Практическая работа №1

Выполнение: ознакомиться с теорией, параллельно выполнять задания по тексту (для этого текущие настройки nginx сохранить в бэкап). Далее сделать задание по настройке двухуровневой архитектуры.

Nginx — это легковесный веб-сервер (на самом деле, не только веб-), созданный Игорем Сысоевым. Nginx используется много где, в том числе в Mail.ru, Яндексе, Рамблере, и великом-великом множестве других высоконагруженных и не очень веб-проектов.

Устанавливается Nginx очень просто:

```
Yum install nginx
service nginx start
```

Логи запросов и ошибок пишутся в /var/log/nginx/

Как работают nginx и его модули, определяется в конфигурационном файле. По умолчанию, конфигурационный файл называется nginx.conf и расположен в каталоге /usr/local/nginx/conf, /etc/nginx или /usr/local/etc/nginx.

Чтобы запустить nginx, нужно выполнить исполняемый файл. Когда nginx запущен, им можно управлять, вызывая исполняемый файл с параметром -s. Используйте следующий синтаксис:

```
nginx -s сигнал
```

Где *сигнал* может быть одним из нижеследующих:

– stop — быстрое завершение

– quit — плавное завершение

– reload — перезагрузка конфигурационного файла

– reopen — открыть заново лог-файлы

Например, чтобы остановить процессы nginx с ожиданием окончания обслуживания текущих запросов рабочими процессами, можно выполнить следующую команду:

```
nginx -s quit
```

Команда должна быть выполнена под тем же пользователем, под которым был запущен nginx.

Изменения, сделанные в конфигурационном файле, не будут применены, пока команда перезагрузить конфигурацию не будет вручную отправлена nginx или он не будет перезапущен. Для перезагрузки конфигурации выполните:

```
nginx -s reload
```

Получив сигнал, главный процесс проверяет правильность синтаксиса нового конфигурационного файла и пытается применить конфигурацию, содержащуюся в нём. Если это ему удаётся, главный процесс запускает новые рабочие процессы и отправляет сообщения старым рабочим процессам с требованием завершиться. В противном случае, главный процесс откатывает изменения и продолжает работать со старой конфигурацией. Старые рабочие процессы, получив команду завершиться, прекращают принимать новые запросы и продолжают обслуживать текущие запросы до тех пор, пока все такие запросы не будут обслужены. После этого старые рабочие процессы завершаются.

Посылать сигналы процессам nginx можно также средствами Unix, такими как утилита kill. В этом случае сигнал отправляется напрямую процессу с данным ID. ID главного процесса nginx записывается по умолчанию в файл nginx.pid в каталоге /usr/local/nginx/logs или /var/run. Например, если ID главного процесса равен 1628, для отправки сигнала QUIT, который приведёт к плавному завершению nginx, нужно выполнить:

```
kill -s QUIT 1628
```

Для просмотра списка всех запущенных процессов nginx может быть использована утилита ps, например, следующим образом:

```
ps -ax | grep nginx
```

Структура конфигурационного файла

Nginx состоит из модулей, которые настраиваются директивами, указанными в конфигурационном файле. Директивы делятся на простые и блочные. Простая директива состоит из имени и параметров, разделённых

пробелами, и оканчивается точкой с запятой (;). Блочная директива устроена так же, как и простая директива, но вместо точки с запятой после имени и параметров следует набор дополнительных инструкций, помещённых внутри фигурных скобок ({и}). Если у блочной директивы внутри фигурных скобок можно задавать другие директивы, то она называется контекстом (примеры: events, http, server и location).

Директивы, помещённые в конфигурационном файле вне любого контекста, считаются находящимися в контексте main. Директивы events и http располагаются в контексте main, server — в http, а location — в server.

Часть строки после символа # считается комментарием.

Раздача статического содержимого

Одна из важных задач конфигурации nginx — раздача файлов, таких как изображения или статические HTML-страницы. Рассмотрим пример, в котором в зависимости от запроса файлы будут раздаваться из разных локальных каталогов: /data/www, который содержит HTML-файлы, и /data/images, содержащий файлы с изображениями. Для этого потребуется отредактировать конфигурационный файл и настроить блок server внутри блока http с двумя блоками location.

Во-первых, создайте каталог /data/www и положите в него файл index.html с любым текстовым содержанием, а также создайте каталог /data/images и положите в него несколько файлов с изображениями.

Далее, откройте конфигурационный файл. Конфигурационный файл по умолчанию уже включает в себя несколько примеров блока server, большей частью закомментированных. Для нашей текущей задачи лучше закомментировать все такие блоки и добавить новый блок server:

```
http {
  server {
  }
}
```

В общем случае конфигурационный файл может содержать несколько блоков server, различаемых по портам, на которых они слушают, и по имени сервера. Определив, какой server будет обрабатывать запрос, nginx сравнивает URI, указанный в заголовке запроса, с параметрами директив location, определённых внутри блока server.

