VIỆN CÔNG NGHỆ THÔNG TIN T3H

Đào tạo chuyên sâu - Trải nghiệm thực tế

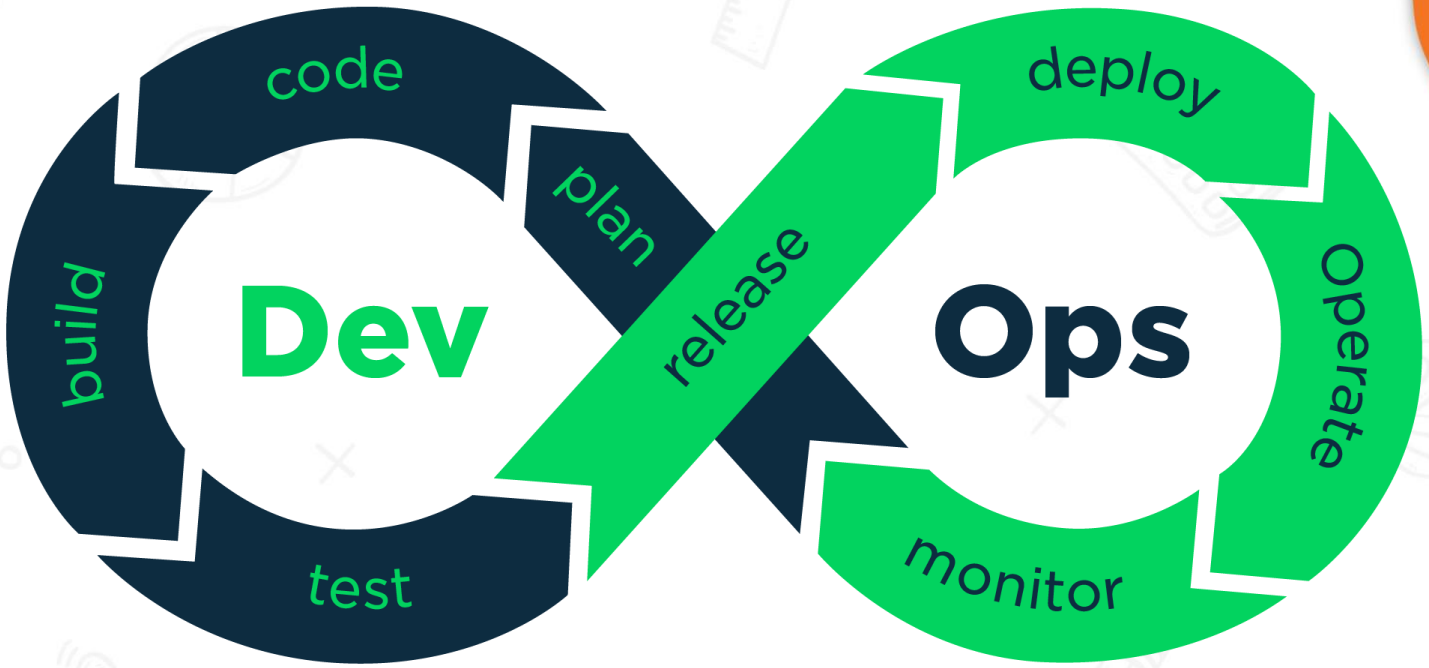
Môn học

DevOps

Giảng viên

Tan Do

0868880797



VIỆN CÔNG NGHỆ THÔNG TIN T3H

Đào tạo chuyên sâu - Trải nghiệm thực tế

Nội dung buổi 04

- YAML
- Docker Compose

YAML là gì?

- YAML Ain't Markup Language
- YAML không phải là 1 ngôn ngữ đánh dấu (như HTML, XML)
- Là 1 định dạng dữ liệu trung gian được thiết kế để người dùng và các ngôn ngữ lập trình cùng hiểu được
- Được dùng vào mục đích tương tự JSON (JavaScript Object Notation), XML (eXtensible Markup Language) nhưng nó lại có nhiều tính năng nổi bật hơn
  - Cấu trúc dữ liệu linh hoạt hơn
  - Hỗ trợ nhiều ngôn ngữ lập trình
  - Diễn đạt và mở rộng dữ liệu hơn
  - Dễ sử dụng vì khá có nhiều kiểu dữ liệu lập trình
- Chi tiết http://www.yaml.org/

