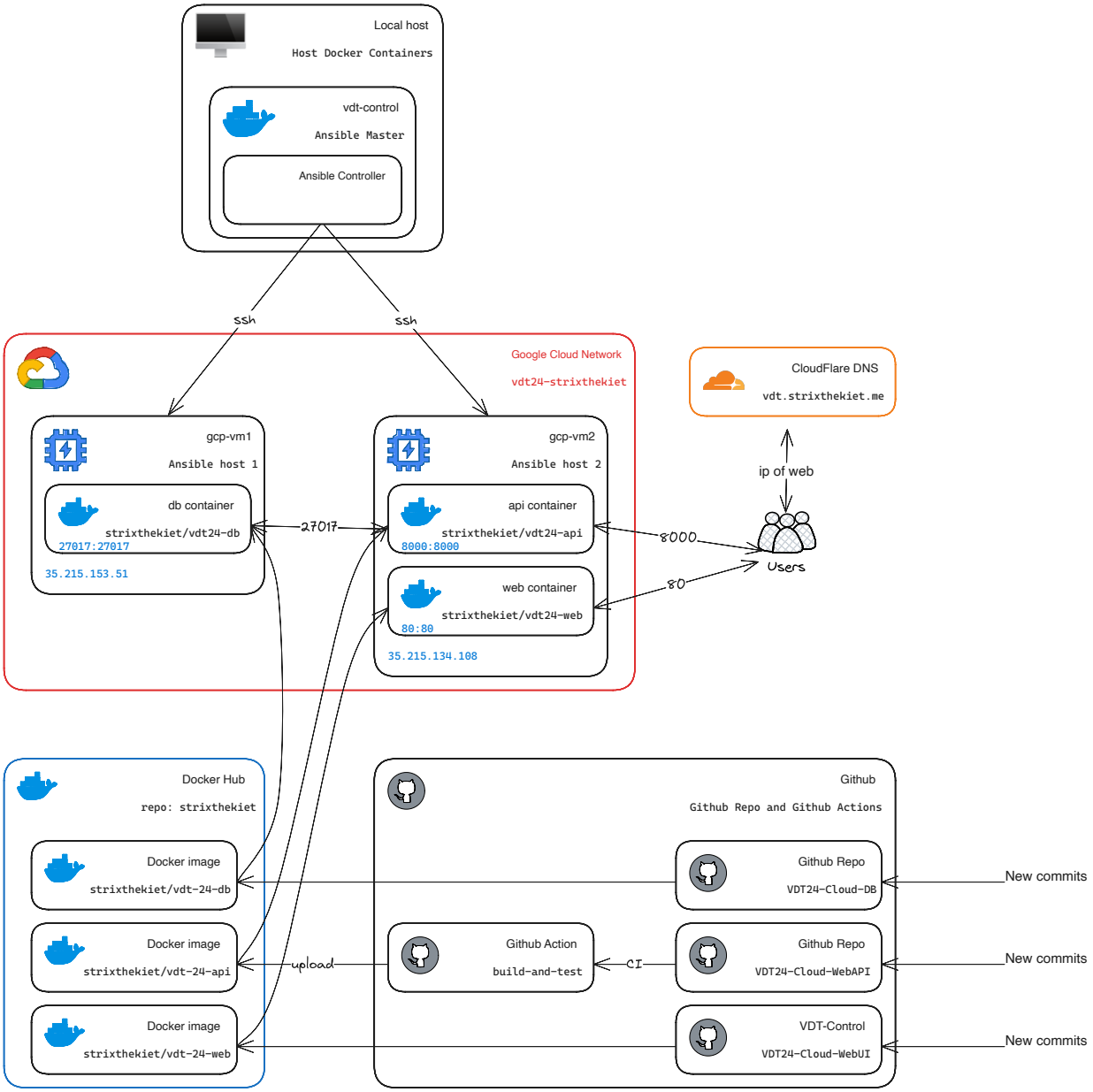
Phát triển một 3-tier web application đơn giản



