Conteneurisation d'application avec Docker
Docker compose

Objectifs :

- Créer un service product exposant une API REST avec Flask pour fournir la liste des produits.
- Créer un service website en PHP/Apache qui consomme l'API Flask et affiche les produits dans un navigateur.
- Utiliser Docker et Docker Compose pour orchestrer les deux services et permettre l'accès via le navigateur.

Structure :

```
PS C:\Users\cyrin\docker_compose> tree /F
Structure du dossier pour le volume OSig    Enable Watch
Le numéro de série du volume est A6B3-8183
C:.
    docker-compose.yml

├───product
        api.py
        Dockerfile
        requirements.txt

└───website
        index.php
```

Service 1 : API REST

Créer une API REST simple avec Flask dans Docker, et la rendre accessible depuis mon navigateur via http://localhost:5000

- requirements.txt

```
PS C:\Users\cyrin\docker_compose\product> cat requirements.txt
Flask==2.3.3
flask-restful==0.3.9
```

- **Dockerfile** pour construire l'image

```
PS C:\Users\cyrin\docker_compose\product> cat Dockerfile
FROM python:3.11-slim
WORKDIR /usr/src/app
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt
COPY . .
CMD ["python", "api.py"]
PS C:\Users\cyrin\docker_compose\product>
```

- api.py : (API simple avec un endpoint /)

```python
from flask import Flask
from flask_restful import Api, Resource

#Initialisation de l'application
app = Flask(__name__)
api = Api(app)

class Product(Resource):
    def get(self):
        return {
            'products':['Ipad pro 14', 'Iphone 13', 'Ordinateur bureautique']
        }

#Definition d'une route
api.add_resource(Product, '/')

#Execution de l'appilcation
if __name__ == '__main__':
    app.run(host='0.0.0.0', port=80, debug=True)
```

- Construction de l'image Docker :

```
PS C:\Users\cyrin\docker_compose\product> docker build -t product-api .
[+] Building 28.1s (11/11) FINISHED                                                                    docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                                   0.1s
 => => transferring dockerfile: 196B                                                                                   0.0s
 => [internal] load metadata for docker.io/library/python:3.11-slim                                                    3.1s
 => [auth] library/python:pull token for registry-1.docker.io                                                          0.0s
 => [internal] load .dockerignore                                                                                      0.1s
 => => transferring context: 2B                                                                                        0.0s
 => [1/5] FROM docker.io/library/python:3.11-slim@sha256:193fdd0bbcb3d2ae612bd6cc3548d2f7c78d65b549fcaa8af75624c      10.8s
 => => resolve docker.io/library/python:3.11-slim@sha256:193fdd0bbcb3d2ae612bd6cc3548d2f7c78d65b549fcaa8af75624c4     0.1s
 => => sha256:1771569cc1299abc9cc762fc4419523e721b11a3927ef968ae63ba0a4a88f2da 251B / 251B                             0.3s
 => => sha256:b3dd773c329649f22e467ae63d1c612a039a0559dec99ffb9ada904ab5c60c55 14.36MB / 14.36MB                       5.6s
 => => sha256:22b63e76fde1e200371ed9f3cee91161d192063bcff65c9ab6bf63819810a974 1.29MB / 1.29MB                         0.8s
 => => sha256:0e4bc2bd6656e6e004e3c749af70e5650bac2258243eb0949dea51cb8b7863db 29.78MB / 29.78MB                       6.2s
 => => extracting sha256:0e4bc2bd6656e6e004e3c749af70e5650bac2258243eb0949dea51cb8b7863db                              2.4s
 => => extracting sha256:22b63e76fde1e200371ed9f3cee91161d192063bcff65c9ab6bf63819810a974                              0.3s
 => => extracting sha256:b3dd773c329649f22e467ae63d1c612a039a0559dec99ffb9ada904ab5c60c55                              1.5s
 => => extracting sha256:1771569cc1299abc9cc762fc4419523e721b11a3927ef968ae63ba0a4a88f2da                              0.0s
 => [internal] load build context                                                                                      0.1s
 => => transferring context: 773B                                                                                      0.0s
 => [2/5] WORKDIR /usr/src/app                                                                                         0.4s
 => [3/5] COPY requirements.txt .                                                                                      0.1s
 => [4/5] RUN pip install --no-cache-dir -r requirements.txt                                                           9.5s
 => [5/5] COPY . .                                                                                                     0.1s
 => exporting to image                                                                                                 3.4s
 => => exporting layers                                                                                                2.1s
 => => exporting manifest sha256:574b438d71611701173c2c6a6a1b54dcc9fe5f72fb0fa298e8b5686b73a68fa4                      0.1s
 => => exporting config sha256:5e4416eb3d3a7de1a924e1e24e12264d7cd4771e9a825cead0c74cd94a169ee7                        0.0s
 => => exporting attestation manifest sha256:2fa713c11bfb3fa214569163bbf663a459db6ab0f85fc9acb7b60e0575ec29fd          0.1s
 => => exporting manifest list sha256:18980a793dc26a15e8a07e974965b11940ce9f76147d81e629c6212612d0e9de                 0.0s
 => => naming to docker.io/library/product-api:latest                                                                  0.0s
 => => unpacking to docker.io/library/product-api:latest                                                               1.0s
```

