

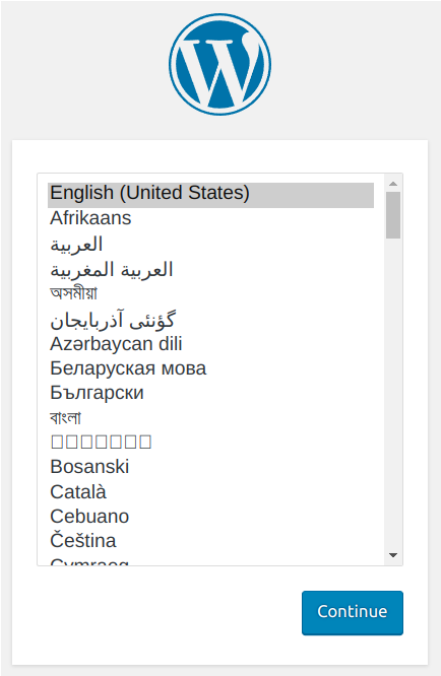
Выполнение практического задания к семинару №7


Все команды выполняются от имени root

Задание	Выполнение
<p>Установить в вирт. машину (Ubuntu) и настроить набор контейнеров для работы WordPress через docker compose (инструкция: https://www.digitalocean.com/community/tutorials/how-to-install-wordpress-with-docker-compose-ru)</p> <p>Часть с настройкой certbot и HTTPS не выполнялась, т.к. нет настоящего домена и белого IP</p>	
<p>Шаг 1 – настройка конфигурации веб-сервера</p>	
Создать директорию проекта с именем wordpress и перейти в эту директорию	<pre>mkdir wordpress && cd wordpress</pre>
Создать директорию для файла конфигурации nginx	<pre>mkdir nginx-conf</pre>
Создать файл настроек серверного блока с директивами для сервера nginx , а также блоками расположения корневой директории документов, обработки PHP и запросов статического контента	<pre>nano nginx-conf/nginx.conf</pre> <p><u>содержимое файла настроек (директивы):</u></p> <pre>server { listen 80; # port 80 IPv4 listen [::]:80; # port 80 IPv6 server_name example.com www.example.com; # real name of my server (here - dummy) index index.php index.html index.htm; # indexes to process outer requests root /var/www/html; # set-up root directory - to mount in Dockerfile location / { # to check files relevant to any single URI request (by index.php) try_files \$uri \$uri/ /index.php\$is_args\$args; } location ~ /\.php\$ { # to process PHP-requests and proxy them into wordpress try_files \$uri =404; fastcgi_split_path_info ^(.+\.php)(/.+)\$; fastcgi_pass wordpress: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 { # to process .htaccess files (nginx does not serve them) deny all; # never be served to users } location = /favicon.ico { # requests to favicon.ico will not be logged log_not_found off; access_log off; } }</pre>

	<pre> location = /robots.txt { # requests to robots will not be logged log_not_found off; access_log off; allow all; } location ~* \.(css gif ico jpeg jpg js png)\$ { # turns off logging to statics expires max; log_not_found off; } } </pre>
Шаг 2 – настройка переменных среды	
Создать файл .env для конфиденциальных значений среды (пароль root для MySQL, имя пользователя и пароль, которые WordPress будет использовать для доступа к БД (файл в главной директории проекта: ~/wordpress)	<p>nano .env</p> <p><u>содержимое файла:</u> MYSQL_ROOT_PASSWORD=your_root_password MYSQL_USER=your_wordpress_database_user MYSQL_PASSWORD=your_wordpress_database_password</p>
Добавить файл .env (и файлы разработки) в файлы .gitignore и .dockerignore – ограничить распространение файла	<p>nano .gitignore nano .dockerignore</p> <p>добавить .env (а также .git, docker-compose.yml, .dockerignore)</p>
Шаг 3 – определение служб с помощью Docker Compose	
Создать файл docker-compose.yml для определения служб (контейнеров)	nano docker-compose.yml
Добавить в файл docker-compose.yml код для определения его версии и службы базы данных db (отступы для вложенности – 2 пробела)	<pre> version: '3' # version of the yaml file services: # each service below – separate container db: # name of the service image: mysql:8.0 # which image to use ("latest" will update itself => may conflict) container_name: db # name of the container restart: unless-stopped # here restart until stopped manually; "no" by default env_file: .env # additional environment variables environment: # add additional env variables - MYSQL_DATABASE=wordpress volumes: # mounting dbdata directory to the /var/lib/mysql in the container - dbdata:/var/lib/mysql command: '--default-authentication-plugin=mysql_native_password' #override CMDs networks: - app-network # this application will join app-network specified below </pre>
Добавить в файл docker-compose.yml код для определения службы wordpress (отступы для вложенности – 2 пробела)	<pre> wordpress: depends_on: # order of dependency => will start after db container - db image: wordpress:5.1.1-fpm-alpine container_name: wordpress restart: unless-stopped env_file: .env environment: - WORDPRESS_DB_HOST=db:3306 # MySQL server on db container, standard MySQL p.3306 </pre>