- Frontend:
  - Github link
  - http://vdt.strixthekiet.me , http://35.215.134.108:80/

VDT2024 - Nguyen Don The Kiet - Student Management

Students

ID	Name	University
0	Lê Minh Việt Anh	Đại học Bách Khoa Hà Nội
2	Hoàng Bá Bảo	Đại học Bách Khoa Hà Nội
3	Phạm Minh Cường	Đại học Công nghệ - Đại học Quốc gia Hà Nội
1	Phạm Quang Bách	VinUniversity
5	Hoàng Việt Dũng	ITMO University
6	Bùi Hoàng Dũng	Đại học Bách Khoa Hà Nội
7	Bùi Trọng Dũng	Đại học Bách Khoa Hà Nội
8	Trần Thùy Dương	Đại học Công nghệ - Đại học Quốc gia Hà Nội
9	Quach Dang Giang	Đại học Công nghệ Thông tin – Đại học Quốc gia TP.HCM
10	Nguyễn Thanh Hà	Đại học Bách Khoa Hà Nội
11	Nguyễn Thu Hà	Đại học Bách Khoa Hà Nội
12	Cù Xuân Hải	ITMO University
13	Nguyễn Trung Hiếu	Đại học Bách Khoa Hà Nội
14	Đặng Việt Hoàng	Đại học Bách Khoa Hà Nội
15	Nguyễn Văn Hùng	Đại học Bách Khoa Hà Nội
16	Nguyễn Quốc Hưng	Đại học Bách Khoa Hà Nội
17	Lê Minh Hương	Đại học Công nghệ - Đại học Quốc gia Hà Nội
18	Ngo Dang Huy	Đại học Công nghệ - Đại học Quốc gia Hà Nội
19	Trần Quang Huy	Học viện Công Nghệ Bưu Chính Viễn Thông
20	Trần Minh Huyền	Đại học Bách Khoa Hà Nội
21	To Huong Thao's BF	strix
22	Lê Hoàng Trường	ITMO University
27	Đỗ Dương Mạnh	ITMO University
28	Nguyễn Ngọc Minh	Học viện Công Nghệ Bưu Chính Viễn Thông
29	Vũ Thế Nam	Đại học Công nghệ - Đại học Quốc gia Hà Nội
30	Ngô Công Bằng	Học viện Công Nghệ Bưu Chính Viễn Thông
31	Nguyễn Huy Thái	Đại học Công nghệ - Đại học Quốc gia Hà Nội
32	Phan Khôi Nguyên	Đại học Bách Khoa Hà Nội
33	Đinh Việt Quang	Đại học Bách Khoa Hà Nội
34	Nguyễn Chí Thành	Đại học Bách Khoa Hà Nội
35	Nguyễn Đình Tiến	Đại học Bách Khoa Hà Nội
36	Đỗ Thu Trang	Đại học Công nghệ - Đại học Quốc gia Hà Nội
37	Nguyễn Thị Mỹ Tú	ITMO University
38	Nguyễn Quang Tuấn	ITMO University
26	Nguyễn Thị Ngọc Mai	Đại học Công nghệ - Đại học Quốc gia Hà Nội
40	Phạm Mạnh Tuấn	ITMO University
41	Bùi Hoàng Vinh	Học viện Công Nghệ Bưu Chính Viễn Thông
39	Hoàng Minh Tuấn	Học viện Công Nghệ Bưu Chính Viễn Thông

