

Опис завдання:

- Зареєструватись на Docker Hub
- Створити не оптимізовану версію власного іміджу та завантажити його на Docker Hub з тегом версії 0.1
- Оптимізувати створений імідж та завантажити його на Docker Hub з тегом версії 0.2

Результат завдання:

- завантажити на сервер в окрему теку зі своїм ім'ям Dockerfile оптимізований та ні
- завантажити текстовий файл з описом виконаних кроків
- вказати адресу власного репозиторію на Docker Hub, перевірити що з нього можна завантажити обидва іміджи та створити з них працюючи контейнери (вказати стеги для цього)

План

1. Створення власного Docker образу

- 1.1 Створення робочої директорії
- 1.2 Створення Dockerfile
- 1.3 Створення образу
- 1.4 Перевірка створеного образу
- 1.5 Запуск контейнера з власного образу
- 1.6 Перевірка роботи сервера у браузері
- 1.7 Завантаження створеного іміджу на DockerHub

2. Оптимізація Dockerfile

- 2.1 Використання мінімальних базових образів
- 2.2 Об'єднання команд RUN
- 2.3 Використання `**.dockerignore**`
- 2.4 Створення образу з оптимізованого Dockerfile
- 2.5 Перевірка створеного образу
- 2.6 Запуск контейнера з власного образу
- 2.7 Перевірка роботи сервера у браузері
- 2.8 Завантаження створеного іміджу на DockerHub

Попередньо маємо налаштовану VM з встановленим Docker, та прописані у файлі hosts IP адреси нашої VM. Також для цього завдання реєструємося на DockerHub

1. Створення власного Docker образу

1.1. Створення робочої директорії

Запускаємо VM, логінімося, запускаємо термінал, створюємо теку my-docker-image та заходимо в неї

Команда:

```
mkdir my-docker-image
```

Відгук:

...

Команда:

```
cd my-docker-image
```

Відгук:

...

1.2 Створення Dockerfile

Команда:

```
nano Dockerfile
```

Відгук:

...

Наповнюємо файл вмістом згідно завдання:

```
FROM ubuntu:latest
RUN apt update
RUN apt install -y nginx
COPY index.html /var/www/html/
CMD ["nginx", "-g", "daemon off;"]
```

Зберігаємо наш файл зі змінами (Cntr+O, Cntr+X)

Додатково створимо й файл index.html

Команда:

```
nano index.html
```

Відгук:

...

Наповнюємо файл вмістом:

```
<html>
  <head>
    <title> lesson 4 </title>
  </head>
  <body>
    <h1> WOW IT WORKS! </h1>
  </body>
</html>
```

Зберігаємо наш файл зі змінами (Cntr+O, Cntr+X)

1.3 Створення образу

Команда:

```
sudo docker build -t my-nginx-image .
```

Відгук:

```
[+] Building 11.8s (9/9) FINISHED
docker:default
=> [internal] load build definition from Dockerfile
0.0s
=> => transferring dockerfile: 162B
0.0s
=> [internal] load metadata for docker.io/library/ubuntu:latest
1.4s
=> [internal] load .dockerignore
0.0s
=> => transferring context: 2B
0.0s
=> CACHED [1/4] FROM
docker.io/library/ubuntu:latest@sha256:72297848456d5d37d1262630108
ab308d3e9ec7ed1c3286a32fe09856619a782
0.0s
=> [internal] load build context
0.0s
=> => transferring context: 141B
0.0s
=> [2/4] RUN apt update
6.0s
=> [3/4] RUN apt install -y nginx
4.1s
=> [4/4] COPY index.html /var/www/html/
0.1s
=> exporting to image
0.2s
=> => exporting layers
0.2s
=> => writing image
sha256:c5d7b98f95ac4566fccf5f43fd1599967c771badeab9f2292e286560ae8
154d6
0.0s
=> => naming to docker.io/library/my-nginx-image
```

1.4 Перевірка створеного образу

Команда:

```
sudo docker images
```

Відгук:

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
my-nginx-image	latest	c5d7b98f95ac	4 minutes ago	132MB

Бачимо що наш імідж створився. Тепер треба перевірити його на працездатність

1.5 Запуск контейнера з власного образу

Команда:

```
sudo docker run -d -p 8080:80 my-nginx-image
```

Відгук:

```
97517a350a4407ec716afd456e8b47d9cceed3d9680c08d4aaa092e8f2eef007
```

