VIỆN CÔNG NGHỆ THÔNG TIN T3H

Đào tạo chuyên sâu - Trải nghiệm thực tế

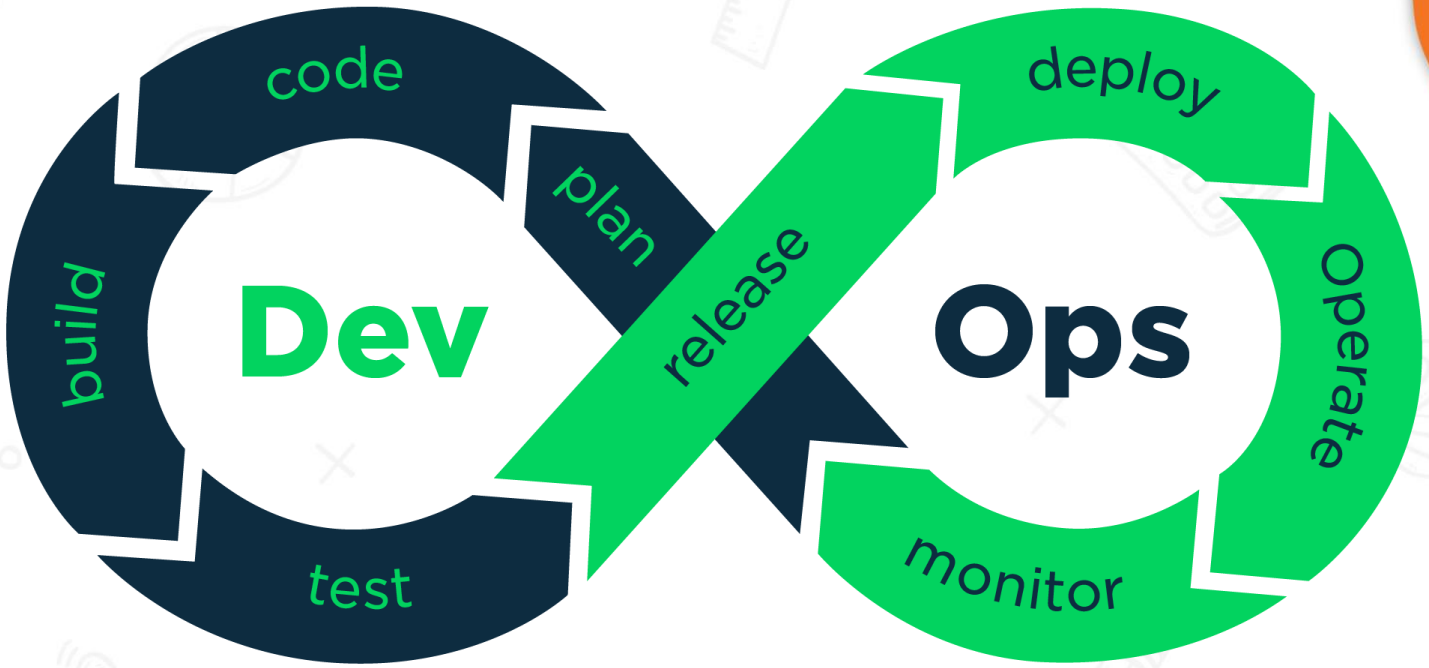
Môn học

DevOps

Giảng viên

Tan Do

0868880797



Nội dung buổi 02

1. Tạo các Apps với Docker
   - Web server
   - Application server
   - Database server
2. Dockerfile
3. Truy cập và xem log các app
4. Các lệnh cơ bản Linux

Tạo Server với Docker

- Ubuntu
  - docker run -it ubuntu
  - docker run -it --rm ubuntu
- CentOS
  - docker run -it centos
  - docker run -it --rm centos

Tạo Web Server với Docker

- NGINX
  - docker run -d -p 80:80 --name my-nginx nginx
  - docker run -it -p 80:80 --name my-nginx nginx
  - docker run -it --rm -p 80:80 --name my-nginx nginx
  - docker run -d -p 80:80 --name my-nginx -v C:/my-website:/usr/share/nginx/html nginx # -v: volumn
- APACHE HTTP
  - docker run -d -p 80:80 --name my-apache httpd