View Details

View Details Student ID: 21

Name: To Huong Thao's BF

Gender: Male

University: strix

Year of Birth: asdasd

Email: asd

Phone Number: asd

Country: asd

Delete Student

Delete Student Student ID: 5555

Student not found

Update Student

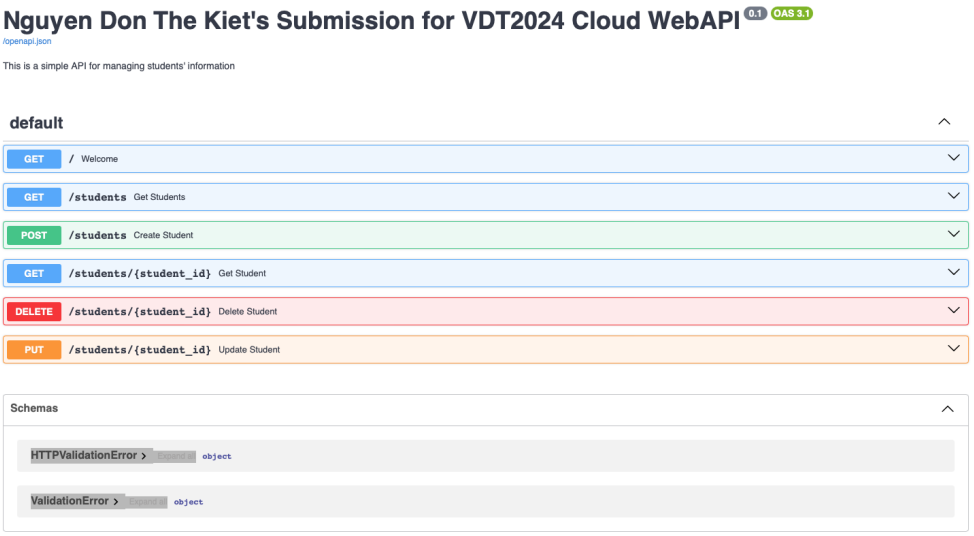
Update Student

Student ID:
Name:
Gender:
University:
Email:
Phone Number:
Country:
Year of Birth:

