ТЕХНОТРЕК

Занятие №5

Системное администрирование Linux

Сергей Клочков

Типовая архитектура сервиса

- Фронтенд отвечает за прием запросов, поступающих от клиентов, их первичную обработку, пересылку запросов на сервер приложений, получение ответов и отправку клиенту.
- Сервер приложений содержит в себе логику обработки запросов. Он взаимодействует с БД для хранения данных сервиса.
- БД отвечает за хранение данных и эффективную работу с ними

Фронтенд

- В качестве фронтенда в данном примере будет использоваться асинхронный HTTP-сервер nginx.
- Благодаря особенностям архитектуры, nginx обеспечивает высокую производительность и низкое потребление ресурсов на каждый коннект.
- В данном примере он будет использоваться как HTTP-сервер и fastcgi-клиент.

Протокол HTTP

- Используется в первую очередь для передачи веб-страниц и их содержимого (изображений, таблиц стилей, файлов javascript и т. п).
- Поскольку штатные средства работы с http (серверы, клиенты, библиотеки для различных ЯП) очень распространены, хорошо отлажены и многие разработчики и сисадмины умеют с ними работать, HTTP часто используется для взаимодействия приложений (т. н. REST API).

URL

- Идентификатором ресурса, получаемого по HTTP или используемого для отправки данных, является URL

Виды HTTP-запросов: GET

- Предназначен для получения содержимого страниц, а не отправки данных

```
> GET / HTTP/1.1
> User-Agent: curl/7.35.0
> Host: www.rambler.ru
> Accept: */*
>
< HTTP/1.1 200 OK
< Server: nginx/1.11.1
< Date: Wed, 19 Oct 2016 10:33:31 GMT
< Content-Type: text/html; charset=utf-8
< Transfer-Encoding: chunked
< Connection: keep-alive
< Keep-Alive: timeout=50
< X-App-Version: 3.22.56
< X-Request-Id: 52036B1BD2FE150C
< X-Sentry-ID: None
< Set-Cookie: proselytize=1; domain=.rambler.ru; path=/; expires=Wed, 20-Jul-17 23:55:55
GMT
< Set-Cookie: ruid=vAsAAPpLB1gba1WaAeEAAAB=; expires=Thu, 31-Dec-37 23:55:55 GMT;
domain=.rambler.ru; path=/
< P3P: CP="NON DSP NID ADMa DEVa TAIa PSAa PSDa OUR IND UNI COM NAV"
```

Виды HTTP-запросов: POST

- POST-запросы предназначены для отправки данных на сервер. Данные передаются в теле запроса.

```
> POST /post HTTP/1.1
> User-Agent: curl/7.35.0
> Host: httpbin.org
> Accept: */*
> Content-Length: 15
> Content-Type: application/x-www-form-urlencoded
>
> foo=bar&fuz=baz
< HTTP/1.1 200 OK
< Server: nginx
< Date: Wed, 19 Oct 2016 10:36:00 GMT
< Content-Type: application/json
< Content-Length: 376
< Connection: keep-alive
< Access-Control-Allow-Origin: *
< Access-Control-Allow-Credentials: true
```

Коды ответа HTTP (успешные запросы)

- 1xx: служебные сообщения.
- 2xx: запрос обработан успешно.
- 3xx: редиректы. 301 – постоянный редирект, кешируемый браузерами навсегда. 302 – временный редирект. Цель редиректа указывается в заголовке Location. 304 – нет изменений (работает в сочетании с заголовком клиента If-Modified-Since)

```
GET /static/images/project-logos/enwiki.png HTTP/1.1
Host: en.wikipedia.org
If-Modified-Since: Mon, 14 Mar 2016 18:08:11 GMT

HTTP/2.0 304 Not Modified
Date: Wed, 19 Oct 2016 10:51:10 GMT
Last-Modified: Mon, 14 Mar 2016 18:08:11 GMT
```

