

Міністерство освіти і науки України
НТУУ «КПІ ім. Ігоря Сікорського»
Навчально-науковий інститут атомної та теплової енергетики
Кафедра цифрових технологій в енергетиці

Лабораторна робота №9-1

з дисципліни «Операційна система UNIX»

Тема «Створення проекту для web-розробки, який
складається з наборів контейнерів з використанням Docker
Compose та Dockerfile»

Варіант №22

Студента 2-го курсу НН ІАТЕ гр. ТР-12

Ковальова Олександра

Перевірила: д.т.н., проф. Левченко Л. О.

Мета роботи. Ознайомитися та набути навичок написання скрипта Dockerfile для створення контейнеру, встановлення Docker Compose, створення контейнерів (сервісів) NGINX, PHP, FPM, MySQL для розробки web-додатку для Magento2.

Теоретична частина.

Docker-compose – це утиліта від авторів оригінального Docker, яка дозволяє об'єднати процес створення контейнерів. Для цього використовуються yaml файли з назвою docker-compose.yml. Compose використовує файли YAML (YAML Ain't Markup Language – мова серіалізації даних) для зберігання конфігурації груп контейнерів. На початку треба вказати версію утиліти, яку можна дізнатись за допомогою команди docker-compose –v. Після цього треба вказати ключове слово version, і в лапках версію (після двокрапки). Потім, після ключового слова services можна перелічувати контейнери, та вказувати їм певні властивості.

Повний опис усіх команд Compose знаходиться на сайті Docker <https://docs.docker.com/compose/reference/>.

Docker-compose створює один образ (base) на основі всіх, що входять в нього.

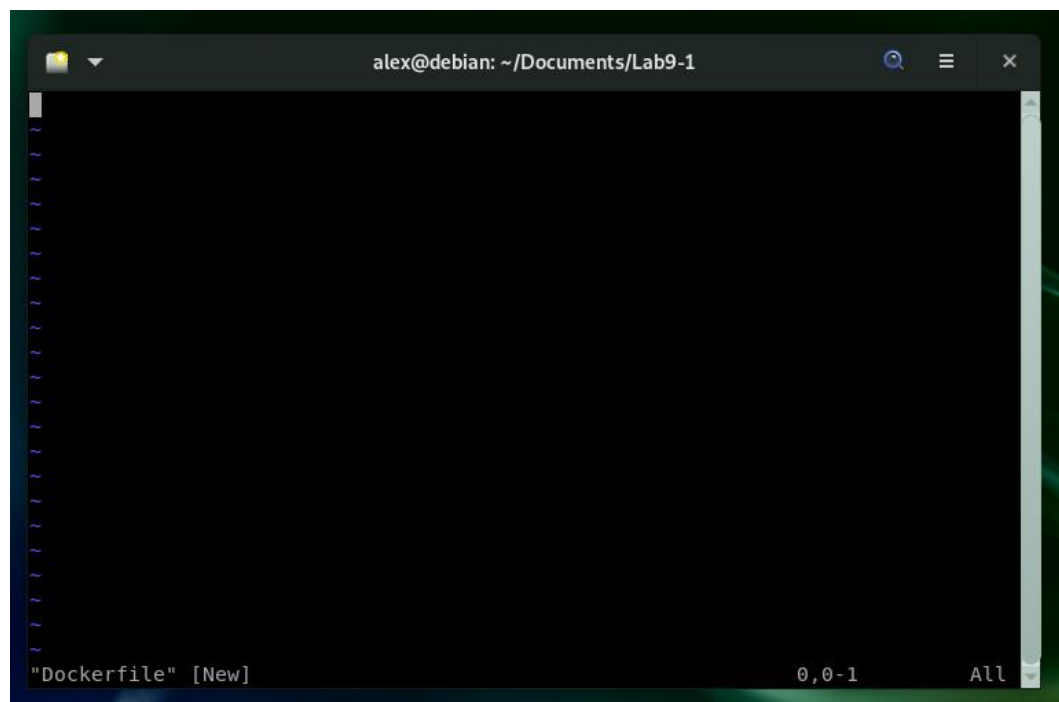
Також, утиліта має механізм Docker з'єднання (links) – найпростіший спосіб забезпечення обміну інформацією між контейнерами на одному хості. Тобто передається інформація про IP-адресу та відкриті порти з одного контейнера у другий. Це дуже спрощує роботу.

Теоретична частина (на практиці).

MongoDB – найбільш популярна нереляційна база даних. Нереляційна – тобто, не використовує стандартну схему таблиць та стовбців [1]. В цих базах даних використовується модель зберігання даних, яка оптимізована під конкретні вимоги типу збережених файлів.

Задача: створити власний образ для встановлення MongoDB.

Створимо порожній файл з назвою Dockerfile (без розширення) за допомогою утиліти vim. Команда – vim Dockerfile:



Коментарі в Dockerfile треба писати після символу решітки. Образ будемо створювати на основі Debian. Запишемо це:

```
alex@debian: ~/Documents/Lab9-1
#####
# Dockerfile to build MongoDB container images
# Based on Debian
#####
```

Для того щоб вказати Dockerfile'у що хочемо створити базовий образ на основі певного, треба вказати ключове слово From. Цей рядок обов'язковий:

```
alex@debian: ~/Documents/Lab9-1
#####
# Dockerfile to build MongoDB container images
# Based on Debian
#####

# Set the base image to Debian
FROM debian
```

Після цього використовуємо інструкцію RUN – вона запускає певні команди в процесі створення образу. Зараз вона потрібна для того, щоб оновити список пакетів та встановити пакет gnupg – він відповідає за створення та підписання електронних цифрових підписів. Використовуємо ключ -y, який означає «На всі питання Так чи Ні відповідати Так». Тобто, якщо оболонка запитає, чи можна скачати п мегабайт для пакету – не треба вказувати свою відповідь, вона автоматично буде «Так». Окрім цього, доцільно було б встановити пакет software-properties-common. За допомогою нього можна додавати репозиторії в список без редагування файлу sources.list, використовуючи команду add-apt-repository. Також, знадобиться команда wget – для неї потрібно завантажити пакет. Виведення займає багато місця, тому переправляємо звичайне виведення (STDIN, дескриптор каналу 1) в пустоту. Будуть виводитись на екран лише помилки та попередження.

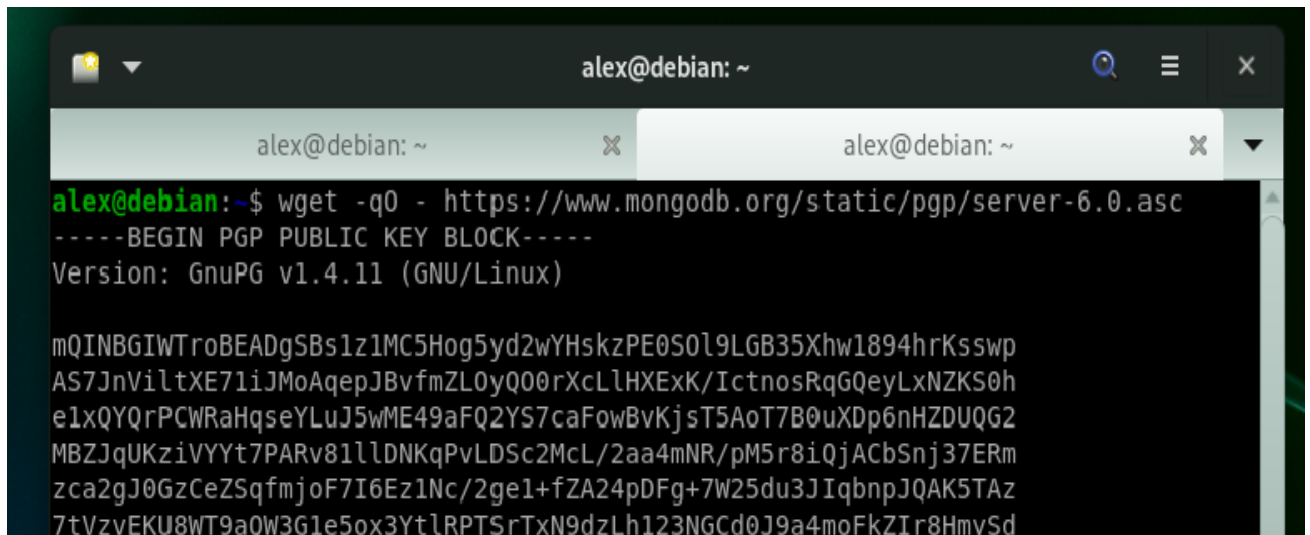
```
# Update the repository sources list and install gnupg2, software-properties-common, wget
RUN apt-get update 1> $VOID

RUN apt-get install -y gnupg 1> $VOID
RUN apt-get install -y software-properties-common 1> $VOID
RUN apt-get install -y wget 1> $VOID
```

