VIET NAM NATIONAL UNIVERSITY HO CHI MINH CITY

UNIVERSITY OF INFORMATION TECHNOLOGY

-----○○🕮○○-----

PROJECT REPORT

Network and System Management

Deploy Web Application using Docker

LECTURER: MS. TRAN THI DUNG

NT132.N12.ATCL – EN

GROUP 08:

DOAN NGUYEN DANG KHOA - 20521463

NGUYEN VIET HOANG - 20520189

NGUYEN DUC TRUNG - 20520956

Ho Chi Minh City, January 6th, 2023

Table of Contents

I. INTRODUCTION

1.1 OVERVIEW

a. Definition

Docker is an open platform that allows developers to build, deploy, run, update and manage containers. Docker provides us the ability to package and run an application in a loosely isolated environment called containers.

b. How does Docker work?

Docker works by executing code using standard methods. It's like a virtual machine that virtualizes user's server hardware so the user don't have to manage it directly. At this point, the container virtualizes the host's operating system. Once Docker is installed on each host, it provides basic commands that can be used to create, initialize, or stop containers.

1.2 COMPONENTS

- **Docker**

  - Docker file

  - Docker image

  - Docker compose

- **Web application**

  - Website

  - Database

1.3 OPERATION

a. Workflow

Here is the typical *Docker* workflow:

1. Find an image on Docker Hub.
2. Pull an image from Docker Hub.
3. Run an image pulled on Docker host.
4. Stop an instance (container).
5. Remove an instance (container).
6. Remove an image.

Docker Hub is a place for people to share and store pre-built images, just like github, it's also use to manging versions of the image.

Here's the command needed for "typical *Docker* workflow" (the order is corresponding to the workflow aforementioned):

1. Go to Docker Hub, find the desired image to run. Ex: ubuntu version 20.04
2. $ docker pull [OPTIONS] NAME[:TAG|@DIGEST]. Details.
   Ex: $ docker pull ubuntu:20.04
3. $ docker run [OPTIONS] IMAGE [COMMAND] [ARG...]. Details.
   Ex: $ docker run -ti --rm ubuntu /bin/bash
4. $ docker stop [OPTIONS] CONTAINER [CONTAINER...]. Details.
   Ex: $ docker stop 5e
5. $ docker rm [OPTIONS] CONTAINER [CONTAINER...]. Details.
   Ex: docker rm 5e
6. $ docker rmi [OPTIONS] IMAGE [IMAGE...]. Details.
   Ex: docker rmi ubuntu:20.04

b. How does Docker run an image?

A Docker image is a file used to execute code in a docker container. It act as a set of instructions to build a docker container.

A Docker container is a lightweight, standalone, executable package of software that includes everything needed to run an application.

To build an image, docker look for instructions in dockerfile. A dockerfile is a representation of an image, it has simple syntax, and is instructions, scripts for image creation. **It will run line by line from top to bottom.**

```
1  FROM python:3.10.8-alpine
2  COPY . /app
3  WORKDIR /app
4  RUN apk add gcc musl-dev python3-dev libffi-dev openssl-dev
5  RUN pip install -r requirements.txt
6  ENTRYPOINT ["python"]
7  CMD ["rest.py"]
```

Figure 1 Simple dockerfile

All instructions can be found here. Here is the breakdown of the "simple dockerfile".

The first line is ALWAYS a FROM instruction. It set the base image of the container. Usage: $ FROM <image>, for example: $ FROM ubuntu:20.04.

Followed by any other instructions that fit. Here some:

- $ COPY <src> <dest>: Copy “things” from <src> on local to <dest> on container.
- $ WORKDIR /path/to/dir: Set the working directory, like cd on unix.
- $ RUN <command>: Executes any commands specified while building the image.
- $ ENTRYPOINT <command> <param>: Configure container as an executable.
- $ CMD [<commands>]: Provide defaults for container, run commands.
  can only be ONE `CMD` in a single dockerfile.

RUN	CMD
Run command and commits the results at build	Does not execute anything at build.
Can have multiple RUN in a single dockerfile	Only one in a single dockerfile
	Provides defaults for ENTRYPOINT Is an intented command

c. How does docker compose work?