В блок server добавьте блок location следующего вида:

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

Этот блок location задаёт “/” в качестве префикса, который сравнивается с URI из запроса. Для подходящих запросов добавлением URI к пути, указанному в директиве root, то есть, в данном случае, к /data/www, получается путь к запрашиваемому файлу в локальной файловой системе. Если есть совпадение с несколькими блоками location, nginx выбирает блок с самым длинным префиксом. В блоке location выше указан самый короткий префикс, длины один, и поэтому этот блок будет использован, только если не будет совпадения ни с одним из остальных блоков location.

Далее, добавьте второй блок location:

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

Он будет давать совпадение с запросами, начинающимися с /images/ (location / для них тоже подходит, но указанный там префикс короче).

Итоговая конфигурация блока server должна выглядеть следующим образом:

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

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

Это уже работающая конфигурация сервера, слушающего на стандартном порту 80 и доступного на локальном компьютере по адресу http://localhost/. В ответ на запросы, URI которых начинаются с/images/, сервер будет отправлять файлы из каталога /data/images. Например, на запрочhttp://localhost/images/example.png nginx отправит в ответ файл /data/images/example.png. Если же этот файл не существует, nginx отправит ответ, указывающий на ошибку 404. Запросы, URI которых не начинаются на /images/, будут отображены на каталог /data/www. Например, в результате запроса http://localhost/some/example.html в ответ будет отправлен файл /data/www/some/example.html.

Чтобы применить новую конфигурацию, запустите nginx, если он ещё не запущен, или отправьте сигнал reload главному процессу nginx, выполнив:

```
nginx -s reload
```

В случае если что-то работает не как ожидалось, можно попытаться выяснить причину с помощью файлов access.log и error.log из каталога /usr/local/nginx/logs или /var/log/nginx.

Настройка простого сервера

Мы настроим базовый сервер, который будет обслуживать запросы изображений из локального каталога и отправлять все остальные запросы на сервер. В этом примере оба сервера будут работать в рамках одного экземпляра nginx.

Во-первых, создайте сервер, добавив ещё один блок server в конфигурационный файл nginx со следующим содержимым:

```
server {
    listen 80;
    root /data/up1;
```

```
    location / {
    }
}
```

Это будет простой сервер, слушающий на порту 80 (стандартный порт 80) и отображающий все запросы на каталог /data/up1 в локальной файловой системе. Создайте этот каталог и положите в него файл index.html. Обратите внимание, что директива root помещена в контекст server. Такая директива root будет использована, когда директива location, выбранная для выполнения запроса, не содержит собственной директивы root.

Далее, используйте конфигурацию сервера из предыдущего раздела и видоизмените её, превратив в конфигурацию сервера для статики.

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

Мы изменим блок location, который на данный момент отображает запросы с префиксом/images/ на файлы из каталога /data/images так, чтобы он подходил для запросов изображений с типичными расширениями файлов. Изменённый блок location выглядит следующим образом:

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

Параметром является регулярное выражение, дающее совпадение со всеми URI, оканчивающимися на.gif, .jpg или .png. Регулярному выражению должен предшествовать символ ~. Соответствующие запросы будут отображены на каталог /data/images.

Когда nginx выбирает блок location, который будет обслуживать запрос, то вначале он проверяет директивы location, задающие префиксы, запоминая location с самым длинным подходящим префиксом, а затем проверяет

регулярные выражения. Если есть совпадение с регулярным выражением, nginx выбирает соответствующий location, в противном случае берётся запомненный ранее location.

Итоговая конфигурация сервера выглядит следующим образом:

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

Этот сервер будет фильтровать запросы, оканчивающиеся на .gif, .jpg или .png, и отображать их на каталог /data/images (добавлением URI к параметру директивы root) и перенаправлять все остальные запросы на сервер, сконфигурированный выше.

Чтобы применить новую конфигурацию, отправьте сигнал reload nginx, как описывалось в предыдущих разделах.

Существует множество других директив для дальнейшей настройки соединения.

Задание на настройку двухуровневой архитектуры:

1. Зарегистрировать на себя домен третьего уровня (бесплатный)
2. Совершить необходимые настройки домена, чтобы открывались данные с вашего сервера
3. Установить nginx
4. Выявить от какого пользователя работает nginx
5. Выявить где сохраняются логи nginx
6. Выбрать и установить оптимальное значение worker_connections
7. Сконфигурировать nginx со следующими настройками (прокомментировать каждую строчку конфигурационного файла):
   a. Выявить от какого пользователя работает nginx
   b. nginx слушал 80 порт по default серверу
   c. Имя сервера соответствовало вашему домену / или IP

d. server_name_in_redirect установлено в off;

e. Задать корректные настройки proxy_set_header так, чтобы сервер открывал запросы с основного хоста по 80 порту, информацию можно посмотреть тут: (http://nginx.org/ru/docs/http/ngx_http_proxy_module.html)

f. Задать типы/список индексных файлов

g. Установить ограничения по максимальному размеру принимаемых файлов

h. Установить оптимальный таймаут соединения

8. Настроить nginx для отдачи всего статичного контента

9. В результате работы необходимо получить сервер, который принимает все запросы через nginx, аи кеширует и отдает статику из отдельной папки