В якості пустоти використовується змінна VOID. Вона записана за допомогою ключового слова ENV. В неї записана адреса /dev/null. Це спеціальний файл, який є символьним пристроєм, представляє собою «пустоту». Таким чином можна подавити виведення, переправляючи все в цей файл.

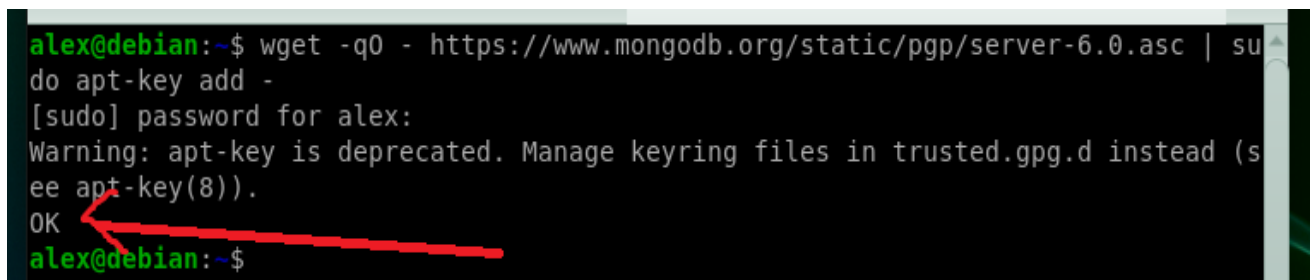
```
ENV VOID /dev/null
```

Додамо ключ верифікації пакету за допомогою утиліти apt-key. Для початку, ключ потрібно отримати – це можна зробити програмою wget, яка відповідає за завантаження певних даних [2]. Ключ -q0 означає, що помилки не треба виводити. Також додаємо посилання на офіційний сайт mongodb, звідки буде завантажуватись ключ. Запустимо команду та отримаємо результат:



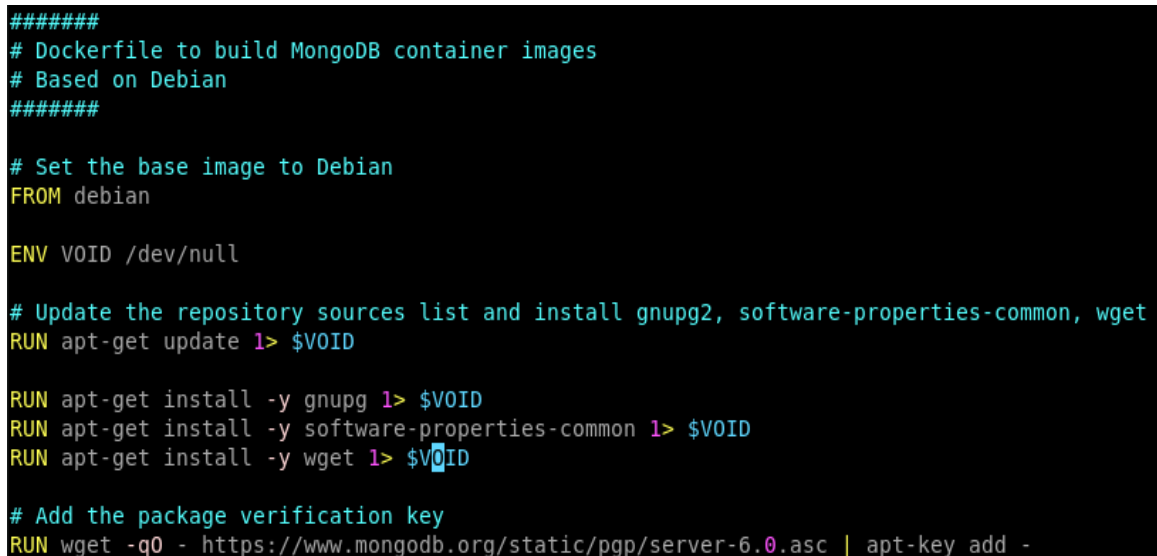
```
alex@debian: ~  
alex@debian:~$ wget -q0 - https://www.mongodb.org/static/pgp/server-6.0.asc  
-----BEGIN PGP PUBLIC KEY BLOCK-----  
Version: GnuPG v1.4.11 (GNU/Linux)  
  
mQINBGIWTroBEADgSBs1z1MC5Hog5yd2wYHskzPE0S0l9LGB35Xhw1894hrKsswp  
AS7JnViltXE71iJMoAqepJBvfmZLOyQ00rXcLlHXExK/IctnosRqGQeyLxNZKS0h  
e1xQYQrPCWRaHqseYLuJ5wME49aFQ2YS7caFowBvKjsT5AoT7B0uXDp6nHZDUQG2  
MBZJqUKziVYYt7PARv81llDNKqPvLDSc2McL/2aa4mNR/pM5r8iQjACbSnj37ERm  
zca2gJ0GzCeZSqfmjoF7I6Ez1Nc/2ge1+fZA24pDFg+7W25du3JIqbnPQAK5TAz  
7tVzvEKU8WT9aQW3G1e5ox3YtlRPTSrTxN9dzLh123NGCd0J9a4moFkZIr8HmySd
```

Цей ключ передаємо за допомогою конвеєра утиліти apt-key, яка додає його до бази своїх ключів.