Why docker compose? To run multiple containers at once, to satisfies the need for isolation like database and website.

```yaml
version: "2.1"
services:
  app:
    build: ./app
    links:
      - db
    ports:
      - "80:5000"
    depends_on:
      db:
        condition: service_healthy
    restart: unless-stopped

  db:
    image: mysql:8.0.31
    ports:
      - "3306:3306"
    environment:
      MYSQL_ROOT_PASSWORD: root
      MYSQL_USER: db_user
      MYSQL_PASSWORD: Passw0rd
      MYSQL_DATABASE: employee_db
    volumes:
      - ./db:/docker-entrypoint-initdb.d/
    command: --default-authentication-plugin=caching_sha2_password
    healthcheck:
      test: [ "CMD", "mysqladmin", "ping", "-h", "localhost" ]
      timeout: 20s
      retries: 10
      interval: 5s
    restart: unless-stopped
```

Figure 2 simple docker-compose.yml

All instructions can be found here. Here is the breakdown of the “docker compose”.

The first line is the docker compose file version, is the entry at the root of the YAML file. Next one is the services that the docker is running, a compose file MUST declare a services as the element, is contains the configuration that is applied to each container. Here the docker compose file define two services: **app** and **db**

- build: The app service uses an image that’s built from the Dockerfile in the ./app directory. While the db uses a public mysql imaged pulled from Docker Hub registry
- links: defines a network link to containers in another service, here it link the app with the db services
- ports: It binds the container and the host machine to the exposed port, 80
- depends_on: Express startup and shutdown dependencies between services. Here it will depend on the db service, with the condition is service_healthy
- restart: defines the policy that the platform will apply on container termination. unless-stopped will restarts a container irrespective of the exit code but will stop restarting when the service is stopped or removed
- environment: defines environment variables set in the container. environment can use either an array or a map.
- The volumes key mounts the project directory (db directory) on the host to /docker-entry-point inside the container, allowing to modify the code on the fly, without having to rebuild the image.
- command overrides the default command declared by the container image
- healthcheck declares a check that’s run to determine whether or not containers for this service are “healthy”.