Команда:

```
sudo docker ps
```

Відгук:

CONTAINER ID	IMAGE	COMMAND	CREATED
STATUS	PORTS		NAMES
88882a034966	my-nginx-image	"nginx -g 'daemon of..."	5 minutes ago
Up 5 minutes	0.0.0.0:8080->80/tcp, [::]:8080->80/tcp		pensive_chandrasekhar

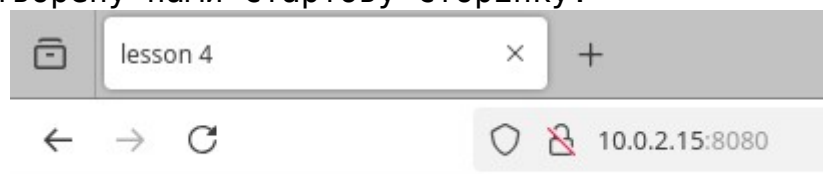
Бачимо що контейнер стартував і працює на портах 8080->80, тепер перевіримо чи добре працює nginx

2.7 Перевірка роботи сервера у браузері

Запускаємо браузер і вводимо адресу:

<http://10.0.2.15:8080>

Отримуємо створену нами стартову сторінку:



WOW IT WORKS!

Для текстової версії домашньої роботи, де картинки не буде, робимо додатковий тест з curl

Команда:

```
curl http://10.0.2.15:8080
```

Відгук:

```
<html>
<head>
  <title> lesson 4 </title>
</head>
<body>
  <h1> WOW IT WORKS! </h1>
</body>
```

</html>

Все працює чудово))

1.7 Завантаження створеного іміджу на DockerHub

Попередньо маємо реєстрацію на DockerHub. Тож авторизуємося

Команда:

```
sudo docker login -u 4l3xb0l0t08
```

Відгук:

Password:

WARNING! Your password will be stored unencrypted in
/root/.docker/config.json.

Configure a credential helper to remove this warning. See
[https://docs.docker.com/engine/reference/commandline/login/
#credential-stores](https://docs.docker.com/engine/reference/commandline/login/#credential-stores)

Login Succeeded

Тегуємо наш образ перед завантаженням на DockerHub.

Ця дія дозволяє зберегти версійність

Команда:

```
sudo docker tag my-nginx-image 4l3xb0l0t08/my-nginx-image:0.1
```

Відгук:

...

Тепер завантажуюмо наш імідж на DockerHub

Команда:

```
sudo docker push 4l3xb0l0t08/my-nginx-image:0.1
```

Відгук:

The push refers to repository [docker.io/4l3xb0l0t08/my-nginx-
image]

c015a105b54e: Pushed

e43fe86d8cb7: Pushed

35ea701590c1: Pushed

4b7c01ed0534: Pushed

0.1: digest:

sha256:83115f219940d4f16ba7bfa827f643cb993f6639ec3c3bdce12061f6093

ece96 size: 1159

Імідж завантажився і ми бачимо його у себе в акаунті на DockerHub
з тегом 0.1

4l3xb0l0t08/my-nginx-image 🌐

Last pushed 2 minutes ago • Repository size: 63 MB

Add a description ✎ ⓘ

Add a category ✎ ⓘ

General Tags Builds Collaborators Webhooks Settings

Tags

This repository contains 1 tag(s).

Tag	OS	Type	Pulled	Pushed
0.1		Image	less than 1 day	6 minutes

[See all](#)

PDF версія домашньої роботи містить всі зображення які демонструють результат. Текстова версія зображень не має

Підсумок першої частини:

Перша частина домашньої роботи завершена, імідж був створений, перевірений і завантажений на DockerHub

2. Оптимізація Dockerfile

2.1 Використання мінімальних базових образів

Для оптимізації використовуємо мінімалістичний імідж Alpine Linux який вже має інстальований Nginx

2.2 Об'єднання команд RUN

Змінюємо наш Dockerfile

Команда:

nano Dockerfile

Відгук:

...