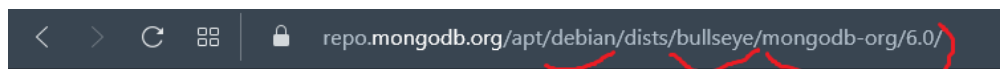
```
alex@debian:~$ wget -q0 - https://www.mongodb.org/static/pgp/server-6.0.asc | su  
do apt-key add -  
[sudo] password for alex:  
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (s  
ee apt-key(8)).  
OK  
alex@debian:~$
```

Додамо цю команду в Dockerfile.



```
#####  
# Dockerfile to build MongoDB container images  
# Based on Debian  
#####  
  
# Set the base image to Debian  
FROM debian  
  
ENV VOID /dev/null  
  
# Update the repository sources list and install gnupg2, software-properties-common, wget  
RUN apt-get update 1> $VOID  
  
RUN apt-get install -y gnupg 1> $VOID  
RUN apt-get install -y software-properties-common 1> $VOID  
RUN apt-get install -y wget 1> $VOID  
  
# Add the package verification key  
RUN wget -q0 - https://www.mongodb.org/static/pgp/server-6.0.asc | apt-key add -
```

За допомогою команди `add-apt-repository` з пакету `software-properties-common` додаємо репозиторій в список репозиторіїв apt – файл `sources.list`, який знаходиться за адресою `/etc/apt/sources.list`. Але – вказане зеркало з дистрибутивами (<http://downloads-distro.mongodb.org/repo/ubuntu-upstart>) вже не працює [3]. На заміну йому прийшов новий (<https://repo.mongodb.org>). Заходимо на нього, та обираємо потрібну версію:



Index of 6.0

[Parent Directory](#)

[Release](#)

[Release.gpg](#)

[main](#)

repo.mongodb.org

Додаємо це посилання в список репозиторіїв. Оновлюємо список пакетів. Встановлюємо пакет `mongodb-org`.

```
# Add MongoDB to the repository sources list
RUN add-apt-repository "deb http://repo.mongodb.org/apt/debian bullseye/mongodb-org/6.0 main"

# Update the repository sources list
RUN apt-get update

# Install MongoDB package (.deb)
RUN apt-get install -y mongodb-org 1> $VOID
```

Створюємо каталог для збереження даних.

```
# Create the default data directory
RUN mkdir -p /data/db
```

Даємо знати контейнеру, що зв'язок буде налаштований через порт 27017:

```
# Expose the default port
EXPOSE 27017
```

Відкриваємо для «зовнішнього світу» порт 27017, який є стандартним (дефолтним) для MongoDB:

```
# Default port to execute the entrypoint
CMD ["--port 27017"]
```

Встановлюємо додаток за замовчуванням: демон (службу) mongod.

```
# Set default container command
ENTRYPOINT usr/bin/mongod
```

Створюємо докер образ за допомогою команди build.

Команда: `sudo docker build -t alex-mongodb .`

Команда `build` використовується для збирання контейнеру в образ. Ключ `-t` означає, що ми хочемо присвоїти ім'я (tag) образу. Крапка в кінці означає «контекст» – тобто, в якому каталогі ми працюємо, в якому каталогі знаходиться `Dockerfile`.

```
alex@debian: ~/Documents/Lab9-1/Mongo
alex@debian:~/Documents/Lab9-1/Mongo$ sudo docker build -t alex-mongodb .
[sudo] password for alex:
Sending build context to Docker daemon  2.56kB
Step 1/14 : FROM debian
latest: Pulling from library/debian
32de3c850997: Pull complete
Digest: sha256:c66c0e5dc607baefefda1d9e64a3b3a317e4189c540c8eac0c1a06186fe353a1
Status: Downloaded newer image for debian:latest
--> 446440c01886
Step 2/14 : ENV VOID /dev/null
--> Running in 8e914433cd4b
Removing intermediate container 8e914433cd4b
--> 351f0ce99ef0
Step 3/14 : RUN apt-get update 1> $VOID
--> Running in 65e51c397f21
```

Можемо бачити, що всі команди виконуються покроково і окремо. Весь образ ділиться на шари, які можемо бачити на скріншоті вище.

При завантаженні є деякі попередження. Наприклад, те що продемонстроване нижче – це попередження пов'язане з утилітою `apt-get`. У неї є консольний інтерфейс, і через це `Docker` викидує дане попередження. Для того щоб це полагодити, треба встановити пакет `apt-utils`, та встановити значення для декількох змінних. Але, це подавить навіть попередження – тому, залишаємо як є. Це `safe-ignored` попередження, тобто його можна ігнорувати [4].

```
--> 83e895130302
Step 4/14 : RUN apt-get install -y gnupg 1> $VOID
--> Running in fbc9b452e0d7
debconf: delaying package configuration, since apt-utils is not installed
Removing intermediate container fbc9b452e0d7
--> b732daf5283d
Step 5/14 : RUN apt-get install -y software-properties-common 1> $VOID
```

Також є попередження щодо використання утиліти `apt-key`. Вона застарівша, і хорошою практикою вважається використання `gpg`. Але – це довше (код) та потребує додаткових пакетів, не дуже добре навантажувати образ зайвими пакетами.

```
--> 2be1fc98ec4c
Step 7/14 : RUN wget -qO - https://www.mongodb.org/static/pgp/server-6.0.asc | apt-key add -
--> Running in 79f8c73a815a
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key(8)).
OK
Removing intermediate container 79f8c73a815a
--> c3b98e471156
```

Образ успішно зібрався:

```
---> Running in 16458df6c36a
Removing intermediate container 16458df6c36a
---> d059a2c64e5e
Step 14/14 : ENTRYPOINT usr/bin/mongod
---> Running in 6be7a89e005a
Removing intermediate container 6be7a89e005a
---> 45c94c735662
Successfully built 45c94c735662
Successfully tagged alex-mongodb:latest
alex@debian:~/Documents/Lab9-1/Mongo$
```

За допомогою команди `docker images` переглянемо список образів. Як бачимо, образ підписаний. Також є базовий образ `debian`.

```
---> 45c94c735662
Successfully built 45c94c735662
Successfully tagged alex-mongodb:latest
alex@debian:~/Documents/Lab9-1/Mongo$ docker images
REPOSITORY          TAG             IMAGE ID         CREATED          SIZE
alex-mongodb         latest          45c94c735662    21 minutes ago   913MB
debian               latest          446440c01886    6 days ago       124MB
alex@debian:~/Documents/Lab9-1/Mongo$
```

Запускаємо контейнер за допомогою команди `docker run` з ключами `-it` (interactive, terminal) та `--name` для того щоб призначити ім'я новому контейнеру та запустити його в інтерактивному режимі. Створюємо його на основі нашого образу, який щойно був згенерований. Можемо бачити багато інформації в json форматі – це означає, що все працює добре.

```
alex@debian: ~/Documents/Lab9-1/Mongo
alex@debian:~/Documents/Lab9-1/Mongo$ sudo docker run -it --name AlexMongoDB alex-mongodb
{"t":{"$date":"2022-12-27T19:11:05.156+00:00"},"s":"I", "c":"CONTROL", "id":23285, "ctx":"-",
"msg":"Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'"
{"t":{"$date":"2022-12-27T19:11:05.161+00:00"},"s":"I", "c":"NETWORK", "id":4915701, "ctx":"mai
n", "msg":"Initialized wire specification", "details":{"socket":{"incomingExternalClient":{"minWireVersi
```

Docker Compose

Завантажуємо останню версію програми з GitHub за допомогою утиліти `curl`. Робимо файл виконуваним використовуючи `chmod`. Дивимось версію програми:

```
alex@debian: /usr/local/bin
alex@debian:~/Documents/Lab9-1/Mongo$ sudo curl -L "https://github.com/docker/compose/releases/download/v2.14.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
% Total % Received % Xferd Average Speed Time Time Time
Dload Upload Total Spent Left Speed
0 0 0 0 0 0 0 0 0:00:00 0:00:00 0:00:00 0
100 42.8M 100 42.8M 0 0 6974k 0 0:00:06 0:00:06 0:00:00 10.2M
alex@debian:~/Documents/Lab9-1/Mongo$ sudo chmod +x /usr/local/bin/docker-compose
alex@debian:~/Documents/Lab9-1/Mongo$ docker-compose --version
Docker Compose version v2.14.2
alex@debian:~/Documents/Lab9-1/Mongo$
```


Створимо файл docker-compose.yml для запуску контейнера hello-world:

```
alex@debian: ~/Documents/Lab9-1/hello-world$ cat docker-compose.yml
services:
  my-test:
    image: hello-world
```

Спочатку треба написати ключове слово “services”. Після нього повинен йти перелік сервісів та дії з ними.