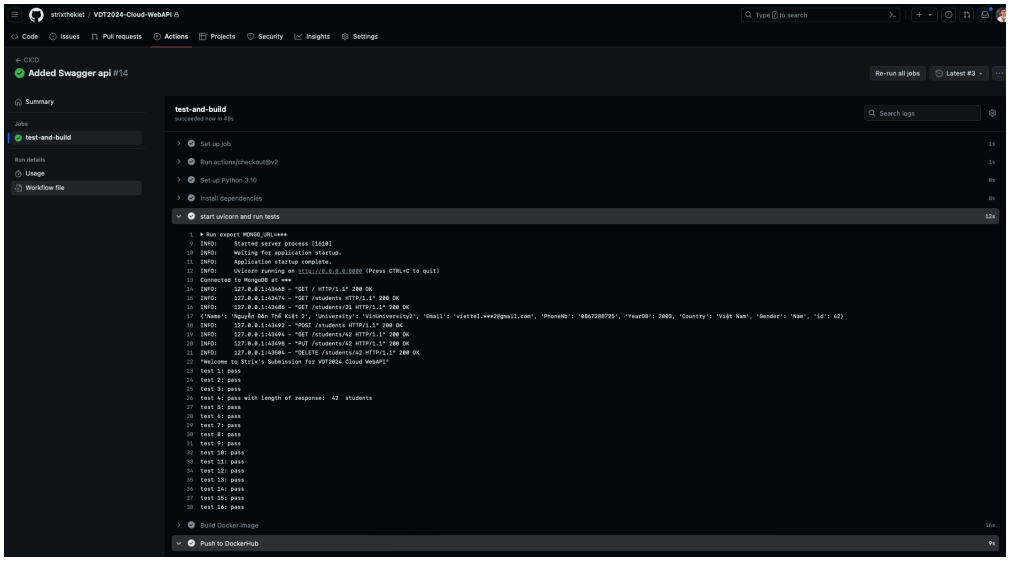
Add Student

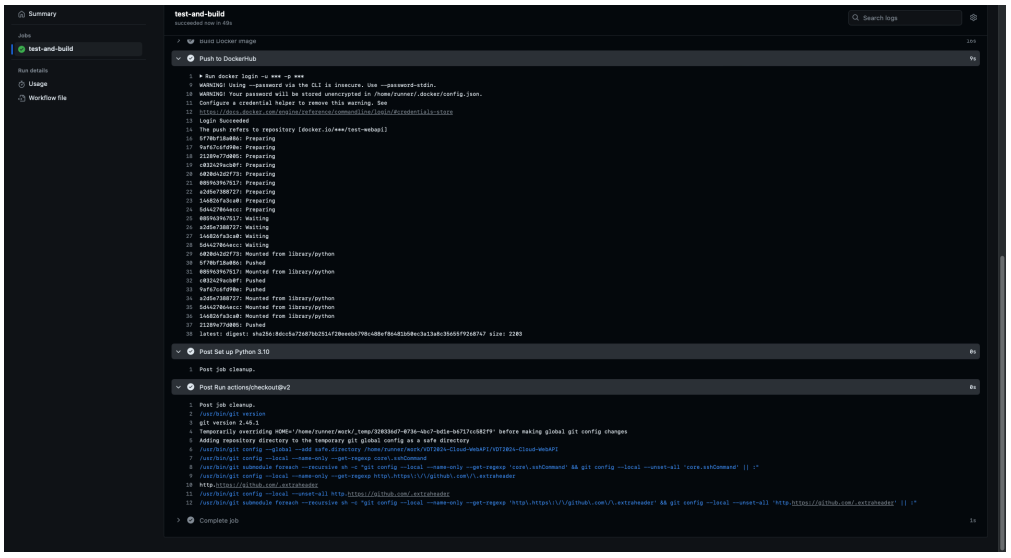
Name:
Gender:
University:
Email:
Phone Number
Country:
Year of Birth:

- API: http://35.215.134.108:8000/docs
  - Github link
  - Documentation:



  - git checkout dev; git commit -m "Added Swagger api":





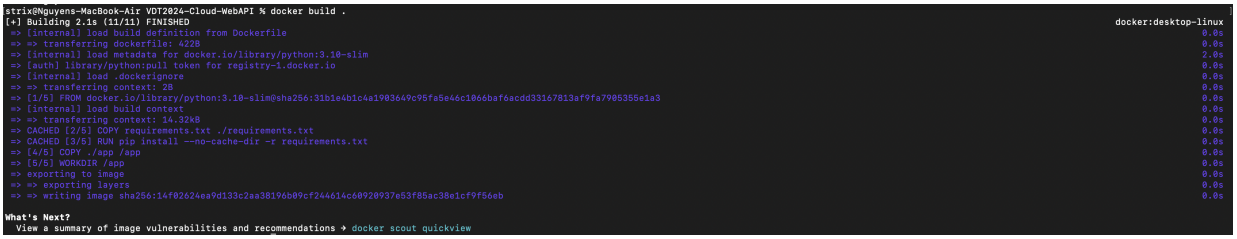
- Database: Mongo db 35.215.153.51:27017
  - Github link

Triển khai web application với DevOps tools & practices

- Database:
  - ```
    FROM --platform=amd64 mongo:7
        EXPOSE 27017
    ```

- API:
  -

```
    FROM --platform=amd64 python:3.10-slim
    # Install dependencies in a single layer with caching,
    # Copy directly into build context
    COPY requirements.txt ./requirements.txt
    RUN pip install --no-cache-dir -r requirements.txt
    # Copy application code
    COPY ./app /app
    WORKDIR /app
    ENV MONGO_URL=35.215.153.51
    # Start FastAPI
    CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port",
"8000"]
```



- Web:
  - ```
    FROM --platform=amd64 nginx
    COPY ./index.html /usr/share/nginx/html/index.html
    EXPOSE 80
    CMD ["nginx", "-g", "daemon off;"]
    ```

