



0



@okubo1504 2020年05月17日に投稿 311 views

編集する



(備忘録) Docker ComposeでTensorFlow + Flask + Nginx環境構築時のメモ

Python nginx Flask docker-compose TensorFlow

はじめに

自分の備忘録用です😓

Docker ComposeでTensorFlow + Flask + Nginxの環境を作る時のメモです。

ちょうど、TensorFlowを使ったアプリの備忘録

を作っていました、切り出して整理しようと思った次第です。。

この手順を実行すれば、TensorFlow使ったWeb APIが動くはず😓

自分用に作った記事なので、分かりにくい点や情報、技術が古いかもしれませんがご了承ください🙏

参考資料

この記事を作るにあたって参考にさせて頂きました🙏

- [docker-composeでunicorn+nginx+flaskを動かしてみた話](#)
- [DockerでDjangoの開発環境を再構築！！！！](#)
- [Docker入門（第六回）～Docker Compose～](#)
- [初心者向けdocker-composeコマンド逆引き](#)
- [docker-compose コマンドまとめ](#)

- [Dockerコマンド よく使うやつ](#)
- [Docker一括削除コマンドまとめ](#)

環境 ※以下のVerでなくても動くと思いますが、古いのでご注意下さい🙇

Ubuntuバージョン

```
$ cat /etc/os-release
NAME="Ubuntu"
VERSION="18.04.4 LTS (Bionic Beaver)"
ID=ubuntu
ID_LIKE=debian
PRETTY_NAME="Ubuntu 18.04.4 LTS"
VERSION_ID="18.04"
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-policy"
VERSION_CODENAME=bionic
UBUNTU_CODENAME=bionic
```

Dockerバージョン

```
$ docker version
Client: Docker Engine - Community
 Version:           19.03.8
 API version:       1.40
 Go version:        go1.12.17
 Git commit:        afacb8b7f0
 Built:             Wed Mar 11 01:25:46 2020
 OS/Arch:           linux/amd64
 Experimental:      false

Server: Docker Engine - Community
 Engine:
  Version:           19.03.8
  API version:       1.40 (minimum version 1.12)
  Go version:        go1.12.17
  Git commit:        afacb8b7f0
```

```
Built: Wed Mar 11 01:24:19 2020
OS/Arch: linux/amd64
Experimental: false
containerd:
  Version: 1.2.13
  GitCommit: 7ad184331fa3e55e52b890ea95e65ba581ae3429
runc:
  Version: 1.0.0-rc10
  GitCommit: dc9208a3303feef5b3839f4323d9beb36df0a9dd
docker-init:
  Version: 0.18.0
  GitCommit: fec3683
```

Docker-Composeバージョン

```
$ docker-compose version
docker-compose version 1.25.5, build unknown
docker-py version: 4.2.0
CPython version: 3.7.4
OpenSSL version: OpenSSL 1.1.1c 28 May 2019
```

※なぜかbuild unknown。時間掛かりそうだったので諦めました😓

ディレクトリ構成

適当に作っています ※凝視したらダメです🙄

ゴミファイルが多いですが、Githubに置いてあります。

ソース

ディレクトリ構成

```
dk_tensor_fw
├─ app_tensor
│   ├── Dockerfile
│   ├── exewhatMusic.py
│   ├── inputFile
│   │   └─ ans_studyInput_fork.txt
│   ├── mkdbAndStudy.py
│   ├── requirements.txt
│   ├── studyModel
│   └─ genre-model.hdf5
```

```
|   |   |— genre-tdidf.dic
|   |   |— genre.pickle
|   |— tfidfWithIni.py
|   |— webQueApiRunServer.py
|— docker-compose.yml
|— web_nginx
|   |— Dockerfile
|   |— nginx.conf
```

