Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и кибербезопасности
Высшая школа программной инженерии

Практическая работа №3

по дисциплине «Сети и телекоммуникации»

Выполнил: Яровой В. Д.
Группа: 5130904/00104

Проверил: Медведев Б. М.

Санкт-Петербург
2024

Содержание

1 Ответы на вопросы

1.1 Вопрос 1

TODO

Зачем нужен протокол HTTP, принцип работы

- Назначение HTTP: Протокол передачи гипертекста (HTTP) является основным протоколом передачи данных в сети Интернет. Он используется для обмена информацией между веб-браузерами и веб-серверами, позволяя загружать веб-страницы, изображения, видео и другие ресурсы.
- Принцип работы: Клиент (обычно веб-браузер) отправляет HTTP-запросы к серверу, запрашивая определенные ресурсы. Сервер отвечает HTTP-ответами, предоставляя запрошенные данные. Взаимодействие происходит по принципу «запрос-ответ», и каждое сообщение состоит из заголовка и (возможно) тела сообщения.

1.2 Вопрос 2

TODO

Формат HTTP-сообщения, HTTP-запросы, HTTP-ответы

- Формат HTTP-сообщения: HTTP-сообщение состоит из двух частей - заголовка и (опционально) тела. Заголовок содержит мета-информацию о сообщении, такую как метод запроса, код ответа, тип содержимого и другие параметры. Тело содержит собственно данные сообщения.
- HTTP-запросы: HTTP-запрос отправляется клиентом к серверу для запроса определенного ресурса.

Пример запроса:

```
GET /index.html HTTP/1.1
Host: www.example.com
```

Заголовки (Headers)

```
Host: www.example.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:100.0) Gecko/20100101 Firefox/
100.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
```

HTTP-ответы: HTTP-ответ возвращается сервером клиенту, содержащий статус выполнения запроса и, при необходимости, данные. Пример ответа:

```
HTTP/1.1 200 OK
Content-Type: text/html

<html>
  <body>
    <h1>Hello, World!</h1>
  </body>
</html>
```

1.3 Вопрос 3

TODO

HTTP-серверы, HTTP-клиенты, прокси-серверы

- HTTP-серверы: Это программы или устройства, которые принимают HTTP-запросы от клиентов и отправляют HTTP-ответы с запрошенными данными. Примеры: Apache, Nginx, Microsoft IIS.
- HTTP-клиенты: Это программы или браузеры, которые отправляют HTTP-запросы к серверам и обрабатывают HTTP-ответы. Примеры: браузеры (Chrome, Firefox), утилиты командной строки (cURL).
- Прокси-серверы: Это серверы, которые действуют как посредники между клиентами и серверами. Они могут кэшировать ресурсы, фильтровать трафик, улучшать безопасность и управлять доступом. Примеры: Squid, Nginx (в режиме прокси).

2 Практические задание

2.1 Задание 1

TODO

Настройте свой веб-сервер (nginx+php+mysql или что-то подобное). Контент на сервере мне не важен, подойдет любая дефолтная CMS (Wordpress, Drupal итд)

2.1.1 Настройка

Будем настривать **Wordpress** с **nginx+php+mysql** в **docker**:

1. Напишем **docker-compose**:

```
version: '3.9'

services:
mysql:
image: mysql:8.0
container_name: mysql8
restart: unless-stopped
env_file: .env
volumes:
- dbfile:/var/lib/mysql
command: '--default-authentication-plugin=mysql_native_password'
networks:
- app

wp:
image: wordpress:5.7.0-php8.0-fpm
container_name: wordpress-5.7.0-php8.0-fpm
depends_on:
- mysql
restart: unless-stopped
env_file: .env
environment:
- WORDPRESS_DB_HOST=mysql:3306
- WORDPRESS_DB_USER=$MYSQL_USER
- WORDPRESS_DB_PASSWORD=$MYSQL_PASSWORD
- WORDPRESS_DB_NAME=$MYSQL_DATABASE
volumes:
- www-html:/var/www/html
networks:
- app

nginx:
image: nginx:1.19.8-alpine
depends_on:
- wp
container_name: nginx-1.19.8-alpine
restart: unless-stopped
ports:
- "80:80"
volumes:
- www-html:/var/www/html
```

```
- ./nginx-conf.d:/etc/nginx/conf.d
networks:
- app

volumes:
www-html:
dbfile:

networks:
app:
driver: bridge
```

