Docker Flask App

Cycle de Projet :

1ere étape :

- Configurer le DockerFile
- Générer le pyc du python
- Faciliter le logging
- Installation du bibliothèques
- Changer en tant qu'user ce qui n'est pas le root

Dockerfile M

frontapp > Dockerfile > ...

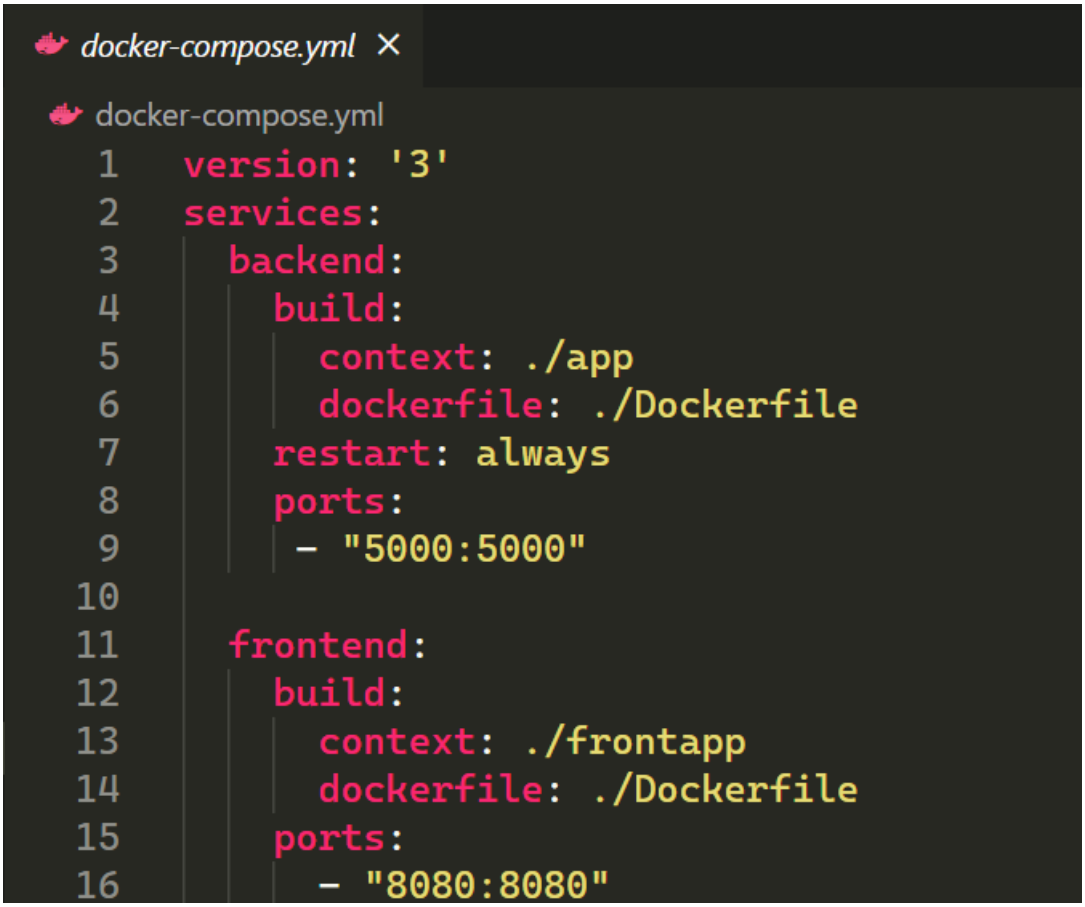
```
1  FROM python:3.8-slim-buster
2
3  EXPOSE 8080
4
5  ENV PYTHONDONTWRITEBYTECODE=1
6
7  ENV PYTHONUNBUFFERED=1
8
9  COPY requirements.txt .
10 RUN python -m pip install -r requirements.txt
11
12 WORKDIR /app
13 COPY . /app
14
15 RUN useradd appuser && chown -R appuser /app
16 USER appuser
17
18 CMD ["python", "app.py"]
19
```

- Configurer .dockerigonre pour ignorer les fichiers unitiles.