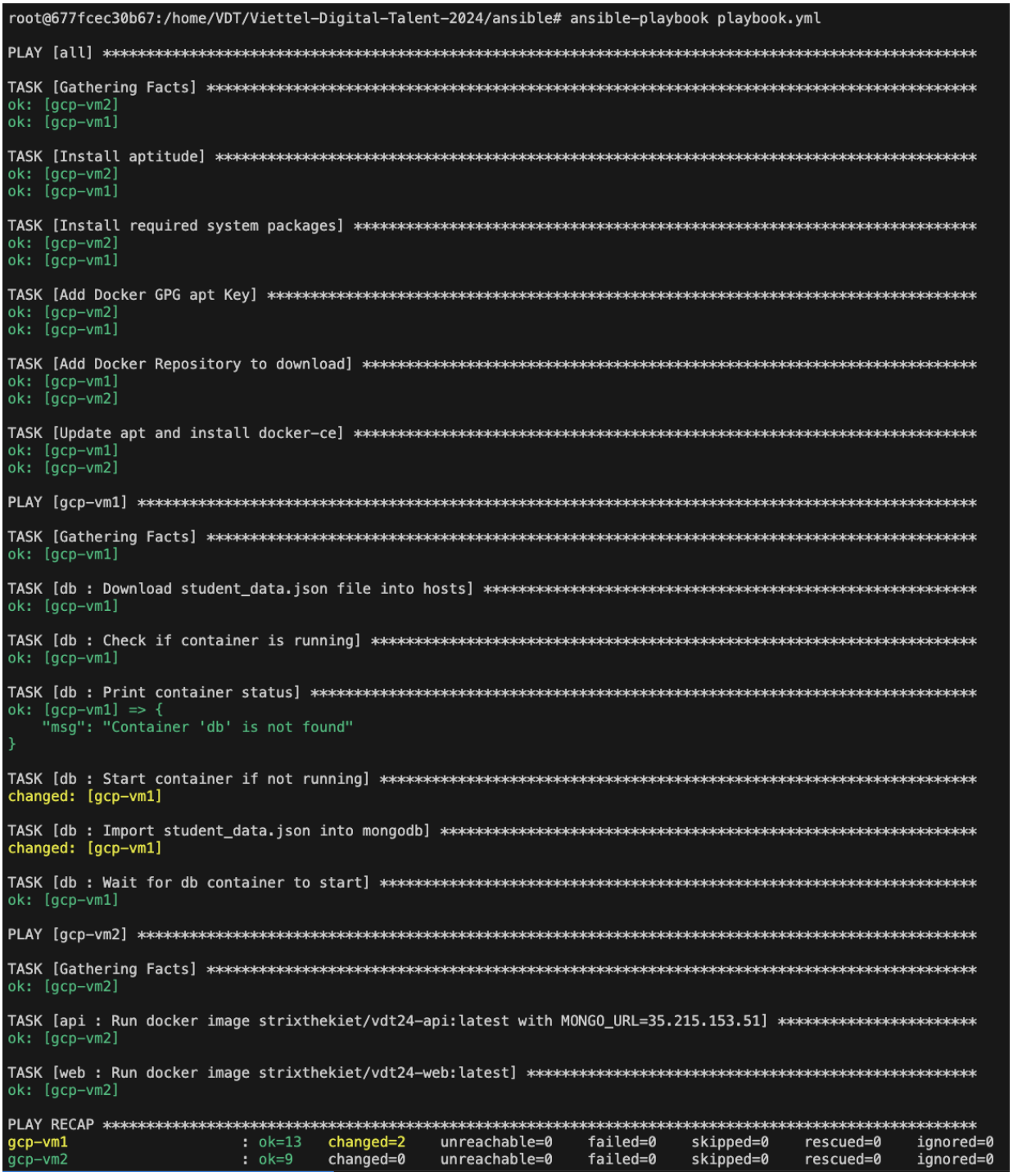
-

- Continuous integration:

```
name: CICD

on:
  pull_request:
    branches: [ main ]
  push:
jobs:
  test-and-build:
    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v2

    - name: Set up Python 3.10
      uses: actions/setup-python@v2
      with:
        python-version: '3.10'

    - name: Install dependencies
      run: |
        python -m pip install --upgrade pip
        pip install -r requirements.txt

    - name: start uvicorn and run tests
      run: |
        export MONGO_URL=${{ secrets.MONGO_IP }}
        uvicorn app.main:app --host 0.0.0.0 --port 8000 & sleep 5
        python app/test_main.py

    - name: Build Docker image
      run: |
        docker build -t vdt24-api:latest .

    - name: Push to DockerHub
      run: |
        docker login -u ${{ secrets.DOCKERHUB_USERNAME }} -p ${{
secrets.DOCKER_HUB_TOKEN }}
        docker tag vdt24-api:latest ${{ secrets.DOCKERHUB_USERNAME
}}/vdt24-api:latest
        docker push ${{ secrets.DOCKERHUB_USERNAME }}/vdt24-api:latest
```

-
-
-
-

- Automation with Ansible:
  - Github link



```
root@677fcec30b67:/home/VDT/Viettel-Digital-Talent-2024/ansible# ansible-playbook playbook.yml

PLAY [all] ********************************************************************************************************

TASK [Gathering Facts] ********************************************************************************************
ok: [gcp-vm2]
ok: [gcp-vm1]

TASK [Install aptitude] *******************************************************************************************
ok: [gcp-vm2]
ok: [gcp-vm1]

TASK [Install required system packages] ***************************************************************************
ok: [gcp-vm2]
ok: [gcp-vm1]

TASK [Add Docker GPG apt Key] *************************************************************************************
ok: [gcp-vm2]
ok: [gcp-vm1]

TASK [Add Docker Repository to download] **************************************************************************
ok: [gcp-vm1]
ok: [gcp-vm2]

TASK [Update apt and install docker-ce] ***************************************************************************
ok: [gcp-vm1]
ok: [gcp-vm2]

PLAY [gcp-vm1] ****************************************************************************************************

TASK [Gathering Facts] ********************************************************************************************
ok: [gcp-vm1]

TASK [db : Download student_data.json file into hosts] ************************************************************
ok: [gcp-vm1]

TASK [db : Check if container is running] *************************************************************************
ok: [gcp-vm1]

TASK [db : Print container status] ********************************************************************************
ok: [gcp-vm1] => {
    "msg": "Container 'db' is not found"
}

