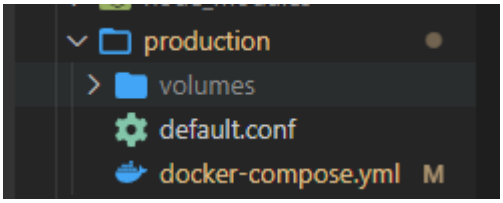
BÁO CÁO TIẾN ĐỘ TUẦN 12:

Trong tuần 11 và 12, em cố gắng hoàn thiện việc build dự án với docker-compose, tiếp đó là deploy dự án thử lên AWS

Dưới đây là những gì em đã thực hiện được

Frontend

Bên trong thư mục src, tạo thư mục production để lưu file docker-compose.yml (file chính chạy dự án) cũng như file default.conf để confix cho nginx



Bên trong thư mục em tạo thêm file dockerfile và docker-compose để build model của dự án

Để chạy dự án frontend, em sẽ dùng máy chủ web nginx, dự án được chạy trên cổng 80. Nginx sẽ đọc file dist được build từ dự án frontend để hiển thị, và sử dụng các confix tại file default.conf

```
nginx:
  image: nginx:latest
  ports:
    - 80:80
  volumes:
    - ../frontend/dist:/usr/share/nginx/html
    - ./default.conf:/etc/nginx/conf.d/default.conf
  networks:
    - app-network
```

Dưới đây là file default.conf để cấu hình cho nginx, cấu hình reverse proxy cho phép api “http://model:8000” từ fastapi bên trong container có thể gọi đến

```
location /api {
  proxy_pass http://model:8000;
  proxy_set_header Host $host;
  proxy_set_header X-Real-IP $remote_addr;
  proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
  proxy_set_header X-NginX-Proxy true;
}
```

Tiếp đến là thay đổi các api gọi đến từ frontend

thay vì gọi đến localhost:8000 như ban đầu thì giờ sẽ gọi đến địa chỉ api…

```
try {
  const res = await axios.get(`api/get_subjects/${stu_id}`);
```

Model + Fastapi:

Fastapi:

Phần này em có thay đổi file requirement.txt là tên các thư mục cần thiết để có thể chạy model,tạo Dockerfile để buld model với fastapi, sau khi build xong dự án sẽ chạy trên cổng 8000

```
FROM python:3.13.3

WORKDIR /chatbotAi/model

COPY requirements.txt . ./

RUN pip install --no-cache-dir -r requirements.txt

COPY . .

EXPOSE 8000
CMD ["uvicorn", "scripts.main:app", "--host", "0.0.0.0", "--port", "8000"]
```

Ollama model:

```
ollama:
  image: ollama/ollama
  ports:
    - 11434:11434
  networks:
    - app-network
```

Về phía ollama bước đầu em sẽ pull image ollama về trước sau đó sẽ tiến hành pull thủ công mô hình “bge-m3:567m” , để kết nối đến ollama sẽ kết nối qua url: “http://ollama:11434”(tên service + port)

Milvus:

```
▷ Run Service
etcd: ···

▷ Run Service
minio: ···

▷ Run Service
standalone: ···
```

Với service chính là standalone, và container_name là milvus-standalone . Để kết nối với milvus sẽ kết nối qua url:"http://milvus-standalone:19530"

```yaml
standalone:
  container_name: milvus-standalone
  image: milvusdb/milvus:v2.5.10
  command: ["milvus", "run", "standalone"]
  security_opt:
    - seccomp:unconfined
  environment:
    ETCD_ENDPOINTS: etcd:2379
    MINIO_ADDRESS: minio:9000
```

Attu:

Em có pull thêm attu giúp review các collection được lưu trong milvus

```yaml
  attu:
    image: zilliz/attu
    container_name: attu
    ports:
      - "3000:3000"
    depends_on:
      - standalone
    networks:
      - app-network
    restart: always
```

Cấu hình mạng:

Ở đây em cấu hình tất cả container đều chung 1 mạng là app-network, giúp các container có thể giao tiếp với nhau cũng như có thể ra vào mạng

```yaml
networks:
  app-network:
    driver: bridge
```

Các bước chạy dự án:

B1:Đảm bảo đã cài đặt docker, docker-destop và wsl

B2.Build dự án frontend

Bước đầu di chuyển vào thư mục frontend:

cd src/frontend

Chạy lệnh sau để build

npm run build để tiến hành build dự án frontend

B3.: Build dự án

Tiếp theo cd ra thư mục gốc:

cd ../../

Di chuyển vào thư mục production:

cd production

Chạy lệnh sau để build dự án

docker compose -p chatbot up -d

Khi build xong, tiến hành pull mô hình bge-m3:567m :

Di chuyển vào container ollama:

docker exec -it chatbot-ollama-1 /bin/bash

Chạy lênh sau để pull model:

ollama pull bge-m3:567m

Deploy dự án trên AWS

Ở đây em dùng console online của AWS để thao tác, clone dự án từ github, sau đó cài đặt như trên, tuy nhiên có 1 vài thay đổi nhỏ sau:

B1.:Swap ram ảo:

Do EC2 bản trial có 1gb ram không đủ để chạy dự án, dó đó em tạo swap ram cho EC2:

Tạo file swap 5GB:

sudo fallocate -l 5G /swapfile

Đặt quyền:

sudo chmod 600 /swapfile

Định dạng file đó thành swap:

sudo mkswap /swapfile

Kích hoạt swap:

sudo swapon /swapfile

Thêm vào /etc/fstab để tự động bật lại sau reboot:

echo '/swapfile none swap sw 0 0' | sudo tee -a /etc/fstab

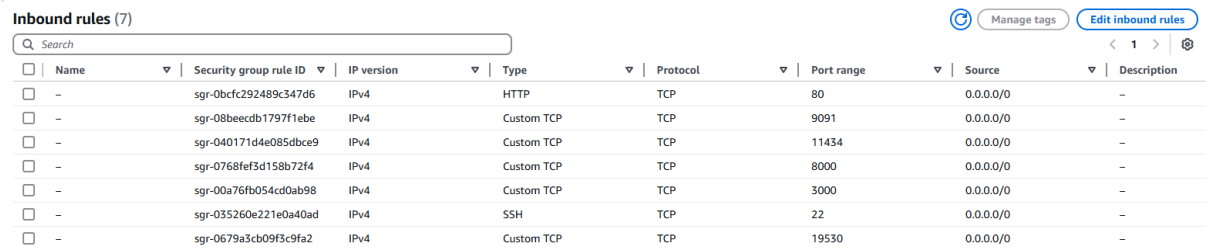
Phần build model khá lâu do đó em push image của phần model + fastapi lên dockerhub, sau đó pull image về với câu lệnh:

docker pull hminh729/chatbot-model

Do pull image về do đó em sửa trong file docker-compose phần model, sẽ không build lại nữa mà lấy từ image hminh729/chatbot-model:latest

```
services:
  model:
    image: hminh729/chatbot-model:latest
    ports:
      - 8000:8000
    expose:
      - 8000
    networks:
      - app-network
    # environment:
    #   - MODEL_NAME=your_model_name
    #   - MODEL_VERSION=1.0.0
```

Tiến hành mở cổng để có thể giao tiếp ra bên ngoài



Inbound rules (7)

Manage tags | Edit inbound rules

Search

1

Name	Security group rule ID	IP version	Type	Protocol	Port range	Source	Description
-	sgr-0bcfc292489c347d6	IPv4	HTTP	TCP	80	0.0.0.0/0	-
-	sgr-08beecdb1797f1ebe	IPv4	Custom TCP	TCP	9091	0.0.0.0/0	-
-	sgr-040171d4e085dbce9	IPv4	Custom TCP	TCP	11434	0.0.0.0/0	-
-	sgr-0768fef3d158b72f4	IPv4	Custom TCP	TCP	8000	0.0.0.0/0	-
-	sgr-00a76fb054cd0ab98	IPv4	Custom TCP	TCP	3000	0.0.0.0/0	-
-	sgr-035260e221e0a40ad	IPv4	SSH	TCP	22	0.0.0.0/0	-
-	sgr-0679a3cb09f3c9fa2	IPv4	Custom TCP	TCP	19530	0.0.0.0/0	-

Dự án chạy trên địa chỉ:

http://52.15.84.235/

Đã đầy đủ tính năng tuy nhiên còn chậm ạ

Tuần 12 này em sẽ sửa lại cấu dự án, loại bỏ các file không cần thiết, comment code, cũng như hoàn thành báo cáo và slide ạ