Наш сервіс називається my-test та базується на образі hello-world. Запустимо командою docker compose up:

```
alex@debian: ~/Documents/Lab9-1/hello-world$ docker-compose up
[+] Running 2/2
  # my-test Pulled                                3.3s
  # 2db29710123e Pull complete                    0.8s
[+] Running 1/1
  # Container hello-world-my-test-1 Created        0.8s
Attaching to hello-world-my-test-1
hello-world-my-test-1 |
hello-world-my-test-1 | Hello from Docker!
hello-world-my-test-1 | This message shows that your installation appears to be working correctly.
hello-world-my-test-1 |
hello-world-my-test-1 | To generate this message, Docker took the following steps:
hello-world-my-test-1 | 1. The Docker client contacted the Docker daemon.
hello-world-my-test-1 | 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
hello-world-my-test-1 |    (amd64)
hello-world-my-test-1 | 3. The Docker daemon created a new container from that image which runs the
hello-world-my-test-1 |    executable that produces the output you are currently reading.
hello-world-my-test-1 | 4. The Docker daemon streamed that output to the Docker client, which sent
hello-world-my-test-1 |    it
hello-world-my-test-1 |    to your terminal.
hello-world-my-test-1 |
hello-world-my-test-1 | To try something more ambitious, you can run an Ubuntu container with:
hello-world-my-test-1 | $ docker run -it ubuntu bash
hello-world-my-test-1 |
```

Переглянемо список запущених сервісів через docker-compose, docker. Також переглянемо список образів докер.

```
alex@debian: ~/Documents/Lab9-1/hello-world$ docker-compose ps -a
NAME                IMAGE              COMMAND             SERVICE    CREATED          STATUS          PORTS
hello-world-my-test-1 hello-world        "/hello"            my-test    5 minutes ago    Exited (0) 5 minutes ago

alex@debian: ~/Documents/Lab9-1/hello-world$ docker ps -a
CONTAINER ID   IMAGE      COMMAND             CREATED          STATUS          PORTS          NAMES
3bbcd5a2f794   hello-world "/hello"            5 minutes ago    Exited (0) 5 minutes ago
518716aa4e89   alex-mongodb "/bin/sh -c usr/bin/" 2 hours ago      Exited (130) 2 hours ago
alex@debian: ~/Documents/Lab9-1/hello-world$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED          SIZE
alex-mongodb   latest   45c94c735662   2 hours ago     913MB
debian         latest   446440c01886   6 days ago      124MB
hello-world    latest   feb5d9fea6a5   15 months ago   13.3kB
```


Видалимо контейнер, а потім образ:

```
alex@debian: ~/Documents/Lab9-1/hello-world
alex@debian:~/Documents/Lab9-1/hello-world$ docker rm hello-world-my-test-1
hello-world-my-test-1
alex@debian:~/Documents/Lab9-1/hello-world$ docker image rm hello-world
Untagged: hello-world:latest
Untagged: hello-world@sha256:c77be1d3a47d0caf71a82dd893ee61ce01f32fc758031a6ec4cf1389248bb833
Deleted: sha256:feb5d9fea6a5e9606aa995e879d862b825965ba48de054caab5ef356dc6b3412
Deleted: sha256:e07ee1baac5fae6a26f30cabfe54a36d3402f96afda318fe0a96cec4ca393359
alex@debian:~/Documents/Lab9-1/hello-world$ docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS          NAMES
518716aa4e89   alex-mongodb   "/bin/sh -c usr/bin/..." 2 hours ago    Exited (130) 2 hours ago          AlexMongoDB
alex@debian:~/Documents/Lab9-1/hello-world$ docker images
REPOSITORY    TAG          IMAGE ID       CREATED        SIZE
alex-mongodb   latest       45c94c735662   3 hours ago    913MB
debian         latest       446440c01886   6 days ago     124MB
alex@debian:~/Documents/Lab9-1/hello-world$
```

Хід роботи

Завданням є формування робочої збірки контейнерів Nginx + PHP-fpm + MySQL + Magento2. Всі сервіси потрібно розкидати по контейнерах та скомпонувати за допомогою docker-compose.

Для початку увімкнемо автозапуск сервісу Docker. Для цього, треба використати інструмент systemctl – програмою, яка контролює головну службу (демон) системи – systemd. Спочатку увімкнемо автозапуск сервісу, а потім запустимо його.

```
alex@debian:~/Documents/Lab9-1$ sudo systemctl enable docker.service
[sudo] password for alex:
Synchronizing state of docker.service with SysV service script with /lib/systemd/sy
emd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable docker
alex@debian:~/Documents/Lab9-1$ sudo systemctl start docker.service
alex@debian:~/Documents/Lab9-1$
```

Створюємо структуру проекту (каталоги). В каталогі MySQL буде зберігатись база даних. В папці Nginx – логи, файл конфігурації та проект. В директорії PHP – Dockerfile та файл php.ini. В корені проекту буде знаходитись файл docker-compose.yml.

```
alex@debian: ~/Documents/Lab9-1/mage/Nginx
alex@debian:~/Documents/Lab9-1$ mkdir mage
alex@debian:~/Documents/Lab9-1$ cd mage
alex@debian:~/Documents/Lab9-1/mage$ mkdir MySQL Nginx PHP
alex@debian:~/Documents/Lab9-1/mage$ cd Nginx
alex@debian:~/Documents/Lab9-1/mage/Nginx$ mkdir core html Logs www
alex@debian:~/Documents/Lab9-1/mage/Nginx$ tree ../
../
├── MySQL
├── Nginx
│   ├── core
│   ├── html
│   ├── Logs
│   └── www
└── PHP

7 directories, 0 files
alex@debian:~/Documents/Lab9-1/mage/Nginx$
```

Створюємо конфігураційний файл для Nginx:

```
alex@debian: ~/Documents/Lab9-1/mage/nginx/core
alex@debian:~/Documents/Lab9-1/mage/nginx$ cd core
alex@debian:~/Documents/Lab9-1/mage/nginx/core$ vim nginx.conf
```

Редактором текста всю роботу слугує vim.

Це мінімальна конфігурація для того, щоб все спрацювало. У першому блоці описуємо порт, який будемо слухати. Потім перераховуємо всі можливі index сторінки, називаємо сервер, створюємо alias (псевдонім) для шляху, по якому знаходиться magento2. Пишемо, які логи нам потрібні і де вони повинні зберігатись. Записуємо, де повинен зберігатись magento2.

У другому блоці прописуємо параметри fastcgi. FastCGI – це клієнт-серверний протокол взаємодії вебсервера та програми.

Третій блок потрібен для вирішення проблеми відображення пустої сторінки. Це вказано в документації magento2.

Четвертий блок потрібен для вирішення помилки з favicon.ico.