II. IMPLEMENTATION

2.1 TOPOLOGY

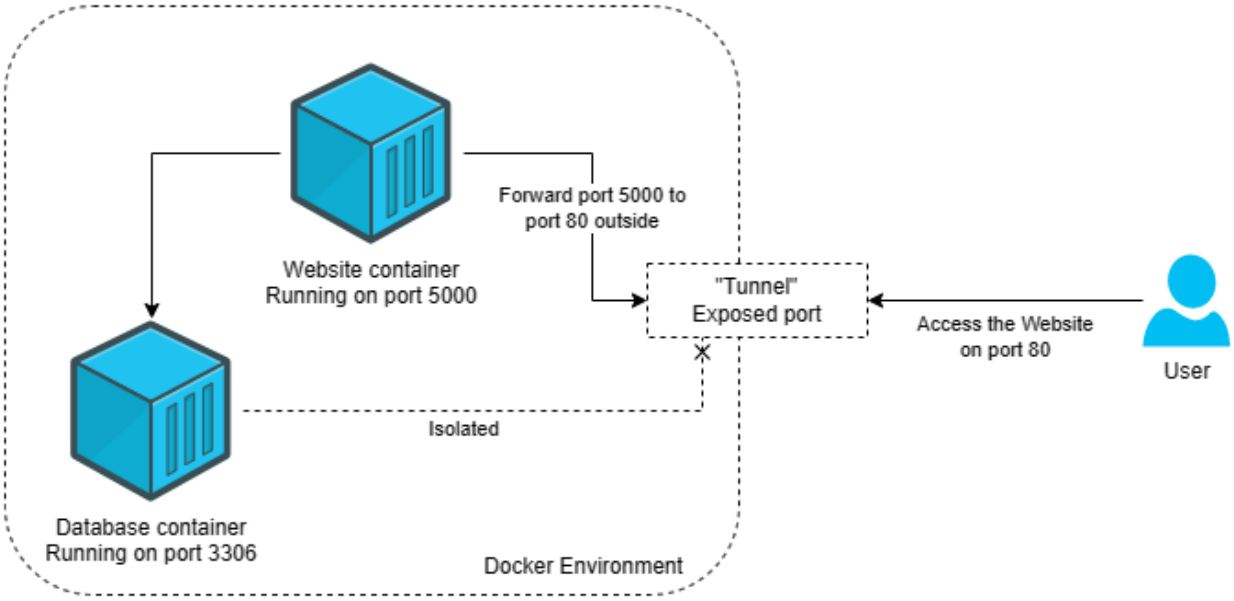


Figure 3 Topology

The database container doesn't need to forward the port "3306" to the outside but in this demo, we forward it to troubleshooting and debugging. Website in running on port 5000 inside docker environment and forwarded to 80 on the outside.We do this to illustrate the port forwarding of the docker compose.

Container	IP	"inside" Port	Exposed Port
Website	local	5000	80
Database	local	3306	3306 (optional)

2.2 INSTALLATION

In this demo, we run on a Windows host so in order to install Docker, WSL2 has to be installed to provide a backend for Docker to run.

a. Install WSL2

In our's setup, we just need to run this command terminal or command prompt: $ wsl --install, reboot and that's it.

b. Install Docker Desktop

Docker Desktop contain Docker it self and Docker Compose.

(1) Download Docker installer in this link.

(2) Run the installer with administrative privilege.

(3) Select options like in Fig. 4(a), click Ok.

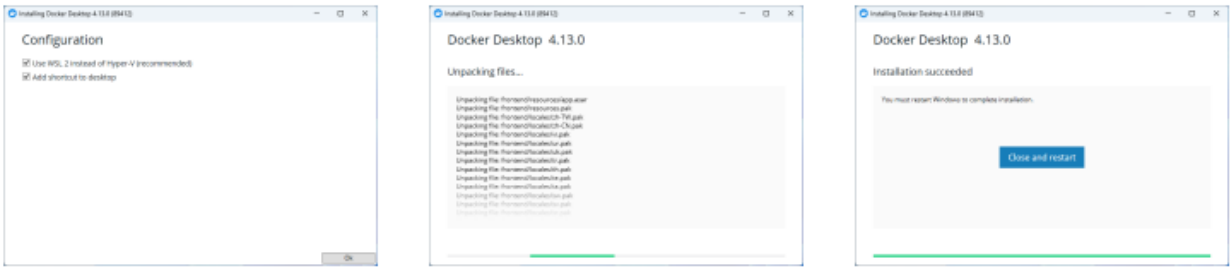
(4) Wait while installer install Docker on system Fig. 4(b).

(5) Click Close and restart in Fig.4(c) to finish installing Docker.

(6) Accept Docker Subscription Service Agreement on Fig. 5.

(7) Wait for Docker to start Fig. 6(a).