Cú pháp YAML

- Giống cú pháp của Python, YAML yêu cầu thụt đầu dòng trước mỗi câu. Thụt đầu dòng bởi các dấu cách (tùy cấu trúc (khối lệnh – block) mà dùng 1 hay 2,4… dấu cách), không dùng tab
- Dùng dấu # để bắt đầu comment
- Dấu “-” để bắt đầu cho 1 list các phần tử
- Kiểu dữ liệu YAML:
  - Mỗi cặp key – value ở YAML đều chứa những value vô hướng
  - Key – Value là yếu tố chủ chốt trong cú pháp của YAML
  - Mỗi mục của nó có thể thuộc một từ điển bất kỳ
  - Key luôn là một chuỗi thì value lại có tính chất vô hướng nên nó có thể là kiểu dữ liệu chuỗi, số hoặc một từ điển

So sánh XML JSON YAML

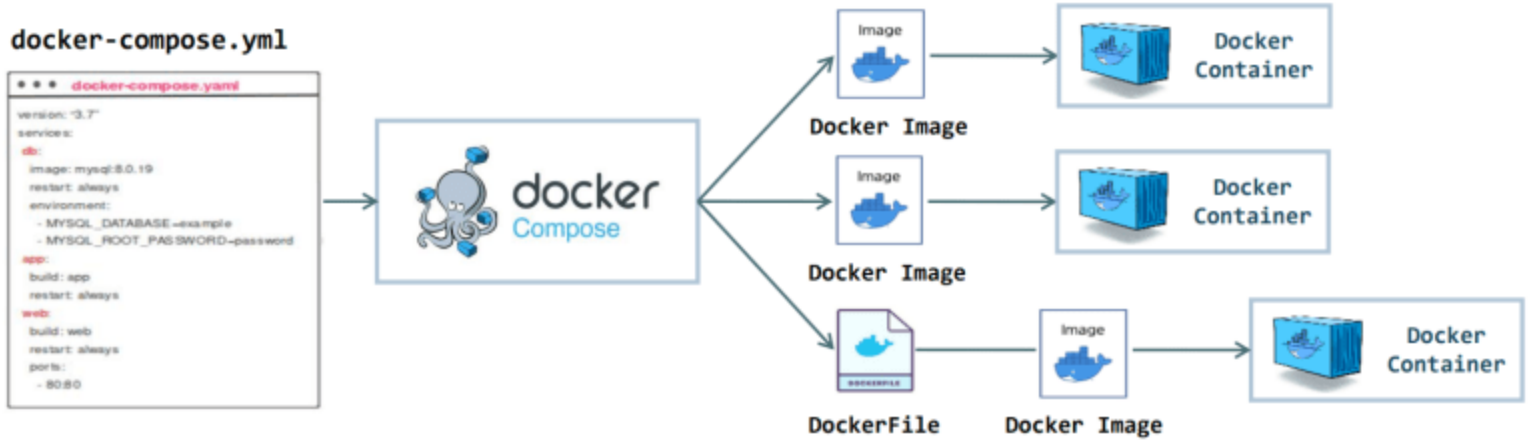
```xml
<?xml version="1.0" encoding="UTF-8"?>
<root>
    <store>Viblo</store>
    <address>Asia</address>
    <fruits>
        <fruit>
            <name>Orange</name>
            <price>$1</price>
        </fruit>
        <fruit>
            <name>Banana</name>
            <price>$2</price>
        </fruit>
    </fruits>
</root>
```

```json
{
  "store": "Viblo",
  "address": "Asia",
  "fruits": [
    {
      "name" : "Orange",
      "price" : "$1"
    },
    {
      "name" : "Banana",
      "price": "$2"
    }
  ]
}
```

```yaml
store: "Viblo"
address: "Asia"
fruits:
  - name: "Orange"
    price: "1$"
  - name: "Banana"
    price: "2$"
```

Docker compose là gì?

