Docker

Duong Pham

Viettel Digital Talent 2024 - Software & Data Engineering

Lesson Outline

1. Docker under the hood
2. Practical Docker

- hello-world container
- ubuntu container
- nginx container
- python container
- mysql container
- nodejs web image & container
  - Create a Dockerfile
  - Build a docker image and push to Dockerhub
  - Docker compose

Hello World container

```
docker container run hello-world
```

```
docker container ls -a
```

```
docker image ls
```

Ubuntu container

```
docker container run ubuntu
```

```
docker ps -a (or docker container ls -a)
docker image ls
```

Ubuntu container

```
docker run -it ubuntu
```

```
docker ps
```

Nginx container

Nginx [engine x] is an open source HTTP web server for web serving, load balancing …

```
docker run nginx
```

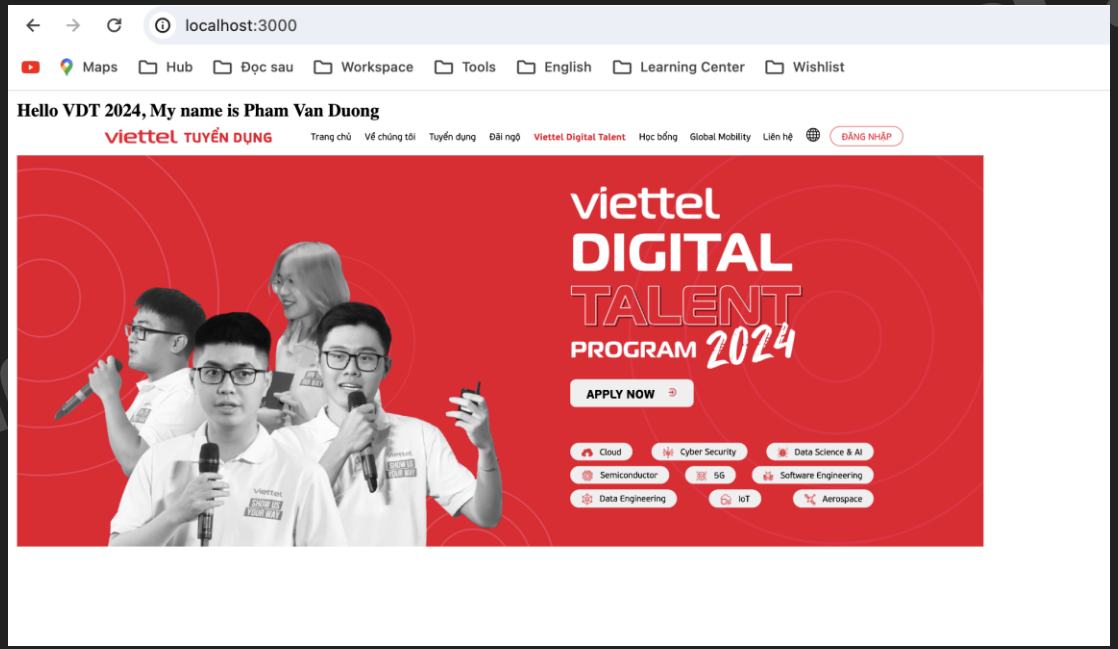
Nginx container

create a custom html page index.html

```
docker run -p 8080:80 -v $PWD:/usr/share/nginx/html nginx
```

Exercise #1

Create a web server using Nginx showing a headline text “Hello VDT 2024, My name is <your-name>” and a logo image, mapping at port 3000



Python container

```
docker run -it python
```

Inside container terminal:

```
print("Hello, world");
1+1
import calendar;
print(calendar.month(2024, 5));
```

Python container

create a python file hello.py

```
print("Hello, world")
```

```
docker run -v $PWD:/app python python3 app/hello.py
```

Exercise #2

Create an python application that take year and month input from user and print out the calendar. (hint: use calendar lib)

```
➜  container-series docker run -it -v $PWD:/app python python3 app/my-calendar.py
Enter the year: 2024
Enter the month: 5
      May 2024
Mo Tu We Th Fr Sa Su
       1  2  3  4  5
 6  7  8  9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30 31
```

MySql container

```
docker run --name mysql -d \
                -p 3306:3306 \
            -e MYSQL_ROOT_PASSWORD=viettel \
            mysql:8
```

Read the log:

```
docker logs mysql --follow
```

MySql container

Access the database in container:

```
docker exec -it mysql mysql -p
```

```
mysql> CREATE DATABASE my_db;
mysql> USE my_db;
mysql> CREATE TABLE Persons (
    PersonID int,
    Name varchar(255)
);
mysql> INSERT INTO Persons VALUES (1, ”duong");
mysql> SELECT * FROM Persons;
```

MySql container

```
Persist data in database

docker stop mysql

docker rm mysql

docker run --name mysql -d \

    -p 3306:3306 \

    -e MYSQL_ROOT_PASSWORD=viettel \

    -v mysql:/var/lib/mysql \

    mysql:8
```

MySql container

Insert some data:

```
mysql> CREATE DATABASE my_db;
mysql> USE my_db;
mysql> CREATE TABLE Persons (
    PersonID int,
    Name varchar(255)
);
mysql> INSERT INTO Persons VALUES (1, ”duong");
mysql> SELECT * FROM Persons;
```