(8) Docker is installed Fig. 6



(a) Docker's installation menu (b) Docker is being installed (c) Setup completed

Figure 4 Docker installation steps

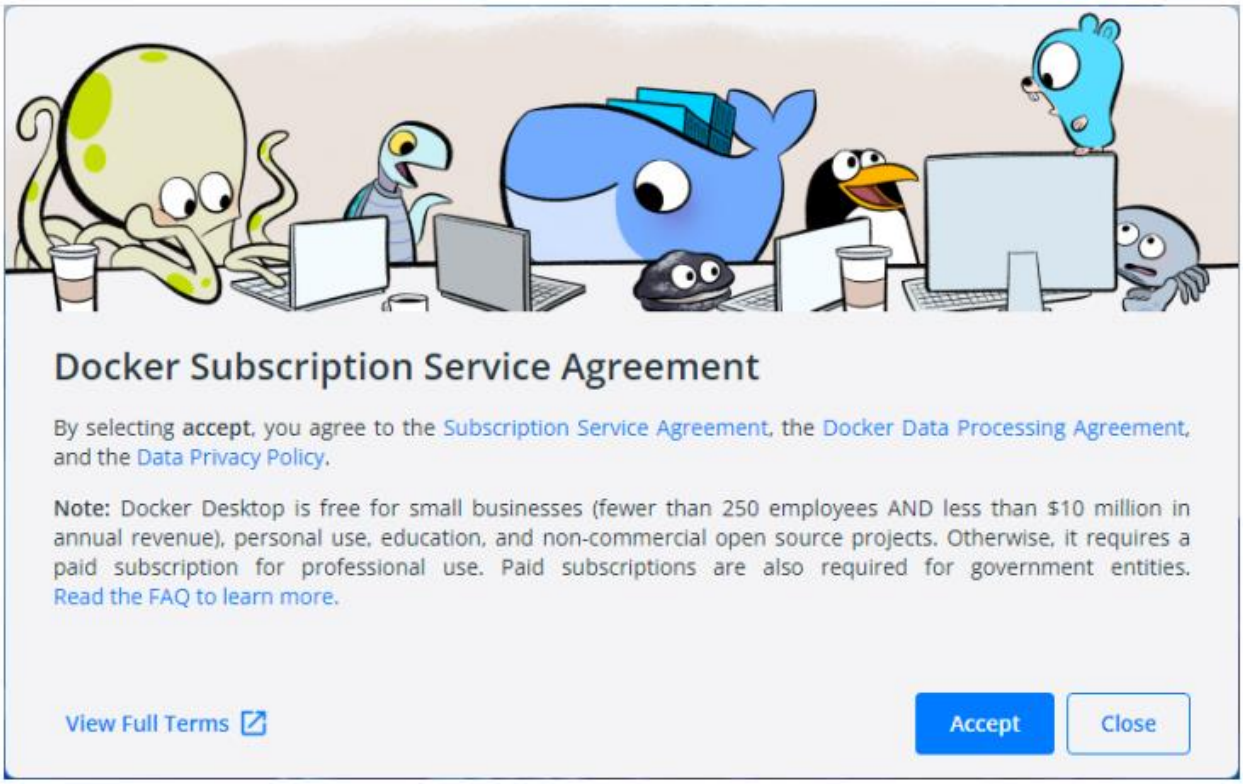
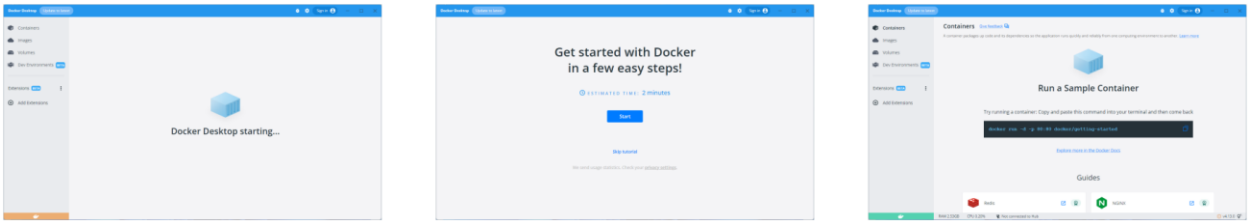


Figure 5 Docker Subscription Service Agreement



(a) Docker is starting (b) Docker get started screen, you can either start the tutorial or skip it (c) Docker welcome screen

Figure 6 Docker is installed

2.3 CONFIGURATION

Source code can be found here: [Khoadnd/webappdocker (github.com)]

Figure 7 Source code link

Step 0: Clone the source code.

$ git clone https://github.com/Khoadnd/webappdocker

- Simple database: (db/employee_db.sql)

```
CREATE DATABASE IF NOT EXISTS employee_db;
USE employee_db;

CREATE TABLE employees (name VARCHAR(20));
INSERT INTO employees VALUES ('KHOA'), ('HOANG'), ('TRUNG');

GRANT ALL ON *.* TO db_user@'%';
```

Figure 8 database

- Simple web application: (app/rest.py)

```
from app import app
from db import mysql

conn = mysql.connect()
cursor = conn.cursor()


@app.route("/")
def main():
    return "Welcome!"


@app.route('/hello')
def hello():
    return 'Hi!'


@app.route('/database')
def read():
    cursor.execute("SELECT * FROM employees")
    row = cursor.fetchone()
    result = []
    while row is not None:
        result.append(row[0])
        row = cursor.fetchone()

    return ",".join(result)


@app.route('/', defaults={'path': ''})
@app.route('/<path:path>')
def catch_all(path):
    return 'You lost? 😠'


if __name__ == "__main__":
    app.run(debug=True, host='0.0.0.0')
```