	<pre>- WORDPRESS_DB_USER=\$MYSQL_USER # using value from .env - WORDPRESS_DB_PASSWORD=\$MYSQL_PASSWORD # using value from .env - WORDPRESS_DB_NAME=wordpress # the same as in MySQL volumes: - wordpress:/var/www/html networks: - app-network</pre>																
Добавить в файл docker-compose.yml код для определения службы webserver (отступы для вложенности – 2 пробела)	<pre>webserver: depends_on: - wordpress image: nginx:1.15.12-alpine container_name: webserver restart: unless-stopped ports: # exposing port 80 to enable configs in nginx.conf - "80:80" volumes: - wordpress:/var/www/html - ./nginx-conf:/etc/nginx/conf.d # any change to file on host reflects in container networks: - app-network</pre>																
Добавить в файл docker-compose.yml код для определения томов и сети (отступы для вложенности – 2 пробела)	<pre>volumes: # containers' data is stored in Host's: /var/lib/docker/volumes/ -> shared use wordpress: dbdata: networks: # enables link <-> containers as they are on the same Docker daemon host app-network: driver: bridge # opens all ports <-> containers on the same bridge network</pre>																
Запустить контейнеры (для работы в фоне)	<pre>docker-compose up -d</pre> <pre>Creating db ... done Creating wordpress ... done Creating webserver ... done root@boris-VB:/home/boris/wordpress#</pre>																
Проверка работы контейнеров	<pre>docker-compose ps</pre> <pre>root@boris-VB:/home/boris/wordpress# docker-compose ps</pre> <table><thead><tr><th>Name</th><th>Command</th><th>State</th><th>Ports</th></tr></thead><tbody><tr><td>db</td><td>docker-entrypoint.sh --def ...</td><td>Up</td><td>3306/tcp, 33060/tcp</td></tr><tr><td>webserver</td><td>nginx -g daemon off;</td><td>Up</td><td>0.0.0.0:80->80/tcp, :::80->80/tcp</td></tr><tr><td>wordpress</td><td>docker-entrypoint.sh php-fpm</td><td>Up</td><td>9000/tcp</td></tr></tbody></table> <pre>root@boris-VB:/home/boris/wordpress#</pre>	Name	Command	State	Ports	db	docker-entrypoint.sh --def ...	Up	3306/tcp, 33060/tcp	webserver	nginx -g daemon off;	Up	0.0.0.0:80->80/tcp, :::80->80/tcp	wordpress	docker-entrypoint.sh php-fpm	Up	9000/tcp
Name	Command	State	Ports														
db	docker-entrypoint.sh --def ...	Up	3306/tcp, 33060/tcp														
webserver	nginx -g daemon off;	Up	0.0.0.0:80->80/tcp, :::80->80/tcp														
wordpress	docker-entrypoint.sh php-fpm	Up	9000/tcp														
Запуск установки WordPress через веб-интерфейс (в окне браузера в Host – после проброса портов 9080:80 в настройках VB →	<pre>localhost:9080</pre> <p>→ localhost:9080/wp-admin/install.php</p>																

