файлы не должны выдаваться запрашивающим, поэтому также нужно раскомментировать запрет на .ht-файлы. Кстати, Apache или **php** через **cgi** могут обрабатывать ту же самую директорию.

Конфигурация nginx

Как мы можем улучшить конфигурационные файлы для сайта?

1. Первым делом, например, перенесем **root** из **location** в **server**:

```
server {
    listen       80;
    server_name  localhost;
    #сюда
    root   /usr/share/nginx/html;
    #charset koi8-r;
    #access_log  /var/log/nginx/log/host.access.log  main;
    location / {
    # было здесь
    #    root   /usr/share/nginx/html;
        index  index.html index.htm;
```

Можно перезапустить сервер и убедиться, что все работает.

2. Дальше создадим директории для доступных и активных сайтов, переместим **default.conf** в доступные сайты и скопируем в **servername.conf**, а также создадим на него символическую ссылку.

```
cd /etc/nginx/conf.d
mv default.conf ../sites-available/
cd ../sites-available/
cp default.conf servername.conf
cd ../sites-available
ln -s /etc/nginx/sites-available/servername.conf
../sites-enabled/servername.conf
```

3. Поправим и **servername.conf**. Обрамим все в **http** и добавим два описания для **server**: одно с **root**, другое с редиректом. Вот что у нас получилось:

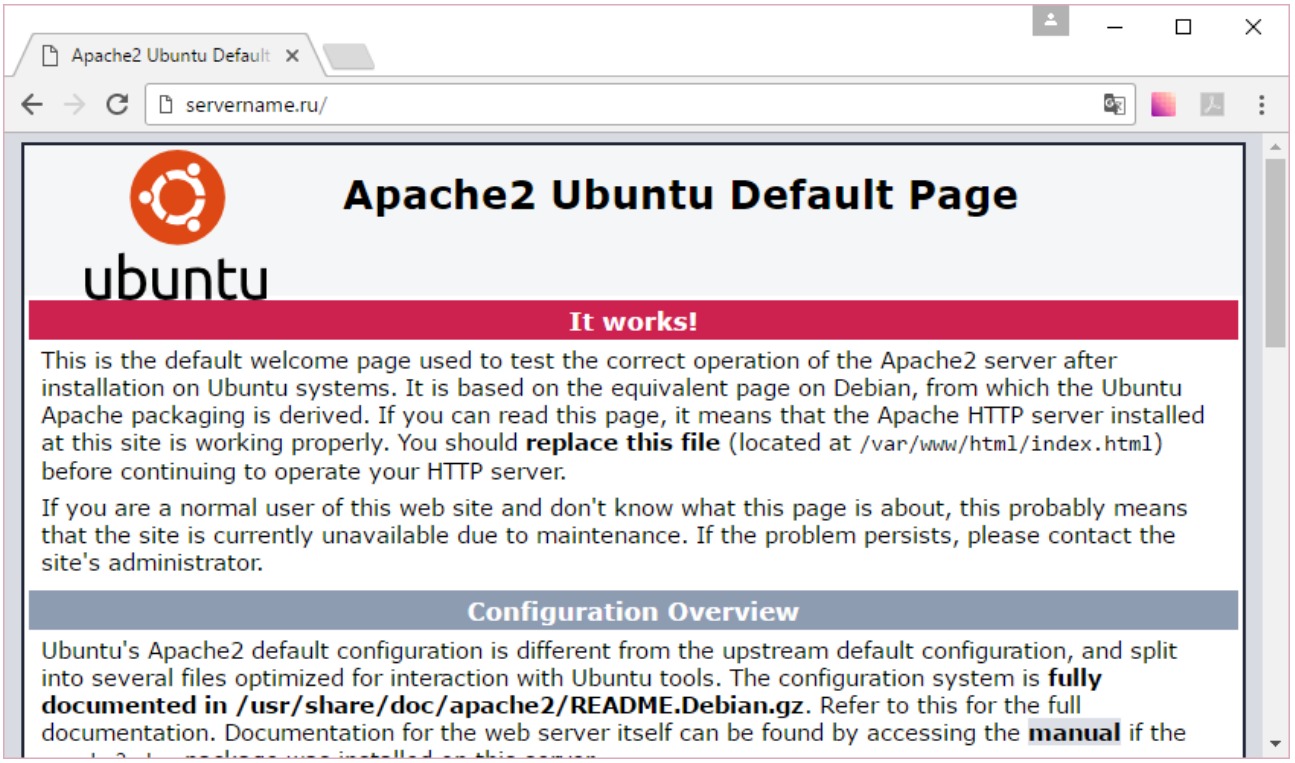
```
#находится в блоке http в nginx.conf
index  index.html index.htm;
```

```
server {
    server_name www.servername.ru;
    return 301 $scheme://servername.ru$request_uri;
}
server {
    server_name  servername.ru;
    root   /usr/share/nginx/servername;
...
}
```