- Là một công cụ được sử dụng để định nghĩa và quản lý các ứng dụng đa-container trong Docker
- Là một phần mềm mã nguồn mở do Docker, Inc. phát triển, cho phép định nghĩa một tập hợp các dịch vụ ứng dụng (services) trong một file cấu hình duy nhất (thường là YAML), sau đó sử dụng lệnh docker-compose để triển khai và quản lý các dịch vụ này



Các tính năng Docker compose

- Định nghĩa dịch vụ:
  - Có thể định nghĩa các dịch vụ khác nhau và cấu hình bằng cách sử dụng Docker Compose. Ví dụ: có thể định nghĩa một dịch vụ database và một dịch vụ web
- Tự động tạo container:
  - Docker Compose sẽ tự động tạo các container cần thiết để triển khai ứng dụng. Nó sẽ tạo các container dựa trên cấu hình đã định nghĩa
- Quản lý mạng:
  - Docker Compose cung cấp các công cụ để quản lý mạng cho các container. Có thể định nghĩa các mạng riêng để giữ cho các container an toàn và đảm bảo chúng không bị tấn công từ bên ngoài
- Quản lý lưu trữ:
  - Có thể định nghĩa các khối lượng lưu trữ để sử dụng cho các container. Docker Compose sẽ giúp quản lý các khối lượng lưu trữ này và đảm bảo chúng được lưu trữ và quản lý một cách an toàn và hiệu quả
- Tích hợp với Docker Swarm:
  - Docker Compose có thể tích hợp với Docker Swarm để triển khai ứng dụng trên một cụm máy chủ Docker. Điều này giúp quản lý và mở rộng các ứng dụng một cách dễ dàng
- Khả năng mở rộng:
  - Docker Compose cho phép mở rộng ứng dụng bằng cách thêm các dịch vụ mới hoặc tăng số lượng container cho các dịch vụ hiện có. Điều này giúp quản lý tốt hơn sự tăng trưởng của ứng dụng

Vì sao nên dùng Docker compose?

- Đơn giản hóa triển khai ứng dụng:
  - Thay vì phải chạy từng lệnh docker run cho từng container, có thể định nghĩa toàn bộ cấu hình trong một file compose và chỉ cần sử dụng docker-compose up để triển khai tất cả các dịch vụ
- Quản lý nhiều container:
  - Docker Compose cho phép quản lý nhiều container đồng thời, với các cài đặt mạng, lưu trữ và môi trường được xác định rõ ràng
- Môi trường nhất quán:
  - Bằng cách sử dụng Docker Compose, có thể đảm bảo rằng mọi người trong nhóm làm việc trên cùng một môi trường phát triển và triển khai, giúp tránh được các vấn đề do khác biệt môi trường gây ra
- Tích hợp với các công cụ CI/CD:
  - Docker Compose dễ dàng tích hợp vào các quy trình CI/CD để tự động hóa việc xây dựng và triển khai ứng dụng
- Dễ dàng mở rộng:
  - Khi ứng dụng mở rộng, có thể chỉnh sửa file compose để thêm hoặc loại bỏ các dịch vụ mà không cần thay đổi nhiều trong cấu hình

Lưu ý quan trọng Docker compose

- Đặt tên cho các container:
  - Để dễ dàng quản lý các container, nên đặt tên cho chúng. Điều này sẽ giúp xác định được container nào đang chạy trong ứng dụng
- Sử dụng mạng riêng:
  - Khi sử dụng Docker Compose, nên sử dụng một mạng riêng để giữ cho các container an toàn và được cô lập khỏi mạng bên ngoài
- Sử dụng các biến môi trường:
  - Sử dụng các biến môi trường trong tệp YAML của để tránh lưu trữ các thông tin nhạy cảm trong tệp YAML. Các biến môi trường có thể được đặt trong một tệp .env riêng biệt
  - Để tránh xung đột khi khởi động container, nên đặt các cổng cho container trong tệp YAML, nên sử dụng cổng được khuyến nghị của Docker
- Các lệnh Docker Compose:
  - Có thể được thực thi thông qua một Makefile. Điều này giúp cho việc sử dụng Docker Compose trở nên đơn giản và hiệu quả hơn

Cài đặt Docker compose

- Docker Desktop:
  - Là ứng dụng dễ cài đặt cho môi trường Windows và macOS
  - Bao gồm cả
    - Docker Client
    - Docker Daemon
    - Docker Compose
    - Docker Content Trust
    - Kubernetes
    - Credential Helper
- Linux
  - sudo curl -L https://github.com/docker/compose/releases/download/v2.5.0/docker-compose-`uname -s`-`uname -m` -o /usr/local/bin/docker-compose
  - sudo chmod +x /usr/local/bin/docker-compose
- Check version
  - docker-compose --version