Figure 9 web application

Step 1: build the containers

$ docker-compose build

```
PS C:\Users\     \Desktop\webappdocker> docker-compose build
[+] Building 45.7s (10/10) FINISHED
 => [internal] load build definition from Dockerfile                                                0.0s
 => => transferring dockerfile: 231B                                                                0.0s
 => [internal] load .dockerignore                                                                   0.0s
 => => transferring context: 2B                                                                     0.0s
 => [internal] load metadata for docker.io/library/python:3.10.8-alpine                             4.7s
 => [internal] load build context                                                                   0.0s
 => => transferring context: 1.46kB                                                                 0.0s
 => [1/5] FROM docker.io/library/python:3.10.8-alpine@sha256:39c3cc0d0144eaecde476f470884aad1798366  9.9s
 => => resolve docker.io/library/python:3.10.8-alpine@sha256:39c3cc0d0144eaecde476f470884aad1798366  0.0s
 => => sha256:7ec3a91b66d9050989fca52618b7d400610c8c2cf9a8bbcb86bc403e0e01f220 622.90kB / 622.90kB  0.8s
 => => sha256:ebea74ce8637967595d874b851ae620c6a2b8e8cdd1a2075c307c788838c4859 12.39MB / 12.39MB    8.9s
 => => sha256:39c3cc0d0144eaecde476f470884aad17983664eb591e26ffcda55a63fc1b097 1.65kB / 1.65kB      0.0s
 => => sha256:62cb64d073a60a041978385fefea85495fd1c17e4299c36469474f49db3e8297 1.37kB / 1.37kB      0.0s
 => => sha256:9d0395fd956cafb3f70fe0d3aaf7dcc647e19b570cf5bec4a99d619cafcd1a53 7.03kB / 7.03kB      0.0s
 => => sha256:c158987b05517b6f2c5913f3acef1f2182a32345a304fe357e3ace5fadcad715 3.37MB / 3.37MB      0.7s
 => => extracting sha256:c158987b05517b6f2c5913f3acef1f2182a32345a304fe357e3ace5fadcad715            0.1s
 => => sha256:ba319d964f14f150c609d2b7f0ca3a06297f25d3858ff154b6aa84e2e95fa954 230B / 230B          1.5s
 => => extracting sha256:7ec3a91b66d9050989fca52618b7d400610c8c2cf9a8bbcb86bc403e0e01f220            0.1s
 => => sha256:42a0f6adb0e9ffe9218ddbb5ad9644f2b8c1a6246f3a010644d61960c27ab042 3.04MB / 3.04MB      2.3s
 => => extracting sha256:ebea74ce8637967595d874b851ae620c6a2b8e8cdd1a2075c307c788838c4859            0.5s
 => => extracting sha256:ba319d964f14f150c609d2b7f0ca3a06297f25d3858ff154b6aa84e2e95fa954            0.0s
 => => extracting sha256:42a0f6adb0e9ffe9218ddbb5ad9644f2b8c1a6246f3a010644d61960c27ab042            0.2s
 => [2/5] COPY . /app                                                                               0.1s
 => [3/5] WORKDIR /app                                                                              0.0s
 => [4/5] RUN apk add gcc musl-dev python3-dev libffi-dev openssl-dev                              22.5s
 => [5/5] RUN pip install -r requirements.txt                                                       7.0s
 => exporting to image                                                                              1.4s
 => => exporting layers                                                                             1.4s
 => => writing image sha256:18f57a0a1d90e484557f45ecd8302ceabff67499c726757b45b92d46ea696aef        0.0s
 => => naming to docker.io/library/webappdocker-app                                                 0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
```