4. Теперь заставим сервер все запросы передавать на другой сервер:

```
#находится в блоке http в nginx.conf
index  index.html index.htm;
server {
    server_name www.servername.ru;
    return 301 $scheme://servername.ru$request_uri;
}
server {
    server_name  servername.ru;
    root   /usr/share/nginx/servername;
    #charset koi8-r;
    #access_log  /var/log/nginx/log/host.access.log  main;
    location / {
    proxy_pass http://192.168.131.27;
    }
    #error_page  404              /404.html;
    # redirect server error pages to the static page /50x.html
    #
    error_page   500 502 503 504  /50x.html;
    location = /50x.html {
        root   /usr/share/nginx/html;
    }

}
```



5. Можно сделать несколько **location**: один обрабатывает /, другой — /wiki, третий — /shop. И направить все на разные сервера.

Но еще интереснее балансировать нагрузку. В Nginx есть реализация **Round Robin**, когда запросы по очереди будут раскидываться по указанным серверам. Для этого в Nginx имеется модуль **upstream**:

```
index  index.html index.htm;
upstream backend {
    server 1.2.3.4       weight=5;
    server 5.6.7.8:8080;

    server 192.168.131.60   backup;
    server server3.example.ru   backup;
}
server {
    server_name www.servername.ru;
    return 301 $scheme://servername.ru$request_uri;
}
server {
    server_name  servername.ru;
    root   /usr/share/nginx/servername;
    #charset koi8-r;
    #access_log  /var/log/nginx/log/host.access.log  main;
    location / {
    proxy_pass http://backend;
    }
    #error_page  404              /404.html;
    # redirect server error pages to the static page /50x.html
    #
    error_page   500 502 503 504  /50x.html;
    location = /50x.html {
```

```
        root   /usr/share/nginx/html;
    }
}
```

Обратите внимание, что, если первые два сервера не ответят, а третий доступен, Nginx вернет данные от Apache2, хотя и с большой задержкой.

Настроим работу с PHP

Устанавливаем модуль **php-fpm** (проверяйте актуальность версий пакетов):

```
apt install php7.0-fpm
systemctl status php7.0-fpm
```

Создадим **/usr/share/nginx/servername/phpinfo.php**:

```
<?php
    phpinfo();
?>
```

Настраиваем наш сайт

```
index  index.html index.htm;
server {
    server_name www.servername.ru;
    return 301 $scheme://servername.ru$request_uri;
}
server {
    server_name  servername.ru;
    root   /usr/share/nginx/servername;
    #charset koi8-r;
    #access_log  /var/log/nginx/log/host.access.log  main;
    location / {
    }
    #error_page  404              /404.html;
    # redirect server error pages to the static page /50x.html
    #
    error_page   500 502 503 504  /50x.html;
    location = /50x.html {
        root   /usr/share/nginx/html;
    }
    # pass the PHP scripts to FastCGI server listening on 127.0.0.1:9000
    #
    location ~ \.php$ {
        fastcgi_index  index.php;
        fastcgi_param  SCRIPT_FILENAME  $document_root$fastcgi_script_name;
        include        fastcgi_params;
        fastcgi_pass unix:/run/php/php7.0-fpm.sock;
    }
```

```
    # deny access to .htaccess files, if Apache's document root
    # concurs with nginx's one
    #
    location ~ /\.ht {
        deny  all;
    }
}
```

