Docker thì e tìm hiểu anh mấy cái
- build image
- Start/Stop Container
- Create custom network
- Xóa image; Dọn cache và các tài nguyên không còn sử dụng
- Cài đặt Docker Compose

FastAPI
- data validation
- concurrency and async
- gunicorn with fastapi
- api design
+ https://linked.art/api/1.0/principles/
+ https://www.mulesoft.com/api-university/four-principles-designing-effective-apis
+ http status code (đọc các code cơ bản ví dụ: 200, 400, 401, 499, 500..)
- Oauth2: https://fastapi.tiangolo.com/tutorial/security/oauth2-jwt/
- CORS

I. DOCKER
1. Thành phần
   - Docker file: bản thiết kế để tạo docker image
   - Image: tạo ra được nhiều docker container
   - Container: mỗi docker container có thể chạy riêng biệt
2. Một số lệnh cơ bản
   - docker search
   - Docker pull
     -> cả 2 above làm việc với docker hub - chỗ submit các image lên để share với nhau
   - Docker build
     muốn build đc 1 image thì phải có docker file, ví du:
     # Dựa trên image cơ bản nào
     FROM python:3

     # Khai báo thư mục làm việc
     WORKDIR
     /Volumes/Data/00.Source/Python/MiAI_Docker_Sample

 # Copy toàn bộ file mã nguồn và các file khác vào image
COPY . .

 # Cài đặt Flask
RUN pip install -r setup.txt (file chứa tất cả nhưng gì cần cài đặt)

 # Thực hiện lệnh chạy
CMD ["python","./example03.py"]

Note: quy tắc đặt tên image để tránh trùng lặp
my_name/image_name

Purpose	**Command**
docker build: build image	**docker build -t image_name . (dấu cách + dấu chấm: chỉ tham số ở thư mục hiện tại)**
xem danh sách images	**docker images**
start image thành container	**docker run --name example01 image_name**
xóa container	**docker container rm container_name**
liên kết cổng 8888 với cổng 5000 trong container ở localhost	**docker run –name example-02 -p 8888:5000 image_name**
container A có thể đọc được từ container B -> phải cho A, B chạy cùng 1 network	**docker network list** **docker network create network_name** **docker run --name A --net network_name -e A_ROOT_PASSWORD=root -d image_name** **docker run --name B --net network_name -p 8000:5000 image_name**
Start or stop an existing container:	**docker start|stop <container_name> (or <container-id>)**
Delete an Image	**docker rmi <image_name>**
Remove all unused images	**docker image prune**

create a custom network	**docker network create my-network** **docker run --name container1 --network my-network image1**
Dọn cache và các tài nguyên không còn sử dụng	**docker system prune -a** **(-a chọn all)**

lệnh: docker build -t image_name . (dấu cách + dấu chấm: chỉ tham số ở thư mục hiện tại)

3. Một số khái niệm

- **Cache**

  Thường đề cập đến các layers khác nhau tạo nên docker image. Khi bạn xây dựng docker image, mỗi instruction trong Dockerfile sẽ tạo một layer trong image. Docker lưu trữ các layers vào bộ đệm để tối ưu hóa các bản dựng tiếp theo.

  Ví dụ: nếu bạn có Dockerfile với nhiều instructions và bạn xây dựng lại image sau khi chỉ sửa đổi một instruction, Docker sẽ sử dụng lại các layers được lưu trong bộ nhớ đệm cho đến thời điểm có instruction đã sửa đổi. Điều này tăng tốc quá trình xây dựng vì Docker không cần tạo lại các layers không thay đổi.

- **Docker Compose**

  Docker Compose là một công cụ giúp định nghĩa và chạy nhiều container Docker trong một ứng dụng. Nó giúp đơn giản hóa việc quản lý các ứng dụng phức tạp bao gồm nhiều container.

  Lợi ích của Docker Compose:

  Dễ sử dụng: Docker Compose sử dụng một file YAML duy nhất để định nghĩa tất cả các container trong ứng dụng của bạn. Điều này giúp việc quản lý các container trở nên dễ dàng hơn nhiều so với việc sử dụng các lệnh Docker thủ công.
  Tự động hóa: Docker Compose có thể tự động hóa việc tạo, khởi động và dừng các container. Điều này giúp bạn tiết kiệm thời gian và công sức.
  Có thể tái tạo: Docker Compose có thể tạo lại môi trường ứng dụng của bạn trên bất kỳ máy nào có cài đặt Docker. Điều này giúp việc cộng tác và triển khai ứng dụng trở nên dễ dàng hơn.

Cách dùng

1. tạo file tên docker-compose.yml
2. Trong docker-compose.yml phải định nghĩa 4 thành phần
    - version: bắt buộc có
    - services: bắt buộc có
    - networks: optional
    - volume: optional
3. cmd

Purpose	Command
run docker-compose file	docker-compose up docker-compose up -d (chạy đc các command khác mà k phải chờ)
kiểm tra các container	docker container ps hoặc docker-compose ps
dừng	docker-compose stop
check lỗi	docker-compose config hoặc docker-compose config -q
khởi động lại	docker-compose start
kiểm tra volume	docker volume ls
	docker volume inspect <tên>
kiểm tra process của các container đang chạy	docker-compose top

-

II. FASTAPI

1. Định nghĩa

   API: là một giao diện lập trình ứng dụng, là một tập hợp các quy tắc và định nghĩa cho phép các phần mềm khác tương tác với nhau

2.