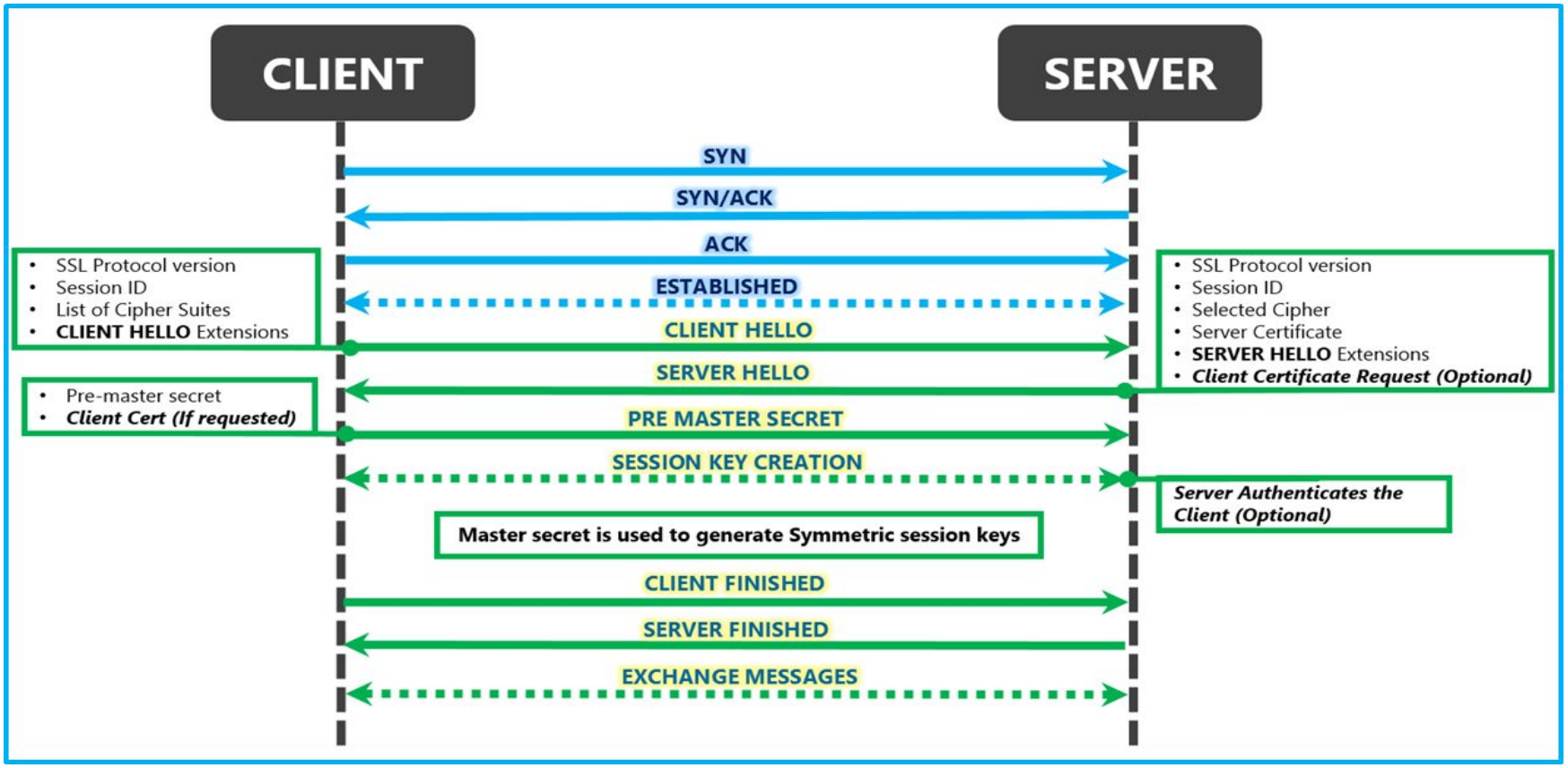
Коды ответа HTTP (ошибки)

- 4xx – ошибки клиента. 400 – некорректный запрос, 401 – требуется авторизация, 403 – доступ запрещен, 404 – страница не найдена.
- 5xx – ошибки сервера. 500 – внутренняя ошибка сервера, перезапрашивать бесполезно; 502 – апстрим (напр. сервер приложений) недоступен; 503 – временная ошибка сервера (можно перезапросить), 504 – истекло время ожидания ответа от апстрима.