docker-composeでローカル環境作るのに必要なファイル

docker-compose.yml

```
version: '3'
services:
##### Appサーバ設定 #####
  app_tensor:
    container_name: app_tensor
    # サービス再起動ポリシー
    restart: always
    # ビルドするdockerファイルが格納されたディレクトリ
    build: ./app_tensor
    volumes:
      # マウントするディレクトリ
      - ./app_tensor:/dk_tensor_fw/app_tensor
    ports:
      # ホスト側のポート：コンテナ側のポート
      - 7010:7010
    networks:
      - nginx_network
##### Appサーバ設定 #####

##### Webサーバ設定 #####
  web-nginx:
    container_name: web-nginx
    build: ./web_nginx
    volumes:
      # マウントするディレクトリ
      - ./web_nginx:/dk_tensor_fw/web_nginx
    ports:
      # ホストPCの7020番をコンテナの7020番にポートフォワーディング
      - 7020:7020
    depends_on:
```

```
# 依存関係を指定。web-serverの起動より前にapp-serverを起動するようになる
- app_tensor

networks:
  - nginx_network

##### Webサーバ設定 #####

networks:
  nginx_network:
    driver: bridge
```

※（参考）上記でポート番号を指定していますが、以下のコマンドで確認してます。

（参考）空いているポートの調べ方

```
# 空いているポート調べる（何も表示されなければ空いてる）
netstat -an | grep 7010
```

Dockerfile←Apサーバ側(Gunicorn)

```
FROM ubuntu:18.04
```

```
WORKDIR /dk_tensor_fw/app_tensor
```

```
COPY requirements.txt /dk_tensor_fw/app_tensor
```

```
RUN apt-get -y update \
    && apt-get -y upgrade \
    && apt-get install -y --no-install-recommends locales curl python3-distutils v
    && curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py \
    && python3 get-pip.py \
    && pip install -U pip \
    && localedef -i en_US -c -f UTF-8 -A /usr/share/locale/locale.alias en_US.UTF-
    && apt-get clean \
    && rm -rf /var/lib/apt/lists/* \
    && pip install -r requirements.txt --no-cache-dir
```

```
ENV LANG en_US.utf8
```

```
CMD ["gunicorn", "webQueApiRunServer:app", "-b", "0.0.0.0:7010"]
```

requirements.txt

```
Flask==1.1.0
```

```
gunicorn==19.9.0
```

```
Keras>=2.2.5
```

```

numpy==1.16.4
pandas==0.24.2
pillow>=6.2.0
python-dateutil==2.8.0
pytz==2019.1
PyYAML==5.1.1
requests==2.22.0
scikit-learn==0.21.2
sklearn==0.0
matplotlib==3.1.1
tensorboard>=1.14.0
tensorflow>=1.14.0
mecab-python3==0.996.2

```

以下のpythonソースが機械学習済みのモデルを使ってある事柄を類推し、
Jsonのレスポンスを返すWeb API本体です。実際の類推しているモジュール
(exeWhatMusic)

は外から読み込んでいます😁

webQueApiRunServer.py

```

import flask
import os
import exeWhatMusic

#ポート番号
TM_PORT_NO = 7010

# initialize our Flask application and pre-trained model
app = flask.Flask(__name__)
app.config['JSON_AS_ASCII'] = False # <-- 日本語の文字化け回避

@app.route('/recommend/api/what-music/<how_music>', methods=['GET'])
def get_recom_music(how_music):
    recoMusicInfos = getRecoMusicMoji(how_music)
    return flask.jsonify({'recoMusicInfos': recoMusicInfos})

# オススメの楽曲名を返す
def getRecoMusicMoji(how_music):

    recMusicName, predict_val = exeWhatMusic.check_genre(how_music)

    #JSON作成
    recoMusicInfoJson = 「

```

```

        {
            'id':1,
            'recoMusicMoji':recMusicName,
            'predict_val':predict_val,
            'how_music':how_music
        }
    ]
    return recoMusicInfoJson

if __name__ == "__main__":
    print(" * Flask starting server...")
    app.run(threaded=False, host="0.0.0.0", port=int(os.environ.get("PORT", TM_PORT)))

