

DevOPS TP2

Dockerfile :

```
FROM python:3.8

RUN apt update
RUN apt install python3

# set the working directory in the container
WORKDIR C:/Users/zmoha/OneDrive/Bureau/tp2DevOPS

COPY app.py ./

# copy the dependencies file to the working directory
COPY requirements.txt .

# install dependencies
RUN pip install -r requirements.txt

# command to run on container start
CMD [ "python3", "./app.py" ]
```

App.py :

```
import flask
from requests import Request, Session, Response
import os

from flask import request

app = flask.Flask(__name__)
app.config["DEBUG"] = True

@app.route('/', methods=['GET'])
def home():
    lat = request.args["lat"]
    long = request.args["lon"]
    apiKey = os.environ['API_KEY']

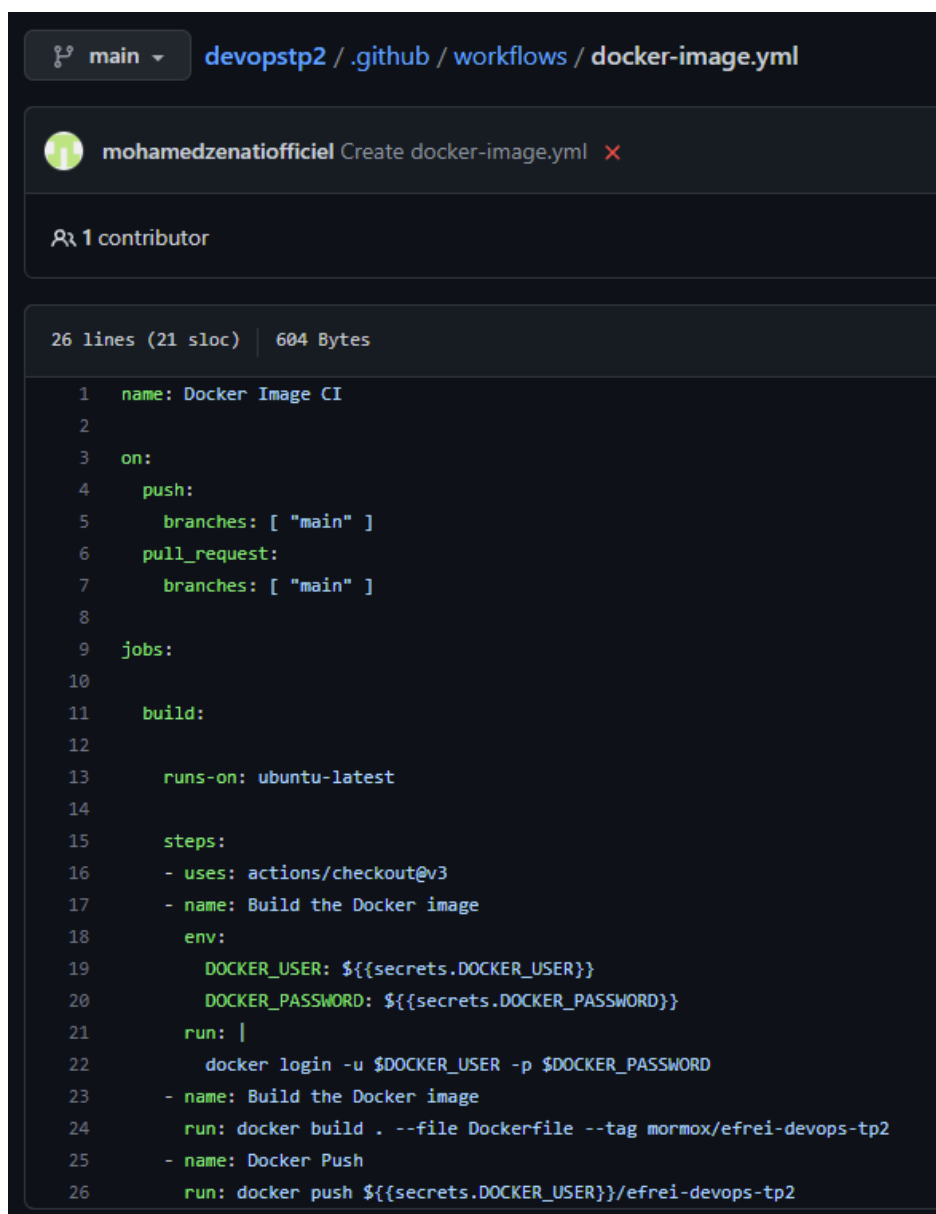
    url = 'http://api.openweathermap.org/data/2.5/weather'
    params = {
        'lat': lat,
        'lon': long,
        'appid': apiKey
```

```
}
session = Session()
requete = Request('GET', url, params=params)
prepped = requete.prepare()
response = session.send(prepped)
return response.json()

app.run(port=8081)
```