FastAPI: Thông tin chi tiết về các tiêu chí bạn quan tâm

1. Xác thực dữ liệu (Data Validation):

FastAPI cung cấp các công cụ mạnh mẽ để xác thực dữ liệu đầu vào cho API của bạn. Sử dụng các mô hình dữ liệu Pydantic, bạn có thể:

- **Định nghĩa cấu trúc dữ liệu:** Xác định các trường, kiểu dữ liệu và ràng buộc cho dữ liệu đầu vào.
- **Tự động xác thực:** FastAPI sẽ tự động kiểm tra dữ liệu đầu vào dựa trên mô hình Pydantic của bạn.
- **Báo cáo lỗi:** FastAPI cung cấp thông tin lỗi chi tiết khi dữ liệu đầu vào không hợp lệ.
- **Hỗ trợ nhiều định dạng:** Xác thực dữ liệu JSON, XML, Form và hơn thế nữa.

Ví dụ:

Python

```python
from pydantic import BaseModel

class UserIn(BaseModel):
    name: str
    email: str
    password: str

@app.post("/users")
def create_user(user: UserIn):
    # Xác thực dữ liệu đầu vào đã được thực hiện tự động
    # ...
```

2. Đồng thời và Bất đồng bộ (Concurrency and Async):

FastAPI được xây dựng dựa trên Starlette, hỗ trợ mô hình lập trình phi đồng bộ (asynchronous) ngay lập tức. Điều này mang lại nhiều lợi ích:

- **Cải thiện hiệu suất:** Xử lý nhiều yêu cầu đồng thời hiệu quả hơn.
- **Tăng khả năng mở rộng:** Hỗ trợ lượng truy cập lớn mà không ảnh hưởng đến hiệu suất.
- **Giảm sử dụng CPU:** Giải phóng tài nguyên CPU để thực hiện các tác vụ khác.

FastAPI cung cấp các công cụ để sử dụng mô hình lập trình phi đồng bộ dễ dàng:

- **Async endpoints:** Sử dụng từ khóa `async` để định nghĩa điểm cuối phi đồng bộ.
- **AsyncIO:** Tích hợp liền mạch với thư viện `asyncio` của Python.
- **Background tasks:** Thực hiện các tác vụ tốn thời gian trong nền mà không chặn yêu cầu.

Ví dụ:

Python

```
import asyncio

@app.get("/users/{user_id}")
async def get_user(user_id: int):
    # Tìm kiếm người dùng trong cơ sở dữ liệu (có thể là tác vụ
tốn thời gian)
    user = await get_user_from_db(user_id)
    # ...
```

3. Gunicorn với FastAPI (Gunicorn with FastAPI):

Gunicorn là một máy chủ web WSGI mạnh mẽ và phổ biến. FastAPI được thiết kế để hoạt động trơn tru với Gunicorn.

Cách triển khai FastAPI với Gunicorn:

1. Cài đặt Gunicorn: `pip install gunicorn`
2. Khởi chạy Gunicorn: `gunicorn --bind 0.0.0.0:8000 main:app`

Lưu ý:

- Thay thế `main:app` bằng tên module và tên ứng dụng FastAPI của bạn.
- Bạn có thể cấu hình Gunicorn thêm bằng các tùy chọn dòng lệnh.

4. Thiết kế API (API Design):

FastAPI khuyến khích các nguyên tắc thiết kế API RESTful tốt nhất, bao gồm:

- **Tài nguyên (Resources):** Mỗi chức năng được biểu thị bằng một tài nguyên.
- **URL rõ ràng:** URL phản ánh cấu trúc tài nguyên.
- **Phương thức HTTP:** Sử dụng các phương thức HTTP phù hợp (GET, POST, PUT, DELETE) cho mỗi hành động.
- **Mã trạng thái HTTP:** Trả về mã trạng thái HTTP phù hợp để phản ánh kết quả yêu cầu.

- **Tài liệu API:** Cung cấp tài liệu API rõ ràng và đầy đủ thông tin.

Công cụ hỗ trợ thiết kế API:

- **Swagger UI:** Tự động tạo tài liệu API tương tác.
- **APACHE Falcon:** Khung API RESTful với các tính năng thiết kế mạnh mẽ.
- **OpenAPI Specification:** Tiêu chuẩn mô tả API RESTful.

Lưu ý:

- Thiết kế API tốt giúp dễ sử dụng, dễ hiểu và dễ bảo trì API của bạn.
- Hãy dành thời gian để thiết kế API cẩn thận trước khi bắt đầu phát triển.

Tóm lại:

FastAPI là một framework web Python mạnh mẽ và linh hoạt, cung cấp nhiều tính năng hữu ích cho việc xây dựng API hiệu quả, dễ sử dụng và có thể mở rộng. Với các công cụ xác thực dữ liệu, lập trình phi đồng bộ, triển khai Gunicorn và hỗ trợ thiết kế API, FastAPI giúp bạn tạo ra các

III. SERVER