Tạo ứng dụng Hello World

- Tạo file docker-compose.yml

```
version: '3.9'
services:
   hello-world:
      image:
         hello-world:latest
```

- Run
  - docker-compose up
  - docker-compose up -d
  - docker ps –a
  - docker-compose down

Các lệnh cơ bản Docker compose

- docker-compose up: Khởi động các container
- docker-compose down: Dừng và xóa các container
- docker-compose ps: Hiển thị trạng thái của các container
- docker-compose build: Tạo image từ Dockerfile trong mỗi dịch vụ
- docker-compose restart: Khởi động lại các container
- docker-compose stop: Dừng các container
- docker-compose rm: Xóa các container không sử dụng
- docker-compose logs: Hiển thị các logs của các container
- docker-compose config: Hiển thị các cấu hình của Docker Compose
- docker-compose exec: Thực thi một lệnh trên một container
- docker-compose port: Hiển thị các port của các container
- docker-compose top: Hiển thị các process đang chạy trong các container

Tạo file Docker compose webserver

```yaml
version: '3.9'

services:
  web:
    image: nginx:latest
    ports:
      - "80:80"
    volumes:
      - ./web:/usr/share/nginx/html
    networks:
      - webnet

networks:
  webnet:
```

Tạo file Docker compose webserver

```
version: '3.9'
services:
  web1:
    image: nginx
    volumes:
      - .:/usr/share/nginx/html
    ports:
      - '81:80'
  web2:
    image: nginx
    ports:
      - '82:80'
```

Tạo file Docker compose mysql server

```yaml
version: '3.9'

services:
  mysql:
    image: mysql:latest
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: your_root_password_here
      MYSQL_DATABASE: your_database_name_here
      MYSQL_USER: your_mysql_user_here
      MYSQL_PASSWORD: your_mysql_password_here
    ports:
      - "3306:3306"
    volumes:
      - mysql-data:/var/lib/mysql

volumes:
  mysql-data:
    driver: local
```

Tạo file Docker compose postgresql server

```yaml
version: '3.9'

services:
  db:
    image: postgres:latest
    environment:
      POSTGRES_USER: example
      POSTGRES_PASSWORD: example
      POSTGRES_DB: example
    volumes:
      - dbdata:/var/lib/postgresql/data
    networks:
      - webnet

networks:
  webnet:

volumes:
  dbdata:
```

Ý nghĩa của các giá trị trong file

- version: phiên bản của Docker Compose. Ở đây sử dụng phiên bản 3
- services: là khu vực khai báo các services cần thiết cho ứng dụng
- web: dịch vụ web, sử dụng image nginx, chia sẻ volume và network với dịch vụ db. Cổng 80 được định nghĩa để web có thể truy cập được từ bên ngoài
- db: dịch vụ db, sử dụng image postgresql, chia sẻ volume và network với dịch vụ web. Các biến môi trường cài đặt cho dịch vụ postgresql được định nghĩa ở phần environment
- networks: danh sách các networks được sử dụng cho container
- webnet: mạng webnet để chia sẻ giữa dịch vụ web và db
- volumes: là option nên config, volumes cho phép mount data từ container ra máy local. Khi config option này thì mỗi lần stop container data của container đó sẽ không bị mất đi
- dbdata: là container chứa thông tin về database.

Tạo file Docker compose

- Web server
- Database server

```yaml
version: '3.9'

services:
  web:
    image: nginx:latest
    ports:
      - "80:80"
    volumes:
      - ./web:/usr/share/nginx/html
    networks:
      - webnet
    depends_on:
      - db
  db:
    image: postgres:latest
    environment:
      POSTGRES_USER: example
      POSTGRES_PASSWORD: example
      POSTGRES_DB: example
    volumes:
      - dbdata:/var/lib/postgresql/data
    networks:
      - webnet

networks:
  webnet:

volumes:
  dbdata:
```

T3H
IT-Institute

VIỆN CÔNG NGHỆ THÔNG TIN T3H

Đào tạo chuyên sâu - Trải nghiệm thực tế

THANK YOU

www.t3h.edu.vn