```

Dockerfile←Webサーバ側(Nginx)

```
FROM nginx:latest
```

```
RUN rm /etc/nginx/conf.d/default.conf
```

```
COPY nginx.conf /etc/nginx/conf.d
```

nginx.conf

```

upstream app_tensor_config {
    # コンテナのサービス名を指定すると名前解決してくれる
    server app_tensor:7010;
}

server {
    listen 7020;
    root /dk_tensor_fw/app_tensor/;
    server_name localhost;

    location / {
        try_files $uri @flask;
    }

    location @flask {
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_redirect off;

        proxy_pass http://app_tensor_config;
    }
}

```

```

    }

    # redirect server error pages to the static page /50x.html
    error_page 500 502 503 504 /50x.html;
    location = /50x.html {
        root /usr/share/nginx/html;
    }

    # 静的ファイルの要求をstaticにルーティング ←使ってませんので不要です。
    location /static/ {
        alias /dk_tensor_fw/app_tensor/satic/;
    }
}

```

出来上がった環境の確認

ビルド&backgroundで起動

```
$ docker-compose up -d --build
```

docker-compose イメージ情報を表示

```
$ docker-compose images
```

Container	Repository	Tag	Image Id	Size
app_tensor	dk_tensor_fw_app_tensor	latest	3b916ea797e0	2.104 GB
web-nginx	dk_tensor_fw_web-nginx	latest	175c2596bb8b	126.8 MB

作り方が悪いのか結構容量大きいような😓

コンテナの一覧表示

```
$ docker-compose ps
```

Name	Command	State	Ports
------	---------	-------	-------


```
app_tensor    gunicorn webQueApiRunServe ...    Up    0.0.0.0:7010->7010/tcp
web-nginx     nginx -g daemon off;             Up    0.0.0.0:7020->7020/tcp, 80/t
```

コンテナに接続 (Apサーバ側)

```
$ docker-compose exec app_tensor /bin/bash
root@ba0ce565430c:/dk_tensor_fw/app_tensor#
```

Apサーバ側のコンテナに入れました。。。

TensorFlowとかKeras入っているか中身を確認

出力結果の表示が長いのでいくつか省きました😓

```
root@ba0ce565430c:/dk_tensor_fw/app_tensor# pip3 list
Package                Version
-----
absl-py                 0.9.0
Flask                   1.1.0
gunicorn                19.9.0
Keras                   2.3.1
Keras-Applications     1.0.8
Keras-Preprocessing     1.1.2
matplotlib              3.1.1
mecab-python3           0.996.2
numpy                   1.16.4
pandas                  0.24.2
Pillow                  7.1.2
pip                     20.1
python-dateutil         2.8.0
pytz                    2019.1
PyYAML                  5.1.1
requests                2.22.0
requests-oauthlib       1.3.0
rsa                     4.0
scikit-learn            0.21.2
six                     1.14.0
sklearn                 0.0
```

```
tensorboard          2.2.1
tensorboard-plugin-wit 1.6.0.post3
tensorflow            2.2.0
tensorflow-estimator 2.2.0
(省略)
```

TensorFlow、Kerasなど一通り入っているようです。。

Webサーバ側のコンテナに接続

```
$ docker-compose exec web-nginx /bin/bash
root@d6971e4dc05c:/#
```

Webサーバ側のコンテナにも入れました。

一応、Webサーバ(Nginx)が起動しているか確認します。

```
root@d6971e4dc05c:/# /etc/init.d/nginx status
[ ok ] nginx is running.
```