Файл виглядає так:

```
server {
    listen 80;
    index index.php;
    server_name magento2.dev;
    set $MAGE_ROOT /var/www/magento2/pub;
    error_log /var/log/nginx/error.log;
    access_log /var/log/nginx/access.log;
    root $MAGE_ROOT;

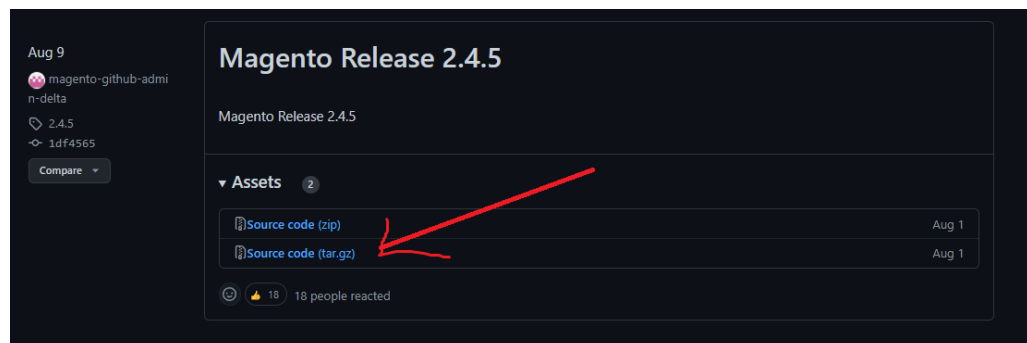
    location ~* \.php$ {
        try_files $uri $uri/ /index.php last;
        fastcgi_split_path_info (.+?\.php)(/.*);
        fastcgi_pass php:9000;
        fastcgi_index index.php;
        include fastcgi_params;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        fastcgi_param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

    location ~* .php/ {
        rewrite (.+\.php)/ $1 last;
    }

    location = /favicon.ico {
        return 204;
        access_log off;
        log_not_found off;
    }
}
```

Зауважимо, що шлях повинен бути до каталогу pub – інакше основна сторінка Magento не запуститься.

Завантажуємо magento2 з офіційного репозиторія на GitHub [5]:



Розпаковуємо у каталог /Nginx/www/magento2 :

```
alex@debian: ~/Documents/Lab9-1/mage/Ngi... x alex@debian: ~/Documents/Lab9-1/mage/Ngi... x
alex@debian: ~/Documents/Lab9-1/mage/Nginx/www$ tar -xzf magento2-2.4.5.tar.gz
alex@debian: ~/Documents/Lab9-1/mage/Nginx/www$ rm magento2-2.4.5.tar.gz
alex@debian: ~/Documents/Lab9-1/mage/Nginx/www$ mv magento2-2.4.5/ magento2/
alex@debian: ~/Documents/Lab9-1/mage/Nginx/www/magento2$ ls
app          COPYING.txt  LICENSE_AFL.txt  README.md
auth.json.sample  dev         LICENSE.txt      SECURITY.md
bin          generated   nginx.conf.sample  setup
CHANGELOG.md  grunt-config.json.sample  package.json.sample  var
composer.json  Gruntfile.js.sample      phpserver           vendor
composer.lock  lib          pub
alex@debian: ~/Documents/Lab9-1/mage/Nginx/www/magento2$
```

Але це невстановлений фреймворк. Забігаючи наперед, це не буде працювати. Потрібно його «встановити». Для цього потрібна утиліта Composer, яка довстановить залежності [6]. Це програма яка вирішує цю проблему спеціально для PHP фреймворків. Скачуємо Composer з офійного сайту:

Manual Download

If you prefer to download the phar manually, here are the available versions:

[Latest Stable \(sha256 / sha256sum / asc\)](#) for PHP 7.2+ users
[Latest Preview \(alpha/beta/RC\) \(sha256 / sha256sum / asc\)](#)
[Latest Snapshot \(sha256 / sha256sum\)](#)
[Latest 1.x \(sha256 / sha256sum\)](#)
[Latest 2.x \(sha256 / sha256sum / asc\)](#)
[Latest 2.2.x LTS \(sha256 / sha256sum / asc\)](#) for PHP 5.3 to 7.1 users

Composer (composer.phar) versions history

2.5.1	2022-12-22	sha256sum	f1b94fee11a5bd6a1aae5d77c8da269df27c705fcc806ebf4c8c2e6fa8645c20	asc	changelog
2.5.0	2022-12-20	sha256sum	b571610e5451785f76389a08e9575d91c3d6e38fee1df7a9708fe307013c8424	asc	changelog
2.4.4	2022-10-27	sha256sum	c252c2a2219956f88089ffc242b42c8cb9300a368fd3890d63940e4fc9652345	asc	changelog
2.4.3	2022-10-14	sha256sum	26d72f2790502bc9b22209e1cec1e0e43d33b368606ad227d327cccb388b609a	asc	changelog

Отримуємо phar файл. Це архів з додатком, написаним на PHP. Для запуску використовується команда php. Для установки потрібно перемістити файл у папку bin:

After running the installer following the [Download page](#) path:

```
mv composer.phar /usr/local/bin/composer
```

If you like to install it only for your user and avoid requirir

Починаємо установку composer:

```
alex@debian: /media/sf_Shared
alex@debian:/media/sf_Shared$ sudo mv composer.phar /usr/local/bin/composer
[sudo] password for alex:
alex@debian:/media/sf_Shared$ composer

Composer version 2.5.1 2022-12-22 15:33:54

Usage:
  command [options] [arguments]
```

Це була не зовсім установка, ми просто перемістили файл туди, де термінал розпізнає звертання до нього командою, без вказування шляху. Тепер, зберемо проект magento2. Переходимо в каталог magento2 та прописуємо composer install:

```
alex@debian: ~/Documents/Lab9-1/mage/nginx/www/magento2$ composer install
Installing dependencies from lock file (including require-dev)
Verifying lock file contents can be installed on current platform.
Warning: The lock file is not up to date with the latest changes in composer.json. You may be getting outdated dependencies. It is recommended that you run 'composer update' or 'composer update <package name>'
Your lock file does not contain a compatible set of packages. Please run composer update.

Problem 1
- Root composer.json requires PHP extension ext-bcmath * but it is missing from your system. Install or enable PHP's bcmath extension.
Problem 2
- Root composer.json requires PHP extension ext-dom * but it is missing from your system. Install or enable PHP's dom extension.
Problem 3
- Root composer.json requires PHP extension ext-gd * but it is missing from your system. Install or enable PHP's gd extension.
Problem 4
- Root composer.json requires PHP extension ext-intl * but it is missing from your system. Install or enable PHP's intl extension.
Problem 5
- Root composer.json requires PHP extension ext-pdo_mysql * but it is missing from your system. Install or enable PHP's pdo_mysql extension.
Problem 6
- Root composer.json requires PHP extension ext-simplexml * but it is missing from your system. Install or enable PHP's simplexml extension.
Problem 7
- Root composer.json requires PHP extension ext-soap * but it is missing from your system. Install or enable PHP's soap extension.
Problem 8
- Root composer.json requires PHP extension ext-xsl * but it is missing from your system. Install or enable PHP's xsl extension.
Problem 9
- Root composer.json requires PHP extension ext-zip * but it is missing from your system. Install or enable PHP's zip extension.
Problem 10
- aws/aws-sdk-php is locked to version 3.224.4 and an update of this package was not requested.
- aws/aws-sdk-php 3.224.4 requires ext-simplexml * -> it is missing from your system. Install or enable PHP's simplexml extension.
Problem 11
- laminas/laminas-feed is locked to version 2.17.0 and an update of this package was not requested.
- laminas/laminas-feed 2.17.0 requires ext-dom * -> it is missing from your system. Install or enable PHP's dom extension.
Problem 12
- laminas/laminas-soap is locked to version 2.10.0 and an update of this package was not requested.
- laminas/laminas-soap 2.10.0 requires ext-dom * -> it is missing from your system. Install or enable PHP's dom extension.
```