Tạo Application Server với Docker

- APACHE TOMCAT
  - docker run -d -p 8080:8080 --name my-tomcat tomcat # p: port
  - Truy cập vào container và copy source example để chạy
    - docker exec -it my-tomcat /bin/bash # -i: interactive, -t: tty teletypewriter (nếu run lỗi trên GitBash thì run trên Command Prompt)
    - ls
    - cd webapps.dist
    - cp -R * ../webapps/ # cp: copy, -R: recursive

Tạo Database Server với Docker

- MySQL
  - docker run --name my-mysql -e MYSQL_ROOT_PASSWORD=my-secret-pw -p 3306:3306 -d mysql
    # -e: environment

Dockerfile là gì?

- Là một tập tin văn bản đơn giản chứa các hướng dẫn để xây dựng một Docker image
- Được sử dụng để định nghĩa cách mà image sẽ được tạo, bao gồm các bước để cài đặt và cấu hình các phần mềm, môi trường, và tài nguyên khác cần cho ứng dụng
- Giúp tự động hóa quy trình xây dựng image và đảm bảo rằng môi trường trong image được xây dựng theo mong muốn
- Có thể sử dụng Dockerfile để tái tạo môi trường phát triển trên các máy chủ khác nhau hoặc chia sẻ môi trường với các thành viên
- Thường chứa các lệnh: FROM, RUN, COPY, ADD, CMD, ENTRYPOINT

Các lệnh thường dùng trong Dockerfile

- FROM: Xác định image cơ sở của image mới
- RUN: Thực thi các lệnh trong môi trường container và tạo ra một layer mới trong quá trình xây dựng image
- COPY và ADD: Sao chép tệp tin và thư mục từ hệ thống tệp tin của máy host vào image
- CMD và ENTRYPOINT: Xác định lệnh mặc định mà container sẽ thực thi khi được khởi chạy

Tạo Web server với Dockerfile

- Tạo file Dockerfile cho NGINX Web Server

```
FROM nginx:alpine
COPY . /usr/share/nginx/html
# COPY ./my-website /usr/share/nginx/html
```

- Build & run
  - docker build -t my-image .
  - docker run -it --rm -d -p 80:80 --name my-server my-image
  - Nếu dùng tên file khác thì sử dụng option -f
    - docker build -t image:version --force-rm -f Dockerfile1 .

Tạo Web server với Dockerfile

- Tạo Apache HTTP Web Server (1)
  - Tạo Dockerfile

```
FROM centos:centos7
RUN yum update -y
RUN yum install httpd httpd-tools -y
RUN yum install vim -y
RUN yum install epel-release -y
RUN yum update -y
RUN yum install htop -y
WORKDIR /var/www/html
EXPOSE 80
ADD . /var/www/html/
ENTRYPOINT [ "httpd" ]
CMD [ "-D", "FOREGROUND"]
```

  - Build & run
    - docker build -t my-image .
    - docker run -it --rm -d -p 80:80 --name my-server my-image

Tạo Web server với Dockerfile

- Tạo Apache HTTP Web Server (2)
  - Tạo Dockerfile

```
FROM centos:centos7
RUN yum update -y \
    && yum install httpd httpd-tools -y \
    && yum install vim -y \
    && yum install epel-release -y \
    && yum update -y \
    && yum install htop -y \
WORKDIR /var/www/html
EXPOSE 80
ADD . /var/www/html/
ENTRYPOINT [ "httpd" ]
CMD [ "-D", "FOREGROUND"]
```

  - Build & run
    - docker build -t my-image .
    - docker run -it --rm -d -p 80:80 --name my-server my-image

