Flask-api-template

API Server 구축을 위해 모델 모듈 수정만으로 서비스 가능한 flask api template 제공

Directory

```
flask_api/
├── .env
├── Dockerfile
├── README.md
├── app
│   ├── __init__.py
│   ├── models.py
│   └── templates.py
├── logs/
├── requirements.txt
├── run.py
└── wsgi.py
```

filep	exp
`.env`	서버에서 사용될 환경변수 파라미터 파일
`Dockerfile`	docker 실행 시 사용될 Dockerfile
`app/__init__.py`	flask app initiation {logging, blueprint, CORS}
`app/models.py`	model load & prediction method module
`app/templates.py`	api template module
`logs/`	logs directory
`requirements.txt`	packages list
`wsgi.py`	excute file for the gunicorn

Debug mode

1. 가상환경 설정 후 필수 라이브러리 설치

```
pip install -r requirements.txt
```

2. flask app 실행

```
 flask run --host 0.0.0.0 --port=5000 --debug --no-reload
```

※ 개발 시 유의

빠른 수정을 위하여 `--no-reload` 옵션을 삭제 가능 하나 Werkzeug reloader는 코드가 변경될 때마다 해당 프로세스를 다시 시작할 수 있도록 하위 프로세스를 생성. 따라서 두개의 프로세스가 생성되는데, 하위 프로세스에서 모델 로드 시 GPU 메모리를 초과할 수 있으므로 모델 크기를 고려하여 `--no-reload` 옵션을 사용할 것

Build and Run docker

Directory : `flask_api_templates/.`

Build

※ 각 호스트 환경에 알맞은 도커 이미지를 위해 Dockerfile 수정 필요 [link]

> **docker build -t [name] [Dockerfile]**
>
> - Dockerfile : Dockerfile path
>
> ```
> docker build -t flask_image .
> ```

Run

※ nvidia-docker 설치 [link]

> **docker run -it --rm -d -p [host:container] --gpus [all / device] --env-file [env file] -v [host : container] [image]**
>
> - -d : 도커 백그라운드 실행
>
>   **option을 지워 포그라운드에서 Error check**
> - p : 호스트 포트와 컨테이너 내 포트 매핑
>
>   host: 호스트에서 사용할 포트 (외부 포트)
>
>   continer: container 내부 연결 포트* (env file의 CONTAINER_PORT 통일)
> - gpus : GPU 할당
>
>   all : 전체 GPU
>
>   '"device=0,1, 2, ..., n"': n번째 GPU
> - enf-file : 환경변수 설정 파일 [link]
> - v : 호스트와 컨테이너 간 마운트
>
>   host : 마운트할 호스트 경로
>
>   container : 마운트할 컨테이너 경로
> - image : build한 image
>
> ```
> docker run -it --rm -d -p 5000:5000 --gpus '"device=0"' --env-file .env -v
> ./logs:/app/logs --name flaskAPI flask_image
> ```

kill

```
docker kill flaskAPI
```

Test

after modify the URL variable in each cmd ...

1. curl

```
# bash

$ curl -X POST -H "Content-Type: application/json" -d '{"text":"KETI is a specialized production technology research institute established in 1988 to promote Korea electronics and information communication industry. KETI conducts technology research and development, industry support, international standardization, and technology talent cultivation in the field of information and communication to serve as a central player in implementing and advancing the nation ICT policies. Its main research areas include artificial intelligence, the Internet of Things (IoT), big data, cloud computing, and more."}' http://0.0.0.0:5000/prediction
```

```
KETI is a specialized production technology research institute established in 1988. it conducts technology research and development, industry support, international standardization, and technology talent cultivation.
```

2. requests

```
#test.py

import requests

api_list=requests.get('http://0.0.0.0:5000/apilist').text
print(api_list)

doc=requests.get('http://0.0.0.0:5000/help/prediction').text
print(doc)

response=requests.post('http://0.0.0.0:5000/prediction',json={"text":"KETI is a specialized production technology research institute established in 1988 to promote Korea's electronics and information communication industry. KETI conducts technology research and development, industry support, international standardization, and technology talent cultivation in the field of information and communication to serve as a central player in implementing and advancing the nation's ICT policies. Its main research areas include artificial intelligence, the Internet of Things (IoT), big data, cloud computing, and more."}).text
print(response)
```

```
# python3 test.py

["/prediction","/apilist","/help/<string:endpoint_name>"]

Args:
    json : {'text': Org Text(str)}

Returns:
    str : Summarization Text

KETI is a specialized production technology research institute established in 1988.
it conducts technology research and development, industry support, international
standardization, and technology talent cultivation.
```

---

nvidia-docker

1. 저장소 및 GPG 키 설정

```
$ distribution=$(. /etc/os-release;echo $ID$VERSION_ID) \
   && curl -s -L https://nvidia.github.io/nvidia-docker/gpgkey | sudo apt-key add -
\
   && curl -s -L https://nvidia.github.io/nvidia-docker/$distribution/nvidia-
docker.list | sudo tee /etc/apt/sources.list.d/nvidia-docker.list
```

2. Nvidia-docker install[

```
$ sudo apt-get update
$ sudo apt-get install -y nvidia-docker2
```

3. docker service restart

```
$ sudo systemctl restart docker
```

.env

```
#.env

CONTAINER_PORT=5000     # 컨테이너 내에서 동작할 gunicorn port*
TIMEOUT=600             # gunicorn timeout
LOG_FILE=logs/app.log   # log file
WORKERS=3               # 실행시킬 gunicorn process 수
```

※ WORKERS : 프로세스의 개수는 cpu core 및 GPU memory, utilize를 고려하여 지정

Dockerfile

호스트 환경에 알맞은 cuda version의 pytorch/pytorch base image로 변경 [homepage]

```
FROM pytorch/pytorch:2.0.0-cuda11.7-cudnn8-devel # base image
...
```

CORS

특정 주소, 도메인 및 포트 설정 [homepage]

```
#__init__.py

CORS(app)
#CORS(app, resources={r'*': {'origins': '*'}})
#CORS(app, resources={r'/*': {'origins': 'http://0.0.0.0:5000}})
```