- Vérification de la création de l'image

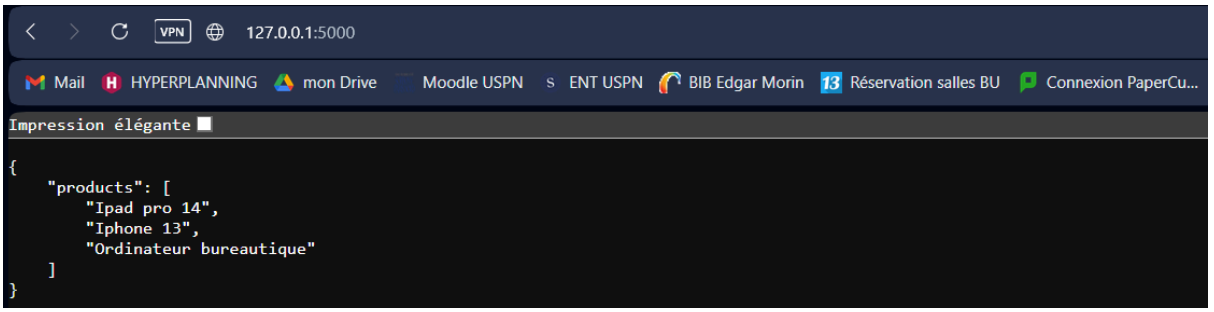
```
PS C:\Users\cyrin\docker_compose\product> docker images
REPOSITORY    TAG       IMAGE ID       CREATED              SIZE
product-api   latest    18980a793dc2   About a minute ago   215MB
redis         latest    43355efd2249   2 weeks ago          202MB
mysql         latest    569c4128dfa6   6 weeks ago          1.27GB
alpine        latest    4b7ce07002c6   8 weeks ago          12.8MB
PS C:\Users\cyrin\docker_compose\product>
```