1. MySQL Service (mysql):
   - Используется образ MySQL версии 8.0.
   - Название контейнера: mysql8.
   - Перезапускается при необходимости.
   - Использует файл с переменными окружения (.env) для настройки окружения.
   - Данные MySQL сохраняются в Docker Volume dbfile.
   - Задана команда для использования стандартной аутентификации MySQL.
2. WordPress Service (wp):
   - Используется образ WordPress версии 5.7.0 с PHP версии 8.0 в роли FPM (FastCGI Process Manager).
   - Название контейнера: wordpress-5.7.0-php8.0-fpm.
   - Зависит от сервиса MySQL (mysql) и перезапускается при необходимости.
   - Использует файл с переменными окружения (.env) для настройки окружения.
   - Данные WordPress сохраняются в Docker Volume www-html.
   - Связан с сетью app.
3. Nginx Service (nginx):
   - Используется образ Nginx версии 1.19.8-alpine.
   - Название контейнера: nginx-1.19.8-alpine.
   - Зависит от сервиса WordPress (wp) и перезапускается при необходимости.
   - Пробрасывает порт 80.
   - Монтирует Docker Volume www-html для обмена файлами с WordPress и локальный каталог ./nginx-conf.d для -настроек Nginx.
   - Связан с сетью app.
4. Volumes:
   - www-html: Используется для хранения данных WordPress и общего доступа с контейнерами wp и nginx.
   - dbfile: Используется для хранения данных MySQL.
5. Networks:
   - app: Используется для связи контейнеров в рамках одной сети.

2. Создадим директорию nginx-conf.d
3. Внутри nginx-conf.d создаим файл конфигурации .conf

```
server {
        listen 80;
        listen [::]:80;

        server_name  my.server.ru;

        index index.php index.html index.htm;
```

```
        root /var/www/html;

        location / {
                try_files $uri $uri/ /index.php$is_args$args;
        }

        location ~ \.php$ {
                try_files $uri =404;
                fastcgi_split_path_info ^(.+\.php)(/.+)$;
                fastcgi_pass wp:9000;
                fastcgi_index index.php;
                include fastcgi_params;
                fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
                fastcgi_param PATH_INFO $fastcgi_path_info;
        }

        location ~ /\.ht {
                deny all;
        }

        location = /favicon.ico {
                log_not_found off;
        }
        location = /robots.txt {
                log_not_found off;
        }
        location ~* \.(css|gif|ico|jpeg|jpg|js|png)$ {
                expires max;
                log_not_found off;
        }
}
```

4. Добавим в nginx-conf.d файл .env с информацие о базе данных

```
MYSQL_ROOT_PASSWORD=Be$t)P@s$w0rdR00T-u$eR
MYSQL_USER=wp_db_user
MYSQL_PASSWORD=W0rdpres$%d@taBa$e%paS$worD
MYSQL_DATABASE=wp_db
```

5. Запустим веб сервер командой

```
docker compose up -d
```

6. Можем просматривать логи с помощью команд

```
docker-compose logs php
docker-compose logs mysql
docker-compose logs nginx
```