Tạo Web server với Dockerfile

- Tạo Web server với source code ReactJS (1)
    - Khởi tạo source code

        npx create-react-app my-react-app
    - Hoặc download source code

        https://github.com/nazaninsbr/React-TodoList

Tạo Web server với Dockerfile

- Tạo Web server với source code ReactJS (2)
  - Tạo Dockerfile

```
# Sử dụng một image chứa Node.js phiên bản gần nhất làm base image
FROM node:latest as build
# Thiết lập thư mục làm việc
WORKDIR /app
# Sao chép package.json và package-lock.json (nếu có) vào thư mục làm việc
COPY package*.json ./
# Cài đặt dependencies
RUN npm install
# Sao chép các file và thư mục từ dự án React vào thư mục làm việc
COPY . .
# Build dự án React
RUN npm run build
# Sử dụng một web server nhỏ để phục vụ ứng dụng React

FROM nginx:alpine
# Sao chép các file build từ stage build vào thư mục mặc định của nginx
COPY --from=build /app/build /usr/share/nginx/html
# Port mặc định của nginx
EXPOSE 80
# Lệnh chạy web server khi container được khởi chạy
CMD ["nginx", "-g", "daemon off;"]
```

Tạo Web server với Dockerfile

- Tạo Web server với source code ReactJS (3)
  - Build & run
    - docker build -t my-react-app .
    - docker run -p 3000:80 my-react-app

Tạo Application Server với Docker

- Nodejs Expressjs API Server (1)
  - Khởi tạo source code
    - mkdir nodejs-api
    - cd nodejs-api
    - npm init –y
    - npm install express

Tạo Application Server với Docker

- Nodejs Expressjs API Server (2)
  - Tạo server.js

```
const express = require('express');
const app = express();
const PORT = 3000;
app.get('/', (req, res) => {
    res.send('API server is running!');
});
app.listen(PORT, () => {
    console.log(`Server đang chạy tại http://localhost:${PORT}`);
});
```

Tạo Application Server với Docker

- Nodejs Expressjs API Server (3)
  - Tạo .dockerignore
    - cat > .dockerignore
    - node_modules
    - CTRL + D
  - Tạo Dockerfile

```
FROM node:latest
WORKDIR /app
COPY package.json .
RUN npm install
COPY . .
EXPOSE 3000
CMD ["node", "server.js"]
```

  - Build & run
    - docker build -t nodejs-api .
    - docker run -p 3000:3000 nodejs-api

Tạo Application Server với Docker

- Python Flask API Server (1)
  - Tạo app.py

```
# app.py
from flask import Flask
app = Flask(__name__)
@app.route('/')
def hello():
    return 'Hello, World!'
if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0')
```

Tạo Application Server với Docker

- Python Flask API Server (2)
  - Tạo Dockerfile

```
# Dockerfile
FROM python:3.12-alpine
# Set working directory
WORKDIR /app
# Copy the current directory contents into the container at /app
COPY . /app
# Install dependencies
RUN pip install --no-cache-dir flask
# Install requirements
# RUN pip install -r requirements.txt
# Expose the port the app runs on
EXPOSE 5000
# Command to run the application
CMD ["python", "app.py"]
```

Tạo Application Server với Docker

- Python Flask API Server (3)
  - Build & run
    - docker build -t my-python-app .
    - docker run -p 5000:5000 my-python-app

Tạo Angular Application với Docker

- Khởi tạo source code
  - ng new angular-docker --ssr false --routing true --style scss
  - npm start
- Hoặc download source code
  - https://github.com/rodrigokamada/angular-docker

Tạo Angular Application với Docker

- Tạo Dockerfile

```
FROM node:alpine

WORKDIR /usr/src/app

COPY . /usr/src/app

RUN npm install -g @angular/cli

RUN npm install

CMD ["ng", "serve", "--host", "0.0.0.0"]
```

- Build & run
  - docker build -t angular-docker .
  - docker run -p 4200:4200 angular-docker

Tạo Vuejs Application với Docker

- Khởi tạo source code
  - vue create docker-vuejs
  - cd docker-vuejs
  - npm run serve
- Hoặc download source code
  - https://github.com/IamManchanda/hello-world-vue