Бачимо багато помилок. Це через те, що відсутні пакети – composer не може приєднати їх до проекту [7]. Але є і проблема – ніякого скрипту для усунення цих помилок не існує. Тобто, потрібно завантажувати всі пакети по одному. Так як composer написаний на PHP, то й залежності всі потребують php розширень. В основному, це доповнення для роботи з базою даних, консолью, zip-файлами та інше.

```
1969 sudo apt install php-bcmath
1970 sudo apt install php-curl
1971 sudo apt install php-dom
1972 sudo apt install php-intl
1973 sudo apt install php-gd
1974 sudo apt install php-pdo_mysql
1975 sudo apt install php-mysql
1976 sudo apt install php-soap
1977 sudo apt install php-zip
```

Так як виявилось, що є проблеми, прописуємо update (install виконається автоматично):

```
alex@debian:~/Documents/Lab9-1/mage/Nginx/www/magento2$ composer update
Loading composer repositories with package information
Info from https://repo.packagist.org: #StandWithUkraine
Updating dependencies
Lock file operations: 1 install, 83 updates, 5 removals
- Removing laminas/laminas-math (3.5.0)
- Removing phpdocumentor/reflection-common (2.2.0)
- Removing phpdocumentor/reflection-docblock (5.3.0)
- Removing phpdocumentor/type-resolver (1.6.1)
- Removing phpspec/prophecy (v1.15.0)
- Upgrading aws/aws-sdk-php (3.224.4 => 3.255.6)
- Upgrading brick/varexporter (0.3.5 => 0.3.7)
- Upgrading codeception/codeception (4.1.31 => 4.2.2)
- Upgrading codeception/module-webdriver (1.4.0 => 1.4.1)
- Upgrading colinmollenhour/php-redis-session-abstract (v1.4.5 => v1.4.7)
- Upgrading composer/ca-bundle (1.3.2 => 1.3.4)
- Upgrading composer/composer (2.2.14 => 2.2.18)
```

Magento2 готовий до використання.

Настав час для налаштування php.

Переходимо в каталог /php/ та створюємо Dockerfile:

```
alex@debian:~/Documents/Lab9-1/mage/PHP$ touch Dockerfile php.ini
alex@debian:~/Documents/Lab9-1/mage/PHP$ vim Dockerfile
```


В цьому файлі написано які пакети та модулі потрібно встановити, вказали де знаходиться корінь, та куди скопіювати налаштування з `php.ini`. Але – там застаріла версія РНР. Змінюємо 7.0 на 7.4 та деякі пакети:

```
FROM php:7.4-fpm

RUN apt-get update && apt-get install -y \
curl \
wget \
git \
libfreetype6-dev \
libjpeg62-turbo-dev \
libxslt-dev \
libicu-dev \
libmcrypt-dev \
libpng-dev \
libxml2-dev \
libzip-dev \

&& pecl install mcrypt-1.0.4 && docker-php-ext-enable mcrypt \
&& docker-php-ext-install -j$(nproc) iconv mysqli pdo_mysql zip \
&& docker-php-ext-configure gd --with-freetype --with-jpeg \
&& docker-php-ext-install -j$(nproc) gd

RUN docker-php-ext-configure intl
RUN docker-php-ext-install intl
RUN docker-php-ext-install xsl
RUN docker-php-ext-install soap

RUN curl -sS https://getcomposer.org/installer | php -- --install-dir=/usr/local/bin --filename=composer

ADD php.ini /usr/local/etc/php/conf.d/40-custom.ini

WORKDIR /var/www/magento2

CMD ["php-fpm"]
```

Створюємо файл `php.ini` та заповнюємо за зразком від розробників:

```
alex@debian: ~/Documents/Lab9-1/mage/PHP
memory_limit = 2G

always_populate_raw_post_data = -1

cgi.fix_pathinfo = 1

fastcgi_split_path_info = 1

max_execution_time = 18000

flag_session.auto_start = off

zlib.output_compression = on

suhosin.session.cryptua = off

display_errors = off

~
```

Переходимо до кореня та налаштовуємо файл `docker-compose`. Це файл який будемо запускати, в якому прописані налаштування контейнерів, їх портів, так як вони будуть взаємодіяти між собою. На початку вказуємо версію `docker-compose`, яку можна дізнатися за допомогою команди `docker -v`. Також прописуємо ключове слово `services`. Воно означає, що нижче буде йти перелік сервісів:

```
alex@debian: ~/Documents/Lab
version: '2.14.2'

services:
```

Перший сервіс – nginx. Грубо кажучи, це веб-сервер. В параметрі image вказуємо значення nginx:latest. Це означає, що контейнер буде створюватись на основі образу останньої версії. Образ завантажиться з DockerHub. Параметр container_name відповідає за назву контейнера, він буде мати ім'я nginx. Ports відповідає за порти, які будуть доступні ззовні. 80 – це стандартний порт для http, 443 – https. Останній – захищений зв'язок, його використовувати не будемо, але завжди потрібно вказувати. Volumes – розділи контейнера. Тобто, вказуємо які папки хосту будуть відповідати каталогам з контейнера. Links – залежності.

```
nginx:
  image: nginx:latest
  container_name: nginx
  ports:
    - "80:80"
    - "443:443"
  volumes:
    - ./Nginx/core:/etc/nginx/conf.d
    - ./Nginx/www:/var/www/
    - ./Nginx/Logs:/var/log/nginx/
    - ./Nginx/html:/usr/share/nginx/html/
  links:
    - php
```

Сервіс бази даних. Створюємо контейнер з образу mysql. Прокидуємо порт на стандартний для баз даних – 3306. Environment – змінні, які передадуться в контейнер через args.

```
mysql:
  image: mysql:latest
  ports:
    - "3306:3306"
  container_name: mysql
  environment:
    - MYSQL_ROOT_PASSWORD=mypassword
    - MYSQL_DATABASE=magento2
    - MYSQL_USER=magento2
    - MYSQL_PASSWORD=magento2
  volumes:
    - ./MySQL:/var/lib/mysql
```

Контейнер php будемо збирати на основі каталогу з Dockerfile, який ми робили раніше, та файлу php.ini. Це логіка сайту, тому порти прокидувати не треба. phpmyadmin – сервіс для керування сайтом та базою даних, прокидуємо порт 8090.

```
php:
  build: ./PHP
  container_name: php-fpm
  volumes:
    - ./Nginx/www:/var/www/
  links:
    - mysql

phpmyadmin:
  image: phpmyadmin/phpmyadmin
  container_name: phpmyadmin
  ports:
    - 8090:80
  links:
    - mysql:db
```


Проект готовий! В корні проекту запускаємо docker-compose up:

```
alex@debian:~/Documents/Lab9-1/mage$ docker-compose up
[+] Running 0/38
:: mysql Pulling 11.1s
:: 0ed027b72ddc Waiting 2.4s
:: 0296159747f1 Waiting 2.4s
:: 3d2f9b664bd3 Waiting 2.4s
:: df6519f81c26 Waiting 2.4s
:: 36bb5e56d458 Waiting 2.4s
:: 054e8fde88d0 Waiting 2.4s
:: f2b494c50c7f Waiting 2.4s
:: 132bc0d471b8 Waiting 2.4s
:: 135ec7033a05 Waiting 2.4s
:: 5961f0272472 Waiting 2.4s
:: 75b5f7a3d3a4 Waiting 2.4s
:: phpmyadmin Pulling 11.1s
:: 214ca5fb9032 Downloading 11.41MB/31.38MB 3.8s
:: cd813a1b2cb8 Download complete 3.7s
:: 63cf7574573d Downloading 6.994MB/91.6MB 3.5s
:: 54c27146d16e Waiting 3.5s
:: 078f4450f949 Waiting 3.5s
:: 5f145e355bc4 Waiting 3.5s
```

Docker-compose запускає всі задачі окремо по контейнерам, а там вони запускаються послідовно. Тут можемо бачити процес завантаження образів з сервера (pulling).