Etape 1 : Création du fichier docker-image.yml

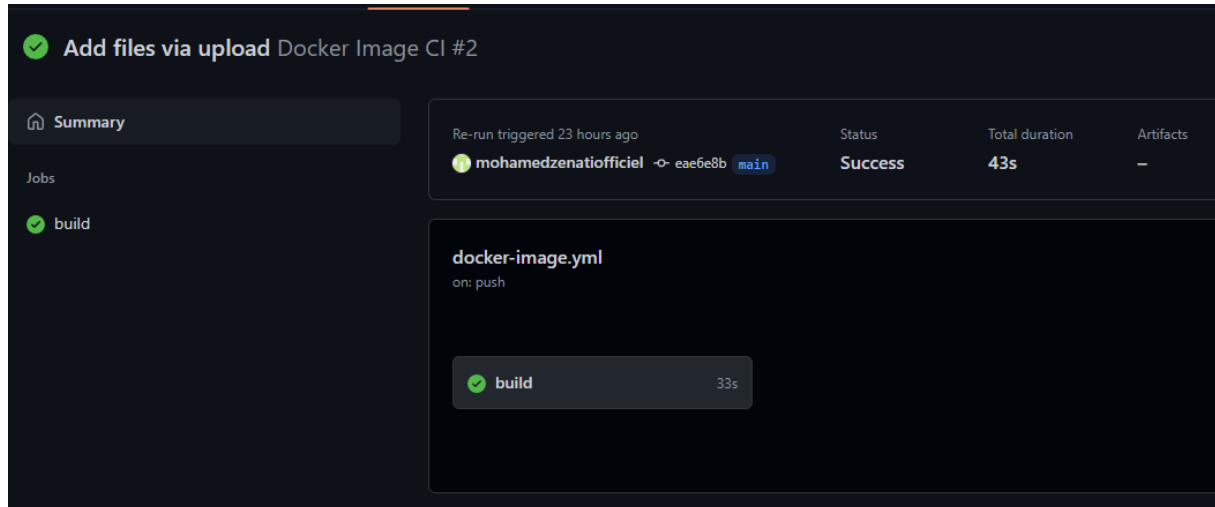
Pour créer ce fichier, il faut aller sur github, créer un repo, aller dans « Actions » -> « New workflow » -> « docker image » puis créer le fichier