- Lancement du conteneur

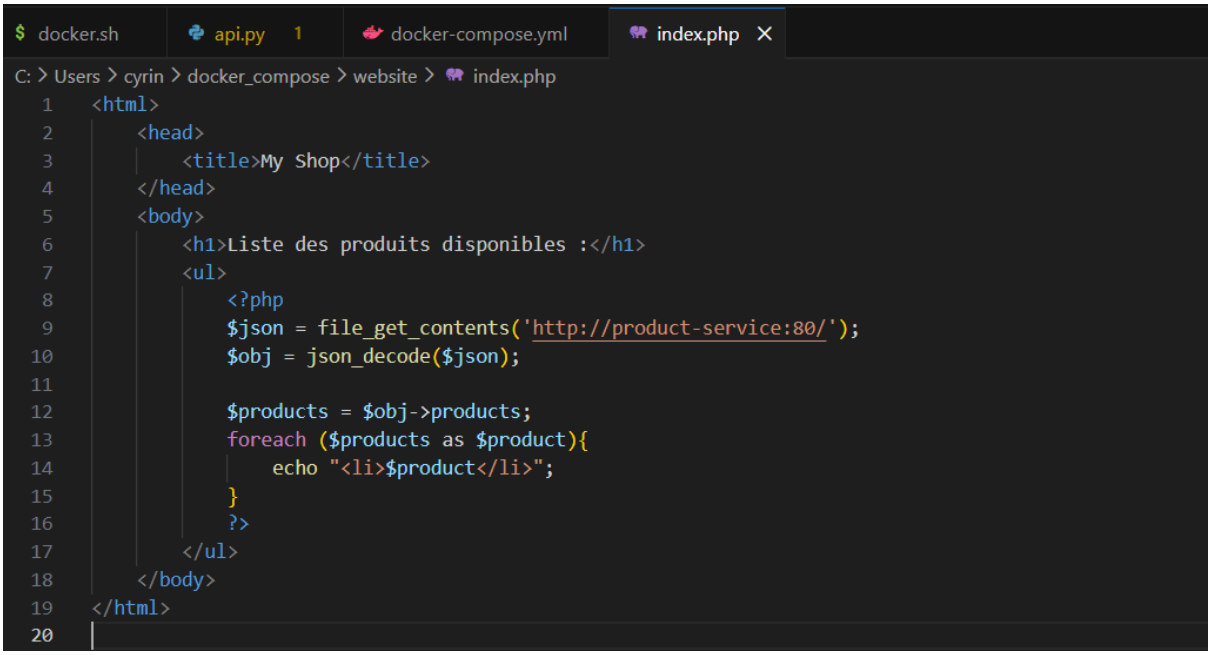
-p 5000:80 → mappe le port 5000 du PC au port 80 du conteneur.

```
PS C:\Users\cyrin\docker_compose\product> docker run -p 5000:80 product-api
 * Serving Flask app 'api'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:80
 * Running on http://172.17.0.2:80
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 134-471-344
172.17.0.1 - - [05/Dec/2025 09:07:06] "GET / HTTP/1.1" 200 -
172.17.0.1 - - [05/Dec/2025 09:07:07] "GET /favicon.ico HTTP/1.1" 404 -
```

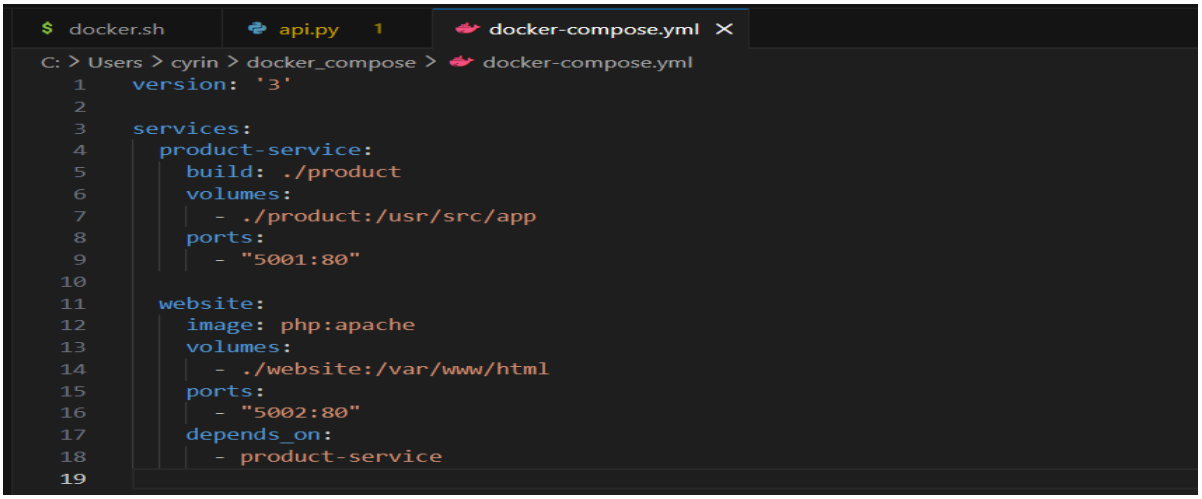
- Test dans le navigateur



Service 2 : website (PHP/Apache)