MySql container

Remove and recreate the container and see data is still there

```
docker stop mysql
docker rm mysql

docker run --name mysql -d \
    -p 3306:3306 \
    -e MYSQL_ROOT_PASSWORD=viettel \
    -v mysql:/var/lib/mysql \
    mysql:8

docker exec -it mysql mysql -p
mysql> show databases;
```

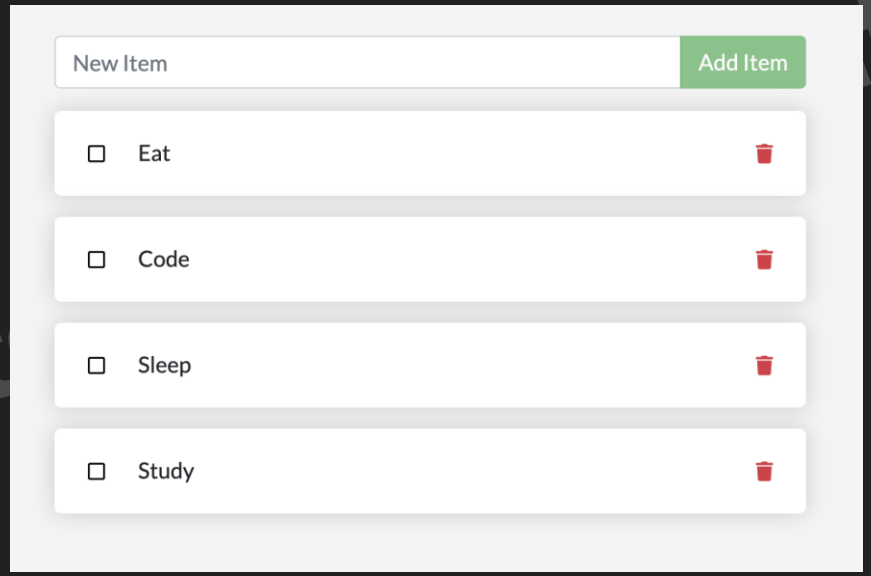
Docker Scout

Container images consist of layers and software packages, which are susceptible to vulnerabilities. These vulnerabilities can compromise the security of containers and applications.
Docker Scout is a solution for proactively enhancing your software supply chain security.

```
docker scout quickview [IMAGE]
```

Nodejs app container

- Download source code here, unzip and cd to app folder.



Exercise #3

Create a Dockerfile for todo app, build the image and run it.
[Optional] Push image to dockerhub.

Hint:

- the todo app run at port 3000
- it is a nodejs app so we need to run `npm install` to install all dependencies.
- the command to run the app is `node src/index.js`

Exercise #4: Persist data of our todo app

By default, the todo app stores its data in a SQLite Database at /etc/todos/todo.db. SQLite is a relational database in which all of the data is stored in a single file.

Your task is to persist our todo app's data.

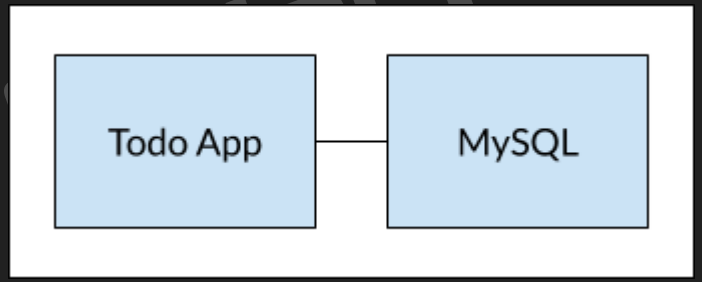
How to test:

- start the container and add few items.
- remove the container: `docker stop <container-name>;`

  `docker rm <container-name>;`
- confirm that the added items are still there.

Multi-Container App

SQLite stores data in a single file, which is not best for production. Let’s introduce a separated database MySQL.



Docker Network

Containers, by default, run in isolation and don't know anything about other processes or containers on the same machine.

Create the network: `docker network create todo-app`
Start MySQL database in the network:

```
docker run -d \
    --network todo-app --network-alias mysql \
    -v todo-mysql-data:/var/lib/mysql \
    -e MYSQL_ROOT_PASSWORD=secret \
    -e MYSQL_DATABASE=todos \
    mysql:8.0
```

Confirm database was created:

```
docker exec -it <mysql-container-id> mysql -p
```

```
mysql> SHOW DATABASES;
```

Run our todo app with MySQL database

The todo app supports the setting of a few environment variables to specify MySQL connection settings:

MYSQL_HOST - the hostname for the running MySQL server

MYSQL_USER - the username to use for the connection

MYSQL_PASSWORD - the password to use for the connection

MYSQL_DB - the database to use once connected

Exercise #5

Modify the Dockerfile to for our app to work with MySQL.

How to test:

- after running both MySql and Todo app container:
  - check the app logs by: `docker logs todo --follow`

```
➜  app docker logs todo --follow
Waiting for mysql:3306.
Connected!
Connected to mysql db at host mysql
Listening on port 3000
```

  - add a few items to the todo list
  - query the database:
    - `docker exec -it <mysql-container-id> mysql -p`
    - `mysql> use todos;`
    - `mysql> select * from todo_items;`

Exercise #6: Using docker compose

Use docker compose to define todo app and MySQL database.

Create a docker-compose.yml file.

```
docker compose up -d
```