Етап збірки закінчений. Йде запуск:

```
# 5f63362a3fa3 Pull complete 77.9s
[+] Building 10.2s (15/15) FINISHED
=> [internal] load .dockerignore 2.4s
=> => transferring context: 2B 0.2s
=> [internal] load build definition from Dockerfile 2.4s
=> => transferring dockerfile: 810B 0.6s
=> [internal] load metadata for docker.io/library/php:7.4-fpm 4.7s
=> [auth] library/php:pull token for registry-1.docker.io 0.0s
=> [1/9] FROM docker.io/library/php:7.4-fpm@sha256:3ac7c8c74b2b047c7cb273469d74fc0d59b857aa44043 0.1s
=> [internal] load build context 0.4s
=> => transferring context: 283B 0.0s
=> CACHED [2/9] RUN apt-get update && apt-get install -y curl wget git libfreetype6-dev libjpeg6 0.0s
=> CACHED [3/9] RUN docker-php-ext-configure intl 0.0s
=> CACHED [4/9] RUN docker-php-ext-install intl 0.0s
=> CACHED [5/9] RUN docker-php-ext-install xsl 0.0s
=> CACHED [6/9] RUN docker-php-ext-install soap 0.0s
=> CACHED [7/9] RUN curl -sS https://getcomposer.org/installer | php -- --install-dir=/usr/local 0.0s
=> CACHED [8/9] ADD php.ini /usr/local/etc/php/conf.d/40-custom.ini 0.0s
=> CACHED [9/9] WORKDIR /var/www/magento2 0.0s
=> exporting to image 0.5s
=> => exporting layers 0.1s
=> => writing image sha256:044037a5776a7e09874ffcd58f290432eb311de976bf4daa2dba732096f1ed0f 0.0s
=> => naming to docker.io/library/mage-php 0.0s
[+] Running 1/2
:: Network mage_default Created 6.8s
```

Всі контейнери запущені. Йде запуск серверу:

```
=> => naming to docker.io/library/mage-
[+] Running 5/5
:: Network mage_default Created
:: Container mysql Created
:: Container php-fpm Created
:: Container phpmyadmin Created
:: Container nginx Created
Attaching to mysql, nginx, php-fpm, phpm
mysql | 2022-12-30 00:07:33+00:00
mysql started
```

Бачимо що база даних готова до підключення. Пробуємо підключитись:

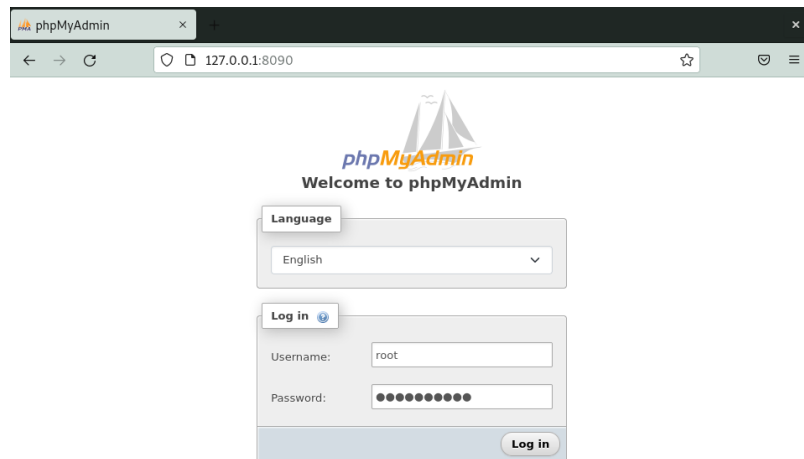
```
mysql | 2022-12-30T00:11:07.232690Z 0 [System] [MY-013602] [Server] Channel mysql_main configured
to support TLS. Encrypted connections are now supported for this channel.
mysql | 2022-12-30T00:11:07.259322Z 0 [Warning] [MY-011810] [Server] Insecure configuration for --
pid-file: Location '/var/run/mysqld' in the path is accessible to all OS users. Consider choosing a diff
erent directory.
mysql | 2022-12-30T00:11:09.104794Z 0 [System] [MY-011323] [Server] X Plugin ready for connections
. Bind-address: '::' port: 33060, socket: /var/run/mysqld/mysqlx.sock
mysql | 2022-12-30T00:11:09.104831Z 0 [System] [MY-010931] [Server] /usr/sbin/mysqld: ready for co
nnections. Version: '8.0.31' socket: '/var/run/mysqld/mysqld.sock' port: 3306 MySQL Community Server
- GPL.
```

Підключаємось по адресу localhost або 127.0.0.1 (можна вказати порт 80, 127.0.0.1:80). Бачимо запрошення до установки. Все працює! Збоку, також, бачимо консоль сервера – бачимо повідомлення від php-fpm про те, що хтось (ми) завантажив сторінку.



```
mysql | 2022-12-30T00:11:07.232690Z 0 [System] [MY-013602] [Server] Channel mysql_main configured
to support TLS. Encrypted connections are now supported for this channel.
mysql | 2022-12-30T00:11:07.259322Z 0 [Warning] [MY-011810] [Server] Insecure configuration for --
pid-file: Location '/var/run/mysqld' in the path is accessible to all OS users. Consider choosing a diff
erent directory.
mysql | 2022-12-30T00:11:09.104794Z 0 [System] [MY-011323] [Server] X Plugin ready for connections
. Bind-address: '::' port: 33060, socket: /var/run/mysqld/mysqlx.sock
mysql | 2022-12-30T00:11:09.104831Z 0 [System] [MY-010931] [Server] /usr/sbin/mysqld: ready for co
nnections. Version: '8.0.31' socket: '/var/run/mysqld/mysqld.sock' port: 3306 MySQL Community Server
- GPL.
php-fpm | 172.25.0.5 - 30/Dec/2022:00:14:15 +0000 "GET /index.php"
```

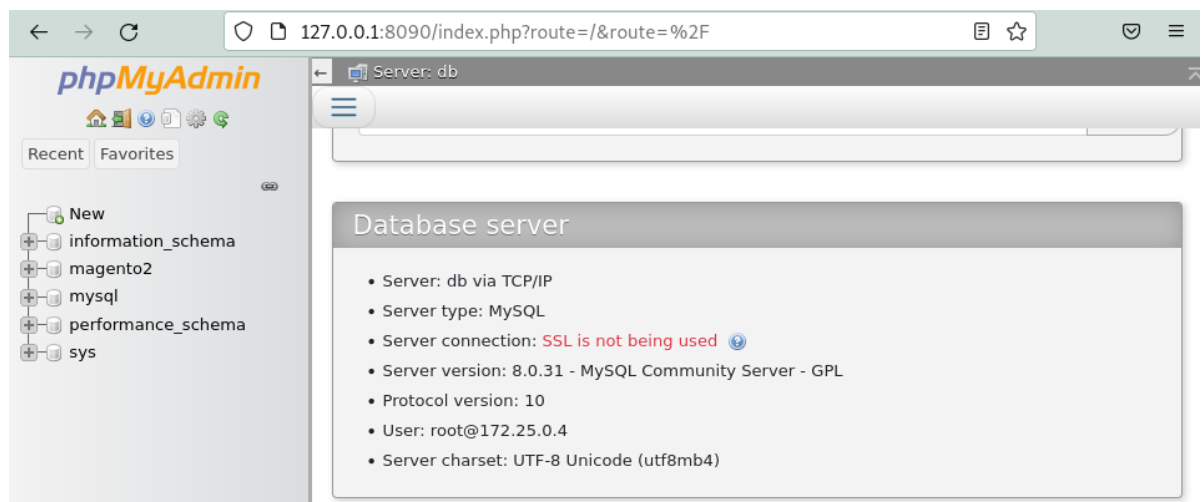
Але, також ще є сторінка адміністратора. Можемо підключитись також через localhost, але тут вже обов'язково вказувати порт – 8090, як було вказано в docker-compose.