Оптимізуємо файл згідно завдання:

```
FROM nginx:alpine3.21
RUN apk update && apk upgrade && rm -rf /var/cache/apk/*
COPY index.html /usr/share/nginx/html/
CMD ["nginx", "-g", "daemon off;"]
```

Зберігаємо наш файл зі змінами (Cntr+O, Cntr+X)

!!!Треба зауважити, що в нашому випадку, nginx та інші необхідні для його роботи пакети, вже інстальовано в імідж з якого ми будуємо свій. Того інструкцію RUN по великому рахунку можна було б і не включати до файлу, але для цього завдання (щоб продемонструвати дію) ми об'єднуємо декілька інструкцій RUN в одну

2.3 Використання .dockerignore

Додатково створюємо файл .dockerignore який допомагає виключити деякі непотрібні в іміджу файли при його створенні

Команда:

```
nano .dockerignore
```

Відгук:

...

Заповнюємо файл даними:

Dockerfile

.git

*.log

Зберігаємо наш файл зі змінами (Cntr+O, Cntr+X)

2.4 Створення образу з оптимізованого Dockerfile

Команда:

```
sudo docker build -t my-nginx-image .
```

Відгук:

```
[+] Building 4.1s (9/9) FINISHED
```

```
docker:default
```

```
=> [internal] load build definition from Dockerfile
```

```
0.0s
```

```
=> => transferring dockerfile: 192B
```

```
0.0s
```

```
=> [internal] load metadata for
```

```
docker.io/library/nginx:alpine3.21
```

```
1.1s
```

```
=> [auth] library/nginx:pull token for registry-1.docker.io
```

```
0.0s
```

```
=> [internal] load .dockerignore
```

```
0.0s
```

```
=> => transferring context: 62B
```

```
0.0s
```

```
=> CACHED [1/3] FROM
```

```
docker.io/library/nginx:alpine3.21@sha256:b471bb609adc83f73c2d95148cf1bd683408739a3c09c0afc666ea2af0037aef
```

```
0.0s
```

```
=> [internal] load build context
```

```
0.0s
```

```
=> => transferring context: 31B
```

```
0.0s
```

```

=> [2/3] RUN apk update && apk upgrade && rm -rf /var/cache/apk/*
2.9s
=> [3/3] COPY index.html /usr/share/nginx/html/
0.0s
=> exporting to image
0.1s
=> => exporting layers
0.1s
=> => writing image
sha256:790a7f7a2b7d69d994b5c1e2ca604f6fa9d50bad1b94195a8f896c71ffc
8f029
0.0s
=> => naming to docker.io/library/my-nginx-image

```

2.5 Перевірка створеного образу

Перевіряємо чи з'явився у нас новий імідж і якого він розміру

Команда:

```
sudo docker images
```

Відгук:

REPOSITORY	TAG	IMAGE ID	CREATED
SIZE			
my-nginx-image	latest	790a7f7a2b7d	About a minute ago
57MB			

Бачимо що розмір іміджу значно зменшився, оптимізація була вдала
Тепер нам треба протестувати створений імідж

2.6 Запуск контейнера з власного образу

Команда:

```
sudo docker run -d -p 8080:80 my-nginx-image
```

Відгук:

```
5417aee6a1ca7aae647539cb3cfe9da732dff904440eab53e35dc1dc0f4d9c83
```

Команда:

```
sudo docker ps -a
```

Відгук:

CONTAINER ID	IMAGE	COMMAND	CREATED
STATUS		PORTS	
NAMES			
5417aee6a1ca	my-nginx-image	"/docker-entrypoint..."	1 minutes ago
	Up 1 minutes	0.0.0.0:8080->80/tcp,	
[::]:8080->80/tcp	sweet_greider		

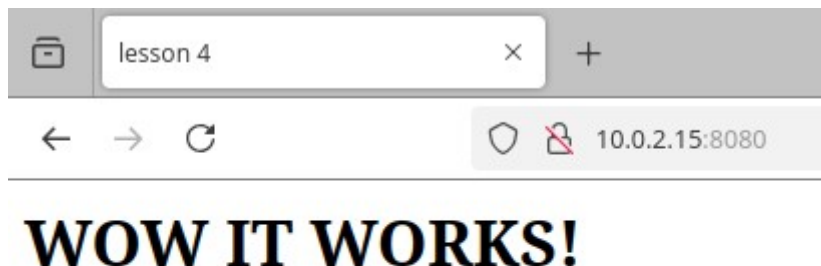
Бачимо що контейнер стартував, працює на відповідних портах і
тепер перевіримо чи добре працює nginx