Nginxも起動しているようです。

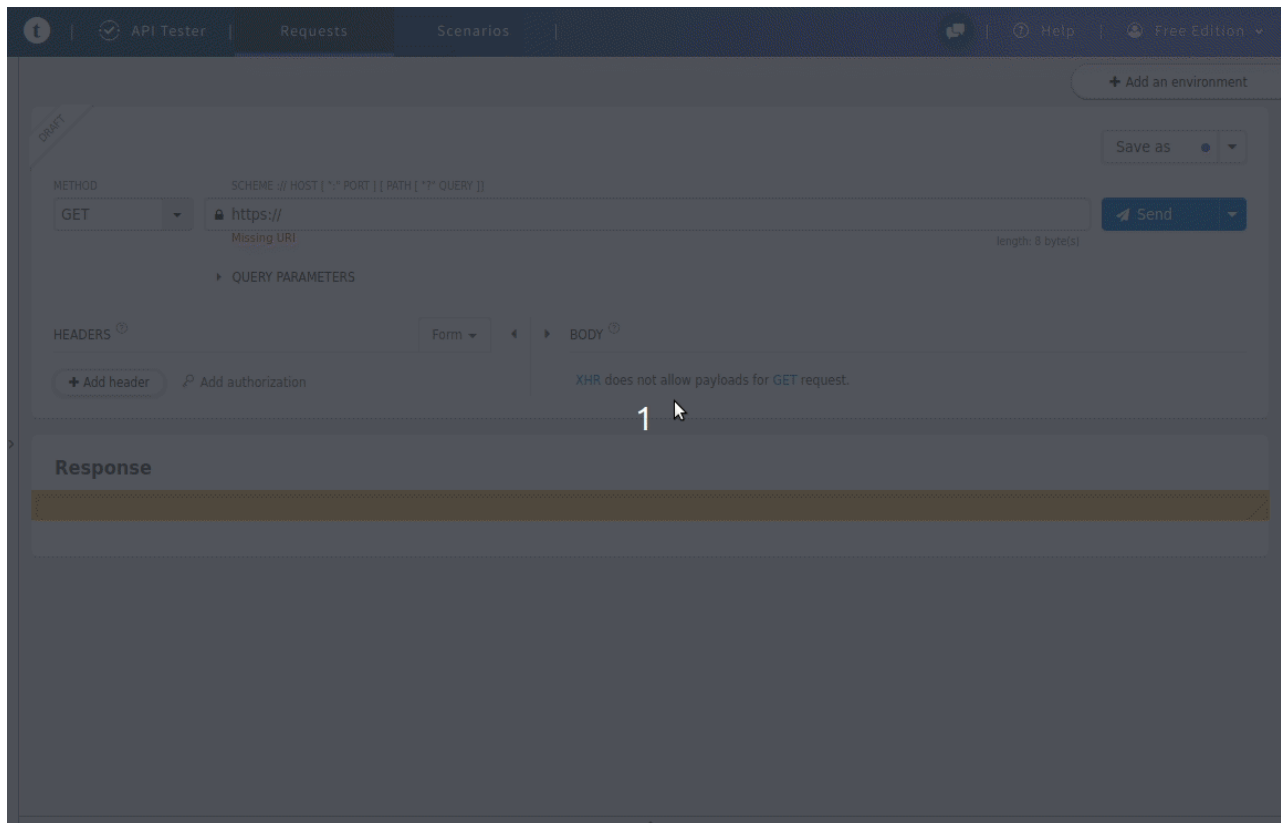
一旦ここまでで実行環境の確認しました。

以下のようにWEB APIが叩ければWEB API側の実行環境できてると思います。。

Web_API実行例

```
http://localhost:7020/recommend/api/what-music/切なくて誰かの幸せ願う歌
```

Web API実行例



ツールは色々あるので何でも良いと思いますが、GIFのようにJSONで返ってきます。

その他のコマンド(備忘です)

※参考資料そのままです。詳細は参考資料等見て下さい🙋

サービス停止

```
$ docker-compose stop
```

サービス開始

```
$ docker-compose start
```

環境をクリーンにしたい時

停止&削除

コンテナ・ネットワーク

```
docker-compose down
```

コンテナ・ネットワーク・イメージ

```
docker-compose down --rmi all
```

コンテナ・ネットワーク・ボリューム

```
docker-compose down -v
```



ストック

LGTM 1



Masayuki Okubo @okubo1504

よろしくお願いします🙏




コメント

この記事にコメントはありません。

投稿する

編集 プレビュー ? 😊

テキストを入力

 画像を選択

0B / 100MB

投稿

How developers code is here.



Qiita

About 利用規約 プライバシー ガイドライン リリース API ご意見 ヘルプ 広告掲載

Increments

[About](#) [採用情報](#) [ブログ](#) [Qiita Team](#) [Qiita Jobs](#) [Qiita Zine](#)

© 2011-2020 Increments Inc.