<p>Network → Advanced → Port Forwarding)</p>	
<p><i>Запустить два контейнера, связанные одной сетью (используя документацию). Первый контейнер БД (например, образ mariadb:10.8), второй контейнер — phpmyadmin. Получить доступ к БД в первом контейнере через второй контейнер (веб-интерфейс phpmyadmin)</i></p>	
<p>Создать директорию проекта с именем mariadb и перейти в эту директорию</p>	<pre>mkdir mariadb && cd mariadb</pre>
<p>Создать файл docker-compose.yml для определения служб (контейнеров)</p>	<pre>nano docker-compose.yml</pre>
<p>Добавить в файл docker-compose.yml код для определения его версии и службы базы данных db (образ mariadb:10.8). Конфиденциальную информацию можно разместить в отдельном файле .env (см. Задание №1), здесь – прямо в конфигурационном файле (отступы для вложенности – 2 пробела)</p>	<pre>version: '3.1' services: db: image: mariadb:10.8 restart: always environment: MARIADB_ROOT_PASSWORD: pass123 MYSQL_USER: "boris" MYSQL_PASSWORD: "123" volumes: - "./mariadb/data:/var/lib/mysql/data/" - "./mariadb/logs:/var/lib/mysql/logs/"</pre>

<p>Добавить в файл docker-compose.yml код для определения службы phpmyadmin (отступы для вложенности – 2 пробела)</p>	<p>phpmyadmin: image: phpmyadmin:latest restart: always ports: - 8081:80 environment: - PMA_ARBITRARY=1</p>
<p>Запустить контейнеры (для работы в фоне)</p>	<p>docker-compose up -d</p> <pre>root@boris-VB:/home/boris/mariadb# docker-compose up -d Creating network "mariadb_default" with the default driver Pulling db (mariadb:10.8)... 10.8: Pulling from library/mariadb d1669123f281: Pull complete 7942299fe584: Pull complete ca116927bbe1: Pull complete 9c0f0b5293ed: Pull complete ee0988afd61a: Pull complete 82d81fccd49d: Pull complete 7e361405ea73: Pull complete be365127aa3f: Pull complete Digest: sha256:456709ab146585d6189da05669b84384518baecd83670c9e5221f8c20a47cf1e Status: Downloaded newer image for mariadb:10.8 Creating mariadb_db_1 ... done Creating mariadb_phpmyadmin_1 ... done</pre>
<p>Сделать проброс портов в VirtualBox хоста для доступа к phpMyAdmin</p>	<p>VB → Network → Advanced → Port Forwarding → new rule 9081:8081</p>
<p>Доступ к MariaDB из браузера хоста</p>	<p>Localhost :9081</p> <p>→</p> 

phpMyAdmin

Recent

Favorites

information_schema

Type to filter these, Enter

1 >>>

New

ALL_PLUGINS

APPLICABLE_ROLES

CHARACTER_SETS

CHECK_CONSTRAINTS

CLIENT_STATISTICS

COLLATIONS

COLLATION_CHARACTER

COLUMNS

COLUMN_PRIVILEGES

ENABLED_ROLES

ENGINES

EVENTS

FILES

GEOMETRY_COLUMNS

GLOBAL_STATUS

GLOBAL_VARIABLES

INDEX_STATISTICS

INNODB_BUFFER_PAGE

INNODB_BUFFER_PAGE

INNODB_BUFFER_POOL

INNODB_CMP

INNODB_CMPMEM

INNODB_CMPMEM_RESE

INNODB_CMP_PER_INDE

Server: db

DatabasesSQLStatusExportImportSettingsVariablesCharsetsEnginesPlugins

General settings

Change password

Server connection collation: utf8mb4_unicode_ci

More settings

Appearance settings

Language: English

Theme: pmahomme View all

Database server

Server: db via TCP/IP

Server type: MariaDB

Server connection: SSL is not being used

Server version: 10.8.8-MariaDB-1:10.8.8+maria~ubu2204 - mariadb.org binary distribution

Protocol version: 10

User: boris@172.22.0.2

Server charset: UTF-8 Unicode (utf8mb4)

Web server

Apache/2.4.57 (Debian)

Database client version: libmysql - mysqlnd 8.2.10

PHP extension: mysqli curl mbstring sodium

PHP version: 8.2.10

phpMyAdmin

Version information: 5.2.1 (up to date)

Documentation

Official Homepage

Contribute

Get support

List of changes

License