Figure 10 docker-compose build

Step 2: Run the containers

$ docker-compose up

```
PS C:\Users\     \Desktop\webappdocker> docker-compose up
[+] Running 12/12
 - db Pulled                                                                                       20.6s
   - 0ed027b72ddc Pull complete                                                                     5.3s
   - 0296159747f1 Pull complete                                                                     5.3s
   - 3d2f9b664bd3 Pull complete                                                                     5.4s
   - df6519f81c26 Pull complete                                                                     5.6s
   - 36bb5e56d458 Pull complete                                                                     5.6s
   - 054e8fde88d0 Pull complete                                                                     5.7s
   - f2b494c50c7f Pull complete                                                                     8.4s
   - 132bc0d471b8 Pull complete                                                                     8.5s
   - 135ec7033a05 Pull complete                                                                    13.1s
   - 5961f0272472 Pull complete                                                                    13.2s
   - 75b5f7a3d3a4 Pull complete                                                                    13.2s
[+] Running 3/3
 - Network webappdocker_default  Created                                                            0.7s
 - Container webappdocker-db-1   Created                                                            0.4s
 - Container webappdocker-app-1  Created                                                            0.1s
Attaching to webappdocker-app-1, webappdocker-db-1
webappdocker-db-1    | 2023-01-06 14:30:01+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.0.31-1.el8 started.
webappdocker-db-1    | 2023-01-06 14:30:01+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'
webappdocker-db-1    | 2023-01-06 14:30:01+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.0.31-1.el8 started.
webappdocker-db-1    | 2023-01-06 14:30:01+00:00 [Note] [Entrypoint]: Initializing database files
webappdocker-db-1    | 2023-01-06T14:30:01.959601Z 0 [Warning] [MY-011068] [Server] The syntax '--skip-host-cache' is deprecated and will
 be removed in a future release. Please use SET GLOBAL host_cache_size=0 instead.
webappdocker-db-1    | 2023-01-06T14:30:01.959669Z 0 [System] [MY-013169] [Server] /usr/sbin/mysqld (mysqld 8.0.31) initializing of serve
r in progress as process 80
webappdocker-db-1    | 2023-01-06T14:30:01.968087Z 1 [System] [MY-013576] [InnoDB] InnoDB initialization has started.
webappdocker-db-1    | 2023-01-06T14:30:02.300987Z 1 [System] [MY-013577] [InnoDB] InnoDB initialization has ended.
webappdocker-db-1    | 2023-01-06T14:30:03.624316Z 6 [Warning] [MY-010453] [Server] root@localhost is created with an empty password ! Pl
ease consider switching off the --initialize-insecure option.
webappdocker-db-1    | 2023-01-06 14:30:06+00:00 [Note] [Entrypoint]: Database files initialized
webappdocker-db-1    | 2023-01-06 14:30:06+00:00 [Note] [Entrypoint]: Starting temporary server
webappdocker-db-1    | 2023-01-06T14:30:07.014841Z 0 [Warning] [MY-011068] [Server] The syntax '--skip-host-cache' is deprecated and will
 be removed in a future release. Please use SET GLOBAL host_cache_size=0 instead.
webappdocker-db-1    | 2023-01-06T14:30:07.015651Z 0 [System] [MY-010116] [Server] /usr/sbin/mysqld (mysqld 8.0.31) starting as process 1
37
webappdocker-db-1    | 2023-01-06T14:30:07.029683Z 1 [System] [MY-013576] [InnoDB] InnoDB initialization has started.
```