Tạo Vuejs Application với Docker

- Tạo Dockerfile

```
# Choose the Image which has Node installed already
FROM node:lts-alpine
# install simple http server for serving static content
RUN npm install -g http-server
# make the 'app' folder the current working directory
WORKDIR /app
# copy both 'package.json' and 'package-lock.json' (if available)
COPY package*.json ./
# install project dependencies
RUN npm install
# copy project files and folders to the current working directory (i.e. 'app' folder)
COPY . .
# build app for production with minification
RUN npm run build
EXPOSE 8080
CMD [ "http-server", "dist" ]
```

- Build & run
  - docker build -t docker-vuejs .
  - docker run -p 8080:8080 -d docker-vuejs

Tạo Vuejs Application với Docker

- Tạo Dockerfile với NGINX

```
# build stage
FROM node:lts-alpine as build-stage
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
RUN npm run build

# production stage
FROM nginx:stable-alpine as production-stage
COPY --from=build-stage /app/dist /usr/share/nginx/html
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

- Build & run
  - docker build -t docker-vuejs .
  - docker run -p 80:80 -d docker-vuejs

Tạo Application với Docker

- SpringBoot
  - https://github.com/hasankadirdemircan/hello-world-spring-boot-docker
  - https://gitlab.com/Raja786/docker-hello-world-spring-boot

Truy cập và xem log các app

- Show log a container
  - docker logs <tên container>
  - docker logs --follow <tên container>

So sánh CMD và ENTRYPOINT

- Giống nhau:
  - Để thực thi 1 task, command hay script
    - CMD [“python3”, “app.py”]
    - ENTRYPOINT [“python3”, “app.py”]

So sánh CMD và ENTRYPOINT

- Khác nhau:
  - Khác nhau ở cách chúng xử lý tham số tại thời điểm chạy
    - docker run container

CMD	Entrypoint
Nhận tham số khi chạy **docker run** để overide command được chỉ định trong CMD	Nhận tham số khi chạy **docker run** để làm tham số cho **Entrypoint**

```
FROM Ubuntu
CMD sleep 5
```

```
docker run ubuntu
```

Command at Startup: **sleep 5**

```
docker run ubuntu sleep 10
```

Command at Startup: **sleep 10**

```
FROM Ubuntu
ENTRYPOINT ["sleep"]
```

```
docker run ubuntu 10
```

Command at Startup: **sleep 10**

```
docker run ubuntu
```

```
sleep: missing operand
Try 'sleep --help' for more information.
```

So sánh CMD và ENTRYPOINT

- **CMD** có thể sử dụng theo các format sau trong Dockerfile

CMD executable, param1, param2

CMD [executable, param1, param2]

CMD sleep 5

CMD [“sleep”, “5”]

- **ENTRYPOINT** có thể sử dụng theo các format sau trong Dockerfile

ENTRYPOINT executable, param1, param2

ENTRYPOINT [executable, param1, param2]

ENTRYPOINT node, app.js

ENTRYPOINT [“node”, “app.js”]

So sánh CMD và ENTRYPOINT

- Nếu như trong Dockerfile sử dụng nhiều **CMD** hoặc **ENTRYPOINT** khác nhau thì CMD hoặc ENTRYPOINT cuối cùng sẽ được thực thi

```
FROM Ubuntu
CMD [“sleep”, "5”]
CMD [“sleep”, "10”]
```

```
docker run ubuntu
```

Command at Startup: **sleep 10**

- Nếu như trong Dockerfile đồng thời CMD và ENTRYPOINT thì CMD sẽ đóng vai trò là arguments cho ENTRYPOINT

```
FROM Ubuntu
CMD [“10”]
ENTRYPOINT [“sleep”]
```

```
docker run ubuntu
```

Command at Startup: **sleep 10**

T3H
IT-Institute

VIỆN CÔNG NGHỆ THÔNG TIN T3H

Đào tạo chuyên sâu - Trải nghiệm thực tế

THANK YOU

www.t3h.edu.vn