2.7 Перевірка роботи nginx сервера у браузері

Запускаємо браузер і вводимо адресу:

<http://10.0.2.15:8080>

Отримуємо створену нами стартову сторінку:



Для текстової версії домашньої роботи, де картинки не буде, робимо додатковий тест з curl

Команда:

```
curl http://10.0.2.15:8080
```

Відгук:

```
<html>
  <head>
    <title> lesson 4 </title>
  </head>
  <body>
    <h1> WOW IT WORKS! </h1>
  </body>
</html>
```

Контейнер з оптимізованої версії нашого іміджу працює чудово

2.8 Завантаження створеного іміджу на DockerHub

Попередньо маємо реєстрацію на DockerHub. Тож авторизуємося

Команда:

```
sudo docker login -u 4l3xb0l0t08
```

Відгук:

Password:

WARNING! Your password will be stored unencrypted in
/root/.docker/config.json.

Configure a credential helper to remove this warning. See
<https://docs.docker.com/engine/reference/commandline/login/#credential-stores>

Login Succeeded

Тегуємо наш образ перед завантаженням на DockerHub.
Ця дія дозволяє зберегти версійність

Команда:

```
sudo docker tag my-nginx-image 4l3xb0l0t08/my-nginx-image:0.2
```

Відгук:

...

Тепер завантажуюмо наш імідж на DockerHub

Команда:

```
sudo docker push 4l3xb0l0t08/my-nginx-image:0.2
```

Відгук:

The push refers to repository [docker.io/4l3xb0l0t08/my-nginx-image]

3e3e42bda25d: Pushed

9b00ee496d02: Pushed

72120687062c: Mounted from library/nginx

469fc702bc62: Mounted from library/nginx

74964efcae21: Mounted from library/nginx

ad4f5bc987ca: Mounted from library/nginx

ef050c9a03b5: Mounted from library/nginx

83c20bc61eb8: Mounted from library/nginx

1024e8977b69: Mounted from library/nginx

a0904247e36a: Mounted from library/nginx

0.2: digest:

sha256:d9892843b539597c9ce8001bced547d41e963448d0cae6399ed9efaef3c3d53c size: 2407

Імідж завантажився і ми бачимо його у себе в акаунті на DockerHub з тегом 0.2

4l3xb0l0t08/my-nginx-image 🐳

Last pushed 3 minutes ago • Repository size: 63 MB



Add a description ✎ ⓘ

Add a category ✎ ⓘ

General Tags Builds Collaborators Webhooks Settings

Tags

This repository contains 2 tag(s).

Tag	OS	Type	Pulled	Pushed
0.2		Image	less than 1 day	3 minutes
0.1		Image	less than 1 day	about 19 hours

[See all](#)

Підсумок другої частини:

Друга частина домашньої роботи завершена, Dockerfile був змінений і оптимізований, додано файл .dockerignore який дозволяє виключити деякі непотрібні файли з іміджу який створюється. Створений імідж був перевірений і завантажений на DockerHub

Перевірка на працездатність завантажених на DockerHub образів

1. Не оптимізований образ

Завантажуємо свій не оптимізований образ з DockerHub

Команда:

```
sudo docker pull 4l3xb0l0t08/my-nginx-image:0.1
```

Відгук:

0.1: Pulling from 4l3xb0l0t08/my-nginx-image

107a4fb0af38: Already exists

127790638414: Already exists

d5cfa168ce3f: Already exists

8325818d709e: Already exists

Digest:

sha256:83115f219940d4f16ba7bfa827f643cb993f6639ec3c3bdce12061f6093e96

Status: Downloaded newer image for 4l3xb0l0t08/my-nginx-image:0.1

docker.io/4l3xb0l0t08/my-nginx-image:0.1

Перевіряємо чи з'явився у нас новий імідж

Команда:

```
sudo docker images
```

Відгук:

REPOSITORY	TAG	IMAGE ID	CREATED
SIZE			
4l3xb0l0t08/my-nginx-image	0.1	c5d7b98f95ac	24 hours ago
132MB			

З'явився все ок. Тепер перевірмо його на працездатність, запустимо контейнер

Команда:

```
sudo docker run -d -p 8080:80 4l3xb0l0t08/my-nginx-image:0.1
```

Відгук:

07c5243fbd78440b4cff98c8bb157181e9e9c9fb1b1b288b5148a0d03d947884

Контейнер стартував, перевірмо чи добре працює nginx

Команда:

```
sudo docker ps
```

Відгук:

CONTAINER ID	IMAGE	COMMAND
CREATED	STATUS	PORTS
NAMES		
be4bd66c8ff9	4l3xb0l0t08/my-nginx-image:0.1	"nginx -g 'daemon of...'"
2 minutes ago	Up 2 minutes	0.0.0.0:8080->80/tcp, [::]:8080->80/tcp
	mystifying_cray	

Контейнер працює чудово і використовує порти 8080->80
Тепер перевіримо його командою curl

Команда:

```
curl http://10.0.2.15:8080
```

Відгук:

```
<html>
<head>
  <title> lesson 4 </title>
</head>
<body>
  <h1> WOW IT WORKS! </h1>
</body>
</html>
```

Версія нашого іміджу 0.1 – повністю працездатна. Успіх))

Зробимо те саме з версією 0.2

2. Оптимізований образ

Завантажуємо свій оптимізований образ з DockerHub

Команда:

```
sudo docker pull 4l3xb0l0t08/my-nginx-image:0.2
```

Відгук:

```
0.2: Pulling from 4l3xb0l0t08/my-nginx-image
1f3e46996e29: Already exists
5215a08fb124: Already exists
f8813b38090d: Already exists
9f41882e104d: Already exists
e92b9802c411: Already exists
4b56e0e1b50d: Already exists
5281c445f8b7: Already exists
a53100808f89: Already exists
b2ff714b7b98: Already exists
0f18f7ac80c6: Already exists
Digest:
sha256:d9892843b539597c9ce8001bced547d41e963448d0cae6399ed9efaef3c3d53c
Status: Downloaded newer image for 4l3xb0l0t08/my-nginx-image:0.2
```

docker.io/4l3xb0l0t08/my-nginx-image:0.2

Перевіряємо чи з'явився цей імідж у нас

Команда:

```
sudo docker images
```

Відгук:

REPOSITORY	TAG	IMAGE ID	CREATED
SIZE			
4l3xb0l0t08/my-nginx-image	0.2	790a7f7a2b7d	5 hours ago
57MB			
4l3xb0l0t08/my-nginx-image	0.1	c5d7b98f95ac	24 hours ago
132MB			

З'явився все ок. Бачимо, що завдяки тегам ми можемо відрізняти ці іміджи. Також бачимо, що в них різний розмір. Тепер перевірмо версію 0.2 на працездатність, запустимо контейнер

Команда:

```
sudo docker run -d -p 8080:80 4l3xb0l0t08/my-nginx-image:0.2
```

Відгук:

```
af0faef8b5122c1a82a190c83a716b6b297e51c510ff8851d448e972a51925d5
```

Контейнер стартував, подивимося на процеси, щоб впевнитися, що все дійсно ок

Команда:

```
sudo docker ps
```

Відгук:

CONTAINER ID	IMAGE	COMMAND
CREATED	STATUS	PORTS
NAMES		
af0faef8b512	4l3xb0l0t08/my-nginx-image:0.2	"/docker-entrypoint...."
	2 minutes ago	Up 2 minutes
	0.0.0.0:8080->80/tcp, [::]:8080->80/tcp	unruffled_tharp

Все ок, бачимо контейнер в процесах, він працює на своїх портах. Тепер можна і командою curl перевірити, чи віддає він нам той index файл по замовченню

Команда:

```
curl http://10.0.2.15:8080
```

Відгук:

```
<html>
  <head>
    <title> lesson 4 </title>
  </head>
  <body>
    <h1> WOW IT WORKS! </h1>
  </body>
</html>
```

Наш оптимізований імідж 0.2 – повністю працездатний. Це успіх))

Домашнє завдання до уроку 4:

- Зареєструватись на Docker Hub
- Створити не оптимізовану версію власного імаджу та завантажити його на Docker Hub з тегом версії 0.1
- Оптимізувати створений імадж та завантажити його на Docker Hub з тегом версії 0.2

Студент: Олександр Болотов

Дата виконання завдання: 14.02.2025