7. завершаем работу серверам

```
docker compose down
```

2.1.2 Результаты

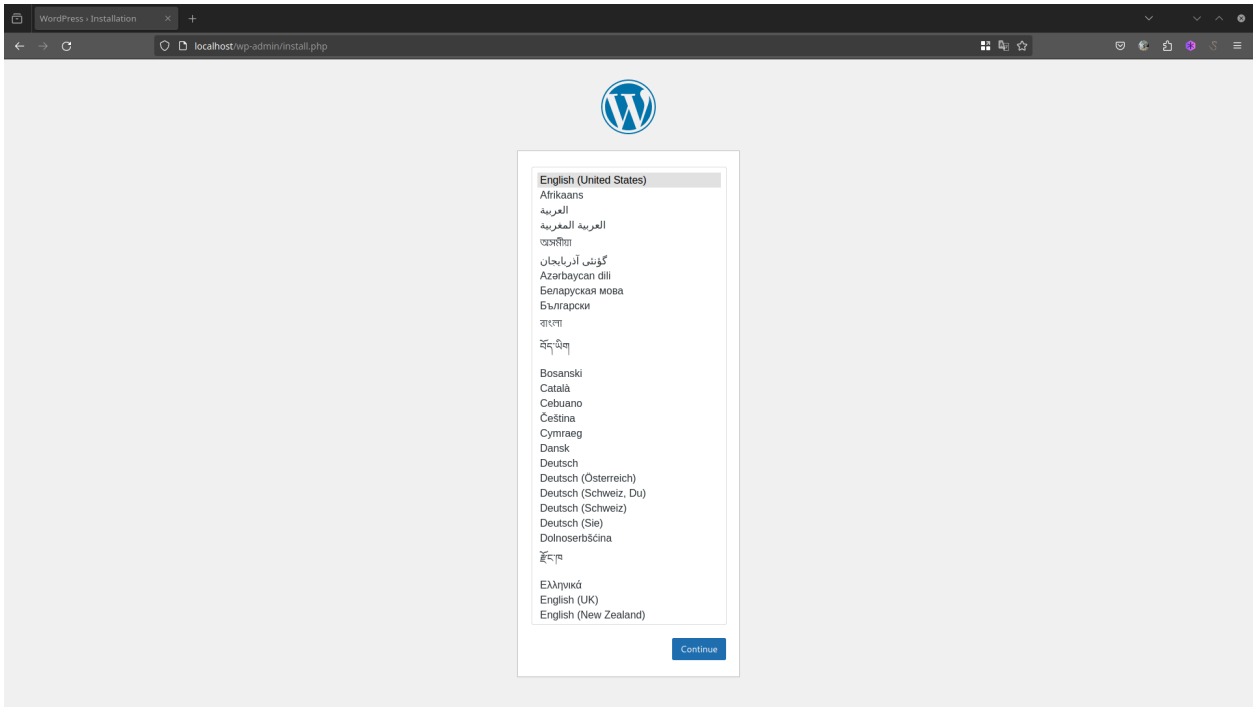

Рис. 1. Шаг 1

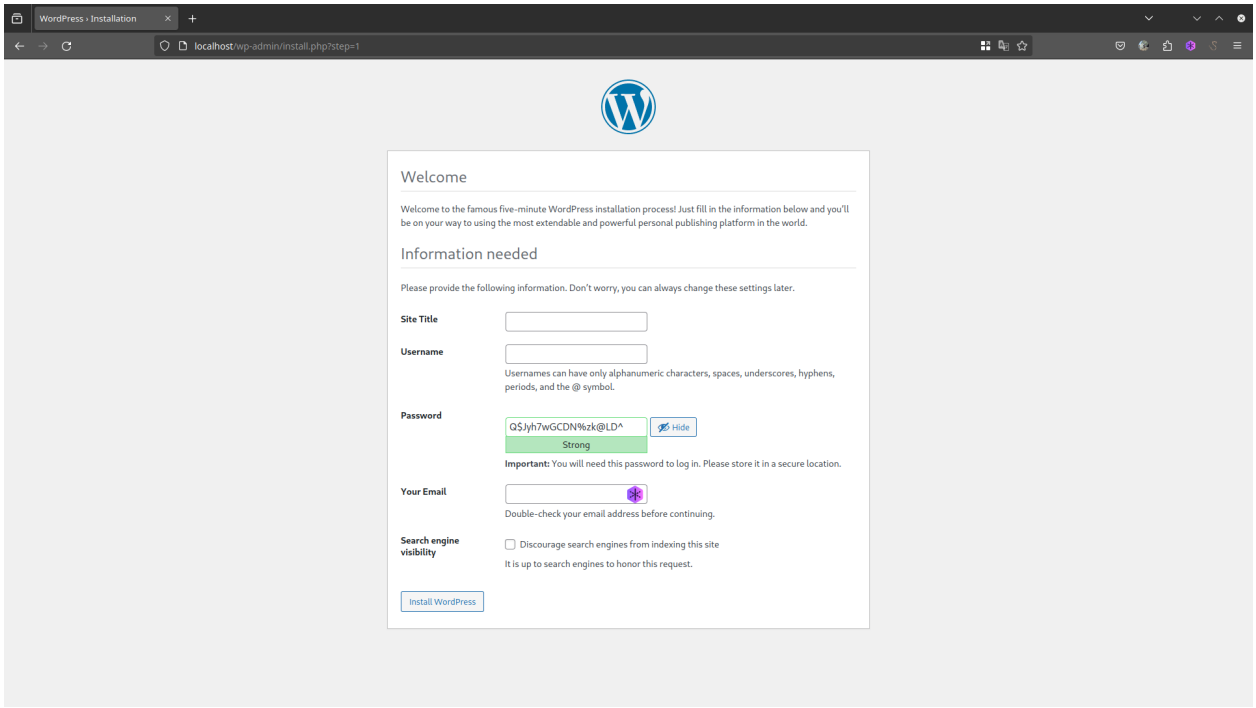

Рис. 2. Шаг 2

2.2 Задание 2

TODO

Расскажите, какие настройки для оптимизации можно использовать (увеличение скорости работы, увеличение надежности, SSL)

Оптимизация веб-сервера и приложений в Docker-контейнерах может включать в себя различные настройки для увеличения производительности, надежности и безопасности. Ниже приведены некоторые общие рекомендации:

2.2.1 Nginx:

1. Оптимизация конфигурации Nginx:
   - Используйте оптимальные параметры воркеров и соединений.
   - Настройте буферизацию и кеширование для статических ресурсов.
   - Включите gzip-сжатие для уменьшения объема передаваемых данных.
2. SSL-настройки:
   - Используйте современные версии протоколов TLS.
   - Включите Perfect Forward Secrecy (PFS) для усиления безопасности.
3. Keepalive и таймауты:
   - Настройте keepalive_timeout и keepalive_requests для эффективного использования соединений.
   - Установите разумные значения таймаутов для обработки запросов.

2.2.2 PHP-FPM:

1. Оптимизация пула PHP-FPM:
   - Настройте размеры пула и параметры ожидания соединений, чтобы соответствовать объему запросов.
   - Используйте динамическое масштабирование пула для более эффективного использования ресурсов.
2. Оптимизация PHP:
   - Включите оптимизации OPcache для уменьшения времени загрузки скриптов.
   - Установите разумные значения параметров memory_limit и max_execution_time.

2.2.3 MySQL:

1. Настройка параметров MySQL:
   - Оптимизируйте параметры конфигурации MySQL, такие как innodb_buffer_pool_size, query_cache_size.

3 Вывод

В результате проделанной работы были настроены и развернуты контейнеры с использованием Docker Compose для веб-сервера (Nginx), сервера приложений (PHP-FPM), базы данных (MySQL), и веб-приложения (WordPress). Осуществлены оптимизации настроек для улучшения производительности, безопасности и поддержки SSL. Система готова к использованию, обеспечивая надежное и эффективное развертывание веб-приложения.