The image shows a GitHub Actions workflow file named 'docker-image.yml' in a dark-themed editor. The file is located in the 'devopstp2' repository under the path '.github / workflows / docker-image.yml'. It was created by 'mohamedzenatiofficiel' and has 1 contributor. The file statistics show 26 lines (21 sloc) and 604 Bytes. The workflow is named 'Docker Image CI' and is triggered on a 'push' to the 'main' branch. It runs on 'ubuntu-latest' and contains a 'build' job. The job has two steps: 'Build the Docker image' and 'Docker Push'. The first step uses 'actions/checkout@v3' and sets environment variables for 'DOCKER_USER' and 'DOCKER_PASSWORD' from secrets. The second step runs 'docker build' with the file 'Dockerfile' and tags the image 'mormox/efrei-devops-tp2'. The third step runs 'docker push' to push the image to the repository.


```
1 name: Docker Image CI
2
3 on:
4   push:
5     branches: [ "main" ]
6   pull_request:
7     branches: [ "main" ]
8
9 jobs:
10
11   build:
12
13     runs-on: ubuntu-latest
14
15     steps:
16     - uses: actions/checkout@v3
17     - name: Build the Docker image
18       env:
19         DOCKER_USER: ${secrets.DOCKER_USER}
20         DOCKER_PASSWORD: ${secrets.DOCKER_PASSWORD}
21       run: |
22         docker login -u $DOCKER_USER -p $DOCKER_PASSWORD
23     - name: Build the Docker image
24       run: docker build . --file Dockerfile --tag mormox/efrei-devops-tp2
25     - name: Docker Push
26       run: docker push ${secrets.DOCKER_USER}/efrei-devops-tp2
```

Etape 2 : On ajoute notre fichier .py, notre dockerfile et notre requirement.txt dans le repo créer précédemment.


Une fois ça fait, dans « Actions » on voit qu'une nouvelle action est ajoutée et est entrain d'être build.



The screenshot shows the GitHub Actions interface for a workflow named "Add files via upload Docker Image CI #2". The workflow is in a "Success" status, triggered by a push to the "main" branch. The "build" job is shown as completed with a duration of 33s. The workflow file is named "docker-image.yml" and is triggered on push. A summary table shows the job "build" as successful.

Re-run triggered 23 hours ago	Status	Total duration	Artifacts
 mohamedzenatiofficiel -> eae6e8b main	Success	43s	—

docker-image.yml
on: push

 build 33s

Désormais, à chaque fois qu'un fichier sera ajouter au repo github, un job va être set up, le docker image sera build et push. Tout ça est désormais automatique.

Etape 3 :

```
(base) PS C:\WINDOWS\system32> docker run -p 8081:8084 --network bridge --env API_KEY=240aa650f4db4e154a07d0459c30a347 mormox/efrei-devops-tp2
Unable to find image 'mormox/efrei-devops-tp2:latest' locally
latest: Pulling from mormox/efrei-devops-tp2
e756f3fdd6a3: Already exists
bf168a674899: Already exists
e604223835cc: Already exists
6d5c91c4cd86: Already exists
2cc8d8854262: Already exists
2767dbfeeb87: Already exists
e5a387f746e4: Already exists
54b770283d3a: Already exists
d829066ff47d: Already exists
beb94a6b13a7: Pull complete
2d6fba2cece2: Pull complete
81c07ba9aaac: Pull complete
01e484094c41: Pull complete
36e2e29b793a: Pull complete
9affed275770: Pull complete
Digest: sha256:2b90e532a124ab26bad1f63d6ec88b637829cacfc49066a5f552de4c53904cab
Status: Downloaded newer image for mormox/efrei-devops-tp2:latest
* Serving Flask app 'app' (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Running on http://127.0.0.1:8081 (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 310-043-674
```

Etape 4 : Problème rencontrer lors du curl

```
(base) PS C:\WINDOWS\system32> curl "http://127.0.0.1:8081/?lat=5.902785&lon=102.754175"
curl : La connexion sous-jacente a été fermée : La connexion a été interrompue de manière inattendue.
Au caractère Ligne:1 : 1
+ curl "http://127.0.0.1:8081/?lat=5.902785&lon=102.754175"
~~~~~~
+ CategoryInfo          : InvalidOperation : (System.Net.HttpWebRequest:HttpWebRequest) [Invoke-WebRequest], WebEx
ception
+ FullyQualifiedErrorId : WebCmdletWebResponseException,Microsoft.PowerShell.Commands.InvokeWebRequestCommand

(base) PS C:\WINDOWS\system32> 
```