Figure 11 docker compose is running

```
webappdocker-db-1   | 2023-01-06 14:30:11+00:00 [Note] [Entrypoint]: Temporary server stopped
webappdocker-db-1   |
webappdocker-db-1   | 2023-01-06 14:30:11+00:00 [Note] [Entrypoint]: MySQL init process done. Ready for start up.
webappdocker-db-1   |
webappdocker-db-1   | 2023-01-06T14:30:12.069425Z 0 [Warning] [MY-011068] [Server] The syntax '--skip-host-cache' is deprecated and will
 be removed in a future release. Please use SET GLOBAL host_cache_size=0 instead.
webappdocker-db-1   | 2023-01-06T14:30:12.070173Z 0 [System] [MY-010116] [Server] /usr/sbin/mysqld (mysqld 8.0.31) starting as process 1

webappdocker-db-1   | 2023-01-06T14:30:12.075238Z 1 [System] [MY-013576] [InnoDB] InnoDB initialization has started.
webappdocker-db-1   | 2023-01-06T14:30:12.161359Z 1 [System] [MY-013577] [InnoDB] InnoDB initialization has ended.
webappdocker-db-1   | 2023-01-06T14:30:12.358273Z 0 [Warning] [MY-010068] [Server] CA certificate ca.pem is self signed.
webappdocker-db-1   | 2023-01-06T14:30:12.358328Z 0 [System] [MY-013602] [Server] Channel mysql_main configured to support TLS. Encrypte
d connections are now supported for this channel.
webappdocker-db-1   | 2023-01-06T14:30:12.360703Z 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 different directory.
webappdocker-db-1   | 2023-01-06T14:30:12.374592Z 0 [System] [MY-011323] [Server] X Plugin ready for connections. Bind-address: '::' por
t: 33060, socket: /var/run/mysqld/mysqlx.sock
webappdocker-db-1   | 2023-01-06T14:30:12.374710Z 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.
webappdocker-app-1  |  * Serving Flask app 'app'
webappdocker-app-1  |  * Debug mode: on
webappdocker-app-1  | WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server inst
ead.
webappdocker-app-1  |  * Running on all addresses (0.0.0.0)
webappdocker-app-1  |  * Running on http://127.0.0.1:5000
webappdocker-app-1  |  * Running on http://172.18.0.3:5000
webappdocker-app-1  | Press CTRL+C to quit
webappdocker-app-1  |  * Restarting with stat
webappdocker-app-1  |  * Debugger is active!
webappdocker-app-1  |  * Debugger PIN: 744-286-485
```

Figure 12 noted that only after the database is up, will the website run

Step 3: Check the result: goto http://localhost

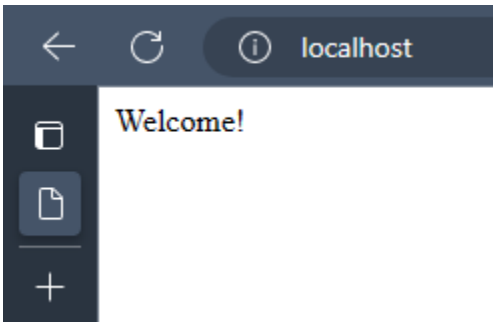


Figure 13 website is up!

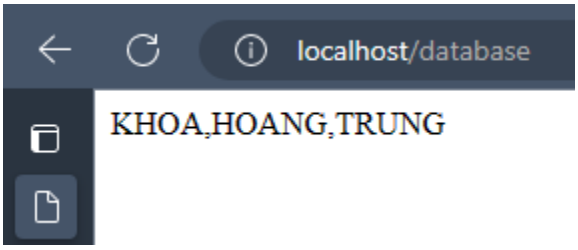


Figure 14 read from the database

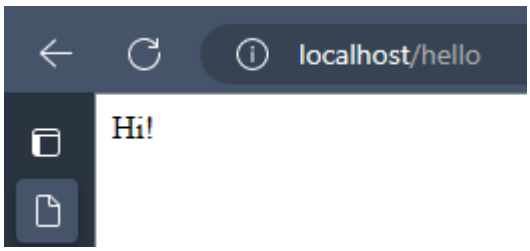


Figure 15 Check if site is working as intended

III. RESULT AND CONCLUSION

We showed a simple way to run a simple web application by using docker and docker compose. Docker makes the whole web application portable, meaning it can be deployed anywhere just by running docker pull and docker run, no need for manual configuration or setup. It “encapsulate” the whole application, just a single container in a virtual environment mimicking a standalone server and a virtual machine.

IV. APPENDIX

1. **TASK**

Member	Task	Self-assessment
Doan Nguyen Dang Khoa	Setup Docker, Demo	100%
Nguyen Viet Hoang	Setup Docker, Research	80%
Nguyen Duc Trung	Setup Docker, Research	80%

2. **ANSWER**

We don’t have any questions for our’s group in the excel file uploaded on courses website.

----------END----------