TASK [db : Start container if not running] ************************************************************************
changed: [gcp-vm1]

TASK [db : Import student_data.json into mongodb] *****************************************************************
changed: [gcp-vm1]

TASK [db : Wait for db container to start] ************************************************************************
ok: [gcp-vm1]

PLAY [gcp-vm2] ****************************************************************************************************

TASK [Gathering Facts] ********************************************************************************************
ok: [gcp-vm2]

TASK [api : Run docker image strixthekiet/vdt24-api:latest with MONGO_URL=35.215.153.51] **************************
ok: [gcp-vm2]

TASK [web : Run docker image strixthekiet/vdt24-web:latest] *******************************************************
ok: [gcp-vm2]

PLAY RECAP ********************************************************************************************************
gcp-vm1                    : ok=13   changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
gcp-vm2                    : ok=9    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Nghiên cứu sâu về một vấn đề, khái niệm trong các chủ đề đã được học - IaC - Infrastructure as Code:

- Khi cập nhật 1/nhiều dịch vụ khác nhau trên 1 môi trường microservices, việc này thường được thực hiện cùng một lúc nhằm đảm bảo thời gian downtime ở mức thấp nhất có thể. Tuy nhiên, update nhiều service 1 lúc thường khó triển khai tất cả các code cùng một lúc nếu không có tự động hóa. Vì thế mà nhiều công cụ tự động hóa ra đời như Terraform, Ansible và Bicept. Tuy nhiên, họ thường chuyên về các lĩnh vực khác nhau. Ở đây, em sẽ thảo luận về dùng IaC (infrastructure as code) tool phổ biến nhất, Terraform.

- Có rất nhiều công cụ giúp chúng ta deploy các cơ sở hạ tầng bằng 1 command (declarative) thay vì phải login từng máy một và manage, phải định cấu hình VM và máy chủ (mạng, lưu trữ, v.v.), mỗi máy chủ yêu cầu một hình ảnh được triển khai, vá hệ điều hành, sau đó cài đặt/định cấu hình ứng dụng, database (imperative).
- Terraform là công cụ IaC được sử dụng rộng rãi nhất hiện nay, do HashiCorp phát triển và là ngôn ngữ được các public cloud support nhiều nhất
  - AWS: https://registry.terraform.io/namespaces/terraform-aws-modules
  - Azure: https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs
  - GCP: https://registry.terraform.io/providers/hashicorp/google/latest/docs
  - Oracle: https://registry.terraform.io/providers/hashicorp/oraclepaas/latest/docs
- Terraform sử dụng ngôn ngữ HashiCorp Configuration Language (HCL) để mô tả cơ sở hạ tầng. HCL là ngôn ngữ khá đơn giản, dễ đọc và học.
  - Giả sử bạn muốn tạo một máy chủ ảo trên AWS EC2. Bạn có thể sử dụng Terraform để mô tả máy chủ ảo với những required variables như sau:
    - `main.tf`