SSL

- Предназначен для установки шифрованных туннелей

```
openssl s_client -connect google.com:443
```



Практическая работа

- Сегодня в качестве практической работы мы развернем на учебных виртуалках движок mediawiki с nginx в качестве фронтенда, mariadb (mysql) в качестве СУБД и php-fpm в качестве среды исполнения php
- Для начала подключим репозиторий nginx

```
[nginx]
name=nginx repo
baseurl=http://nginx.org/packages/centos/$releasever/$basearch/
enabled=1
gpgcheck=0
```

Nginx и статика

- Для установки nginx нужно выполнить yum install nginx
- В рамках предыдущего домашнего задания вы должны были подготовить rpm-пакет. Сейчас мы сделаем репозиторий, доступный по HTTP, и положим в него этот пакет

```
# Подготовка репозитория

yum install createrepo
mkdir -p /var/www/repo
cp <ваш пакет> /var/www/repo/
createrepo /var/www/repo/
```

Конфигурация nginx

- Откроем конфигурационный файл /etc/nginx/nginx.conf и разберем его содержимое.
- Чтобы настроить репозиторий, откроем /etc/nginx/conf.d/default.conf и внесем внутрь секции server следующее:

```
location /repo/ {
     root /var/www/;
     autoindex on;
}
```

- После этого запустим nginx и проверим доступность репозитория.
- Добавим конфигурацию репозитория в систему.

СУБД mysql (mariadb)

- MySQL – наиболее распросраненная из реляционных СУБД.
- В связи с разногласяими владельцев ТМ mysql и основной команды разработки был основан форк – mariadb.
- Установим СУБД с помощью команды yum install mariadb-server.
- Аккаунт суперпользователя БД – root. По умолчанию разрешена авторизация под root с локального хоста без пароля. Для простоты не будем это менять.

Создание БД и пользователя

- Создание базы данных и пользователя, имеющего полные права доступа к данной БД
- В MySQL права доступа выдяются на комбинацию имени пользователя и хоста (хостов), с которых он устанавливает соединение.

```
mysql -u root
> CREATE DATABASE `wiki` DEFAULT CHARACTER SET = 'utf8mb4' DEFAULT COLLATE =
'utf8mb4_general_ci';
> GRANT ALL ON `wiki`.* TO 'wiki'@'localhost' IDENTIFIED BY 'wiki';
> ^D
mysql -u wiki -pwiki wiki
>
```

Подготовка зависимостей mediawiki

- Для целей демонстрации развертывание движка будет выполнено из тарбола с сайта, поэтому установим зависимости.

```
yum install php php-mbstring php-cli php-embedded php-intl php-bcmath php-xmlrpc
php-dba php-xml php-common php-pdo php-mysql php-fpm php-gd
```

Выполним предварительную конфигурацию php: найдем в /etc/php/php.ini закомментированную строчку с date.timezone и впишем в нее:

```
date.timezone = Europe/Moscow
```

PHP-FPM

- В качестве среды исполнения PHP-скриптов будет использоваться php-fpm.
- Это сервер, принимающий запросы по протоколу fastcgi. По умолчанию он принимает запросы на 127.0.0.1:9000.
- php-fpm при старте сразу запускает набор воркеров для обработки запросов. В случае нехватки он запускает дополнительных воркеров вплоть до лимита.

```
service php-fpm start
```

Подготовка содержимого движка

- Начнем с создания директории /var/www/wiki. Владельцем этой директории должен быть пользователь, под которым запущен php-fpm
- Нужно скачать тарбол LTS-версии (https://releases.wikimedia.org/mediawiki/1.23/mediawiki-1.23.15.tar.gz) и распаковать его содержимое в /var/www/wiki.

Конфигурация nginx

- Необходимо настроить отдачу статических файлов с помощью nginx и отправку запросов к php-скриптам на php-fpm по протоколу fastcgi
- Для этого необходимо: запросы на *.php сразу отправлять на php-fpm, а для прочих запросов проверять существование файла и если файла нет, видоизменять запрос и отправлять на php-fpm

Конфигурация nginx (продолжение)

- В /etc/nginx/conf.d/default.conf добавим следующее:

```
location /wiki/ {
    root /var/www;
    index index.php;
    try_files $uri $uri/ @rewrite;
}

location @rewrite {
    rewrite ^/wiki/(.*)$ /wiki/index.php?title=$1&$args;
}

location ~ \.php$ {
    root /var/www;
    fastcgi_pass 127.0.0.1:9000;
    fastcgi_param SCRIPT_FILENAME /var/www/$fastcgi_script_name;
    include fastcgi_params;
}
```

- После этого нужно выполнить service nginx reload и открыть в браузере http://<ip виртуалки>/wiki/

Первичная настройка движка

- Следуя несложным указаниям, настройте движок. В конце вам предложат скачать файл LocalSettings.php и положить его в директорию движка - /var/www/wiki/.
- Каким образом можно автоматизировать установку, чтобы не приходилось выполнять ручные операции через веб-интерфейс?