docker-compose.yml

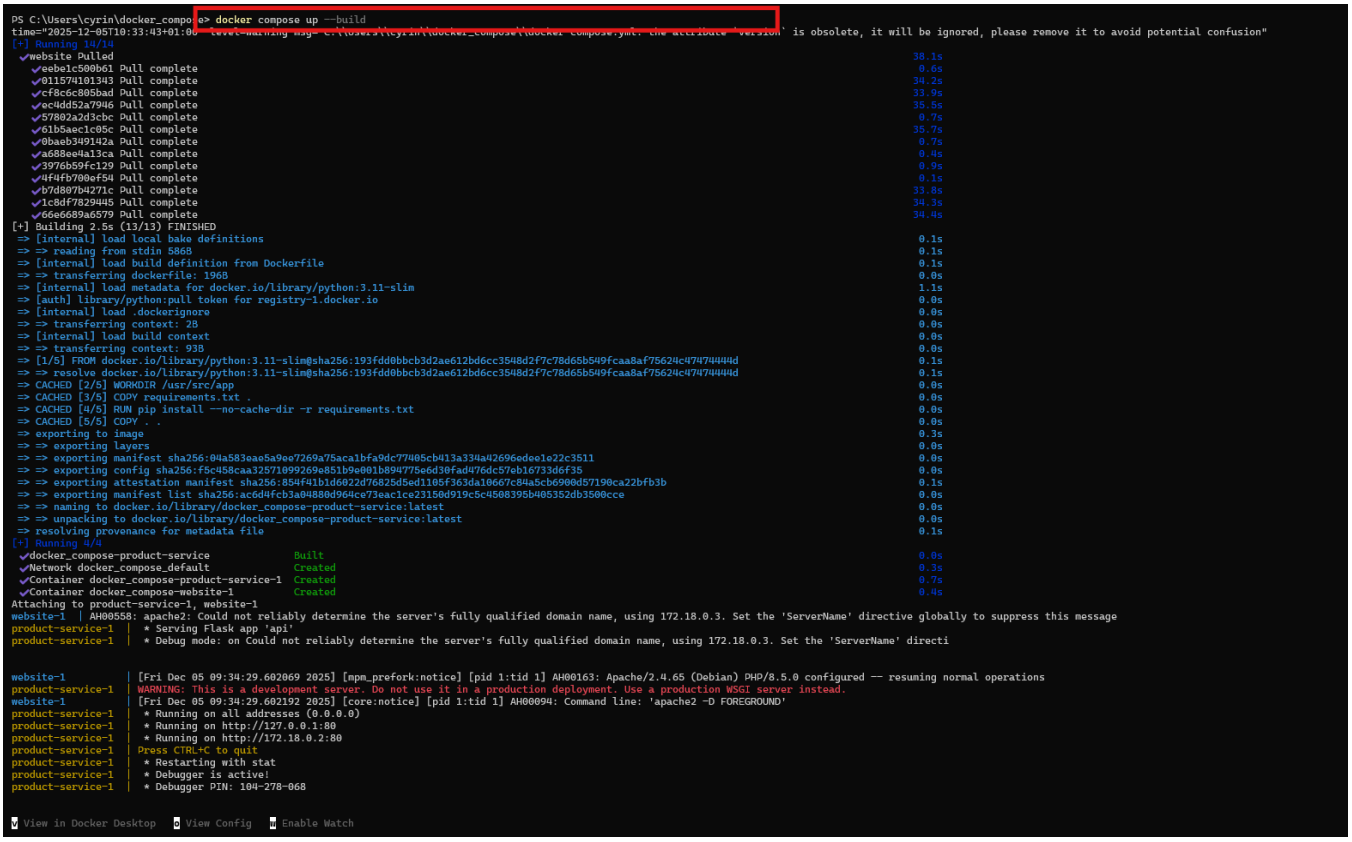


- product-service :
  - Construite à partir du Dockerfile dans product/
  - Expose port 5001 sur l'hôte

- website :
  - Image PHP/Apache officielle
  - Expose port 5002 sur l'hôte
- Dépend de product-service pour que Docker Compose démarre dans le bon ordre

Lancer les services :

Dans le terminal, à la racine docker_compose/



- Docker Compose construira l'image Flask et démarrera les deux services.
- Les services seront accessibles depuis le navigateur :
  - http://localhost:5001 → API Flask (JSON)
  - http://localhost:5002 → Site PHP affichant la liste des produits

Liste des produits disponibles :

- Ipad pro 14
- Iphone 13
- Ordinateur bureautique