```
resource "aws_instance" "demo_web_server5" {
  ami           = "ami-id" # AMI's ID
  instance_type = "t2.micro"
  tags = {
        Name = "Web Server"
  }
}
```

- Terraform dùng modules như Ansible nên sẽ rất phù hợp với cross-cloud deployment với mỗi service trên một cloud khác nhau và deploy thành micro-services. Các micro-service có thể communicate với nhau thông qua các variables trên terraform's state.

```
├── main.tf # File chính chứa cấu hình cơ sở hạ tầng
├── variables.tf # File chứa các variable
├── outputs.tf # File chứa các output trả về giá trị sau khi chạy Terraform
├── terraform.tfvars # File chứa giá trị cụ thể cho các biến
├── modules/ # Thư mục chứa các custom modules
│       └── my_module.tf # File định nghĩa một module
└── providers.tf # File cấu hình các nhà cung cấp cloud (optional)
```

  - **main.tf:** Đây là file chính chứa cấu hình cơ sở hạ tầng, sử dụng các resource blocks để định nghĩa các tài nguyên mong muốn (ví dụ: máy chủ ảo, bucket lưu trữ, database, v.v.). Resource blocks có thể được nested hoặc defined bằng 1 variable khác và được dùng nhiều lần trên nhiều parents như 1 function.

    - **variables.tf:** File này giúp định nghĩa các biến (variable) để cấu hình được linh hoạt hơn, các thể sử dụng lại các biến này trong nhiều resource blocks khác nhau của nhiều provider khác nhau. Ví dụ như variable IP của các services, secrets, network name, subnet cidr block, ...
    - **outputs.tf:** File này dùng để khai báo các output trả về giá trị sau khi chạy Terraform (ví dụ: public IP, tên, internal IP của máy chủ ảo).
    - **terraform.tfvars:** File này chứa các giá trị cụ thể cho các biến được định nghĩa trong variables.tf. Để dễ dàng thay đổi cấu hình mà không cần sửa đổi code trong main.tf hoặc variables.tf.
    - **modules/:** Thư mục này (tùy chọn) chứa các modules. Modules là các đơn vị cấu hình riêng biệt có thể tái sử dụng trong nhiều project khác nhau.
- Để quản lý các state của infrastructure, Terraform tạo 1 file là file state để lưa state của các infrastructure, nhưng file state này trở nên rất phức tạp và dài khi được dùng cho multi-cloud deployment.
- Ưu điểm của terraform là terraform chạy các command để tạo ra các máy chủ ảo bằng cách parallel. Vì các command ấy sẽ được các public cloud dùng để tạo nên các infrastructures nên terraform có thể tạo các insfrastructure mà không bị lock 1 các parallel.