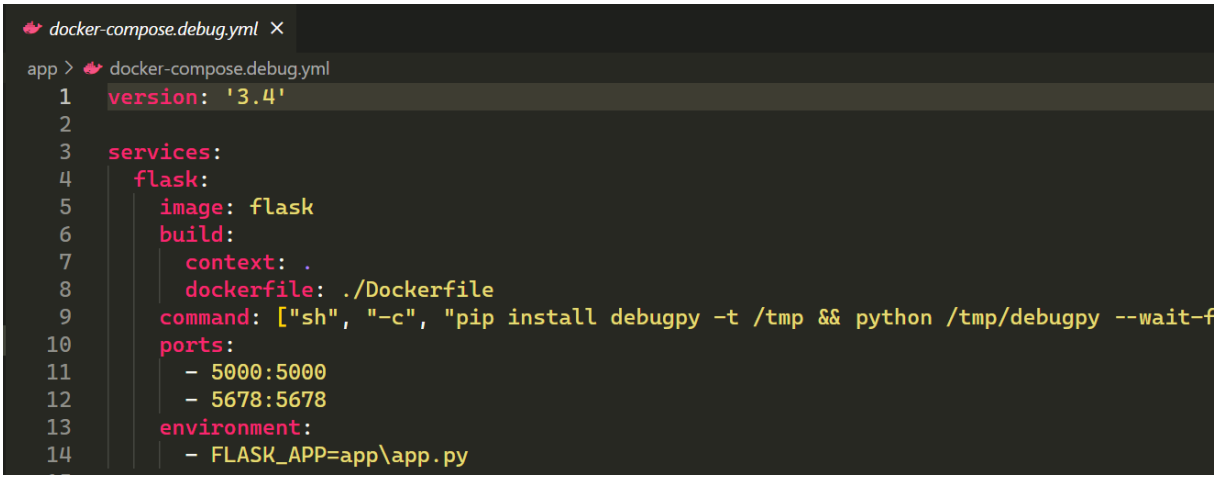
2eme étape :

Configurer le docker compose pour générer les commandes du dockerFile.

Dans lequel j'ai instancier le backend dans un conteneur et le frontend dans un autre l'uns contient le port 5000 et l'autre 8080.

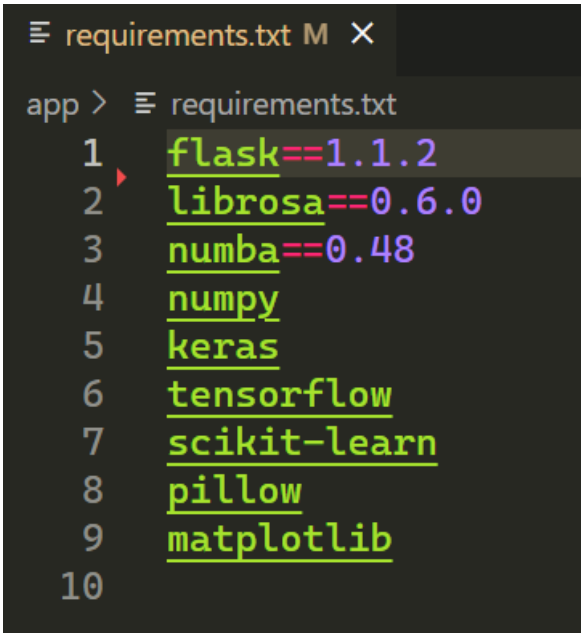


Configurer le dock-compose.debug pour debugger les commandes.



- Configurer le fichier requirements.txt

Pour vous assurer que les dépendances des applications sont transférées de votre environnement virtuel/machine hôte vers votre conteneur



- Développer le unittests.py :

C’est une partie du Machine Learning

unittests.py 1

app > unittests.py > ...

```python
import unittest
import app

BASE_URL = 'http://127.0.0.1:5000/'



class TestFlaskApi(unittest.TestCase):

    def setUp(self):
        self.app = app.app.test_client()
        self.app.testing = True

    def test_svm(self):
        rv = self.app.get(BASE_URL+'uploader')
        self.assertEqual(rv.status, '200 OK')

    def test_vgg(self):

        rv = self.app.get(BASE_URL+'uploadervgg')
        self.assertEqual(rv.status, '200 OK')


if __name__ == '__main__':
    import xmlrunner
    runner = xmlrunner.XMLTestRunner(output='test-reports')
    unittest.main(testRunner=runner)
```

3eme étape :

Développer le backend :

app.py 2

frontapp > app.py > ...

```python
from flask import Flask
from flask import render_template
app=Flask(__name__, template_folder='templates')


@app.route('/upload')
def upload_files():
    return render_template("form.html")


if __name__ == "__main__":
    app.run(host="0.0.0.0", port=8080)
```

4eme étape :

Développer le frontend :

J'ai utilisé une requête 'POST ' dans le chemin est http://localhost:5000/uploadervgg et une autre dans le chemin est http://localhost:5000/uploadervgg

```
    <div class="box">
        <h1>Test VGG</h1>
        <form action = "http://localhost:5000/uploadervgg" method = "POST"
            enctype = "multipart/form-data">
            <input type = "file" name = "file" />
            <input class ="submit"type = "submit"/>
        </form>

        <h1>Test SVM</h1>
        <form action = "http://localhost:5000/uploader" method = "POST"
            enctype = "multipart/form-data">
            <input type = "file" name = "file" />
            <input class ="submit" type = "submit"/>
        </form>
    </div>
    </body>
</html>
```