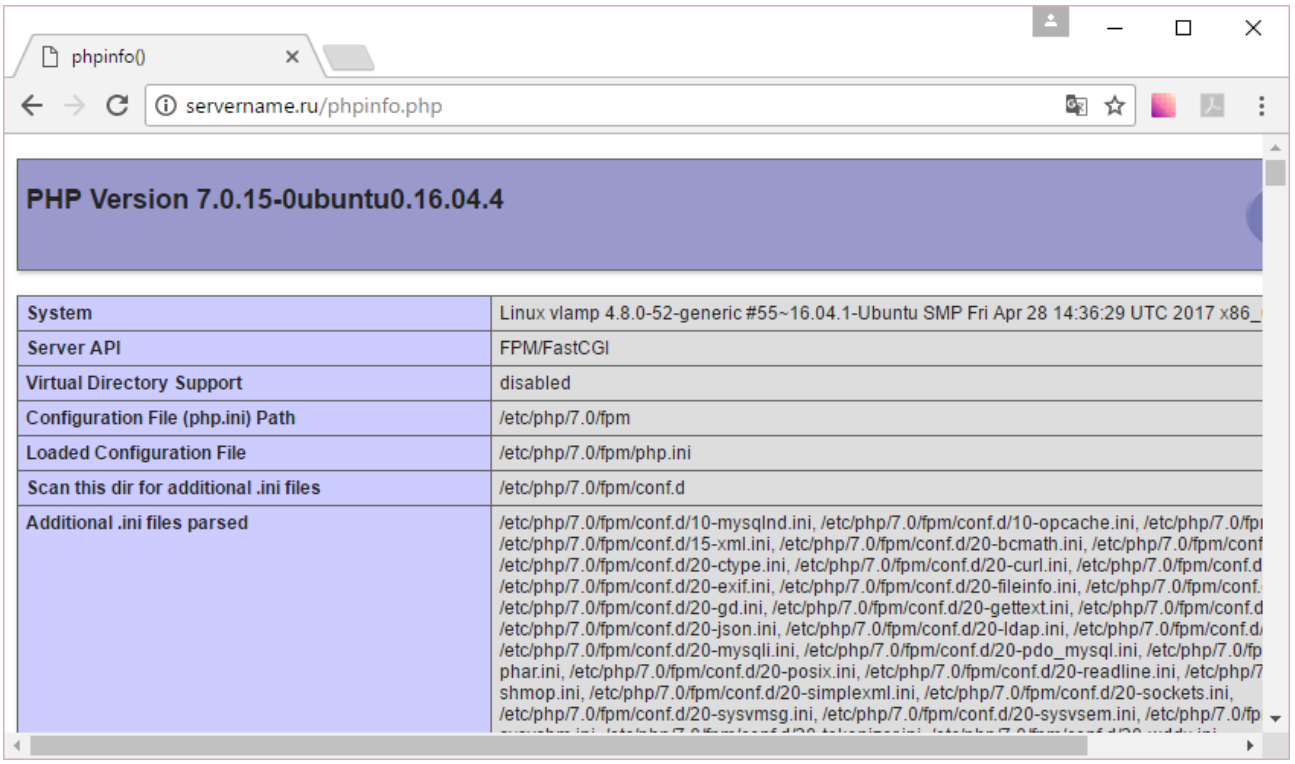
Проверяем. Если ошибка, у нас есть логи **/var/log/php7.0-fpm.log**, **/var/log/nginx/error.log**.

Если ошибка возникнет — скорее всего, из-за пользователя и группы, которым принадлежит файл **fastcgi_pass unix:/run/php/php7.0-fpm.sock**. Настройте так, чтобы и Nginx и **php7.0-fpm** могли к нему обратиться. Сделать это можно, добавив пользователя Nginx, от имени которого работает веб-сервер, в группу **www-data**, от имени которой работает **php-fpm**.

```
# usermod -a -G www-data nginx
```

Более того, все файлы в директории веб-сервера тоже лучше всего сделать принадлежащими пользователю **www-data**.

Если все сделали правильно:



Практическое задание

1. Установить Apache2. Прислать скриншоты работающего сервера.
2. Установить MySQL. Проверить работу, через консольного клиента, проверить команды **select user from mysql.users;** и **select * from users;**

3. Установить **php7.2** и **phpmyadmin**.

4. * Зайти пользователем **root** и попробовать там создать новую тестовую БД и пользователя для работы с ней. Создать в ней пару таблиц и заполнить их произвольным содержимым. Потом зайти в консольного клиента MySQL новым пользователем и вывести содержимое каждой из таблиц в новой базе данных в консоли, используя команды.

5. * Установить Nginx и настроить его на работу с **php-fpm**.

6. * Настроить Nginx в качестве балансировщика. Используя **mod_upstream**, раскидывать весь входящий трафик по трем Apache2-серверам, находящимся в локальной сети.

Примечание. *Задания 4–6 даны для тех, кому упражнений 1–3 показалось недостаточно.*

Дополнительные материалы

1. https://www.nginx.com/resources/wiki/start/topics/tutorials/install/

2. https://www.nginx.com/resources/wiki/start/topics/tutorials/config_pitfalls/

3. http://nginx.org/ru/docs/beginners_guide.html

4. https://ruhighload.com/%D0%91%D0%B0%D0%BB%D0%B0%D0%BD%D1%81%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B0+%D0%B1%D1%8D%D0%BA%D0%B5%D0%BD%D0%B4%D0%BE%D0%B2+%D1%81+%D0%BF%D0%BE%D0%BC%D0%BE%D1%89%D1%8C%D1%8E+nginx