В консолі відразу відображається спроба завантажити сторінку. Також, бачимо статус 200 (OK) – це означає, що все добре.

```
mysql | 2022-12-30T00:11:09.104831Z 0 [System] [MY-010931] [Server] /usr/sbin/mysqld: ready for connections. Version: '8.0.31' socket: '/var/run/mysqld/mysqld.sock' port: 3306 MySQL Community Server - GPL.
php-fpm | 172.25.0.5 - 30/Dec/2022:00:14:15 +0000 "GET /index.php"
phpmyadmin | 172.25.0.1 - [30/Dec/2022:00:20:37 +0000] "GET / HTTP/1.1" 200 6198 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0"
phpmyadmin | 172.25.0.1 - [30/Dec/2022:00:20:45 +0000] "GET /js/vendor/jquery/jquery.min.js?v=5.2.0 HTTP/1.1" 200 31255 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0"
phpmyadmin | 172.25.0.1 - [30/Dec/2022:00:20:48 +0000] "GET /js/dist/name-conflict-fixes.js?v=5.2.0 HTTP/1.1" 200 339 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0"
phpmyadmin | 172.25.0.1 - [30/Dec/2022:00:20:48 +0000] "GET /js/vendor/jquery/jquery-ui.min.js?v=5.2.0 HTTP/1.1" 200 68358 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0"
phpmyadmin | 172.25.0.1 - [30/Dec/2022:00:20:51 +0000] "GET /js/dist/cross_framing_protection.js?v=5.2.0 HTTP/1.1" 200 613 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0"
phpmyadmin | 172.25.0.1 - [30/Dec/2022:00:20:51 +0000] "GET /js/dist/functions.js?v=5.2.0 HTTP/1.1" 200 37759 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0"
phpmyadmin | 172.25.0.1 - [30/Dec/2022:00:20:51 +0000] "GET /js/messages.php?l=en&v=5.2.0 HTTP/1.1" 200 8199 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0"
phpmyadmin | 172.25.0.1 - [30/Dec/2022:00:20:51 +0000] "GET /js/dist/codemirror/addon/lint/sql-lint.js?v=5.2.0 HTTP/1.1" 200 800 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0"
phpmyadmin | 172.25.0.1 - [30/Dec/2022:00:20:51 +0000] "GET /js/vendor/codemirror/lib/codemirror.js?v=5.2.0 HTTP/1.1" 200 24680 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0"
```

Спробуємо зайти під паролем mypassword. Логін – root:



На сайті можемо бачити інформацію про базу даних, власне самі таблиці, та ще багато інформації.

В окремій вкладці терміналу перевіримо, чи працюють наші контейнери:

```
alex@debian:~/Documents/Lab9-1/mage$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS
4bdeab6cf645   nginx:latest   "/docker-entrypoint..." 22 minutes ago Up 21 minutes 0.0.0.0:80->80/tcp,
7f3c9c14850c   mage-php       "docker-php-entrypoi..." 22 minutes ago Up 21 minutes 9000/tcp
e49736876285   phpmyadmin/phpmyadmin "/docker-entrypoint..." 22 minutes ago Up 21 minutes 0.0.0.0:8090->80/tcp
41b3aa8bf849   mysql:latest   "docker-entrypoint.s..." 22 minutes ago Up 21 minutes 0.0.0.0:3306->3306/t
alex@debian:~/Documents/Lab9-1/mage$
```

Зручність docker-compose полягає в тому, що загальні налаштування всіх сервісів відбуваються в одному файлі, контейнери «піднімаються» також разом, і зупиняти їх треба не окремо, а разом однією командою.

Робота виконана, сервер працює. Зупиняємо командою docker-compose stop:

```
alex@debian:~/Documents/Lab9-1/mage$ docker-compose stop
[+] Running 4/4
  :: Container nginx          Stopped                  15.0s
  :: Container phpmyadmin     Stopped                  14.9s
  :: Container php-fpm        Stopped                  6.0s
  :: Container mysql          Stopped                  7.5s
alex@debian:~/Documents/Lab9-1/mage$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS        NAMES
alex@debian:~/Documents/Lab9-1/mage$
```

Контейнери зупинені. Завдання виконане.

Робота була виконана на останній версії дистрибутиву Debian – 11 (Bullseye). Характеристики системи, виведені за допомогою утиліти neofetch:

```
alex@debian
-----
OS: Debian GNU/Linux 11 (bullseye) x86_64
Host: VirtualBox 1.2
Kernel: 5.10.0-20-amd64
Uptime: 7 hours, 1 min
Packages: 2065 (dpkg)
Shell: bash 5.1.4
Resolution: preferred
DE: GNOME 3.38.6
WM: Mutter
WM Theme: Adwaita
Theme: Green-Submarine [GTK2/3]
Icons: Adwaita [GTK2/3]
Terminal: gnome-terminal
CPU: Intel i3-8130U (2) @ 2.207GHz
GPU: 00:02.0 VMware SVGA II Adapter
Memory: 640MiB / 1982MiB
```



Контрольні запитання:

1) Що таке Docker Compose?

Docker Compose – це утиліта, яка полегшує збірку і запуск системи, що складається з декількох контейнерів, пов'язаних між собою. Утиліта спрощує організацію процесів контейнерів Docker, включаючи запуск, зупинку і налаштування зв'язків і томів всередині контейнера.

2) Що таке Dockerfile?

Це скрипт, який дозволяє автоматизувати процес побудови контейнерів шляхом виконання відповідних команд (дій) в базовому образі (base) для формування нового образу.

3) Які вам відомі команди для роботи з Dockerfile?

FROM – основа образу, RUN – запустити команду, тощо

4) У чому полягає алгоритм створення проекту для розроблення web застосування?

Алгоритм полягає у компонуванні всіх сервісів (мікросервісів) між собою. Якщо це робити без Docker – то все це з великою ймовірністю не буде працювати на іншій машині, тому що, можливо, не будуть встановлені залежності, або ще з якихось причин. Docker ізолює всі сервіси по контейнерам, і тому вони будуть працювати будь-де. Але, ці контейнери потрібно пов'язати між собою, з цією задачею справляється програма Docker Compose.

Висновок: за результатами виконання цієї лабораторної роботи було ознайомлено з процесом написання скриптів Dockerfile, створенням контейнерів, компонуванням за допомогою Docker Compose. Також, був створений веб-додаток за допомогою Nginx, PHP-fpm, Magento2 та база даних MySQL.

Додаткові джерела:

- 1) [MongoDB.com – MongoDB documentation](https://www.mongodb.com/docs/)
- 2) [Linux.die.net – wget utility documentation](https://linux.die.net/~wget/)
- 3) [Jira.MongoDB.com – Is repo down permanently?](https://jira.mongodb.com/browse/IS-1000)
- 4) [StackOverflow – apt-utils problems](https://stackoverflow.com/questions/14223860/apt-utils-problems)
- 5) [GitHub.com – Magento 2 repository](https://github.com/magento/magento2)
- 6) [PHP Composer – official website](https://getcomposer.org/)
- 7) [StackOverflow – problems with PHP extensions](https://stackoverflow.com/questions/14223860/apt-utils-problems)