Fiche Pratique Docker - Atelier 3

Objectif général

Mettre en pratique les compétences Docker : manipuler des conteneurs, écrire des Dockerfiles, gérer le stockage persistant et utiliser Docker Compose et GitLab CI/CD.

Exercice 1 : Manipuler des conteneurs

Objectifs : Lancer un conteneur depuis une image publique, arrêter, démarrer et inspecter les conteneurs

Étapes :

1. Puller une image publique (nginx)

```
docker pull nginx:latest
```

1. Lancer un conteneur Nginx

```
docker run -d --name mon-nginx -p 8080:80 nginx:latest
```

1. Vérifier si le conteneur fonctionne

```
docker ps
```

Accédez à http://localhost:8080

1. Inspecter le conteneur

```
docker inspect mon-nginx
```

1. Arrêter / démarrer un conteneur

```
docker stop mon-nginx
docker start mon-nginx
```

1. Supprimer un conteneur

```
docker rm -f mon-nginx
```

Récapitulatif : Conteneur lancé, inspecté, arrêté et supprimé.

Exercice 2 : Dockerfile pour une application Flask

Objectifs : Conteneuriser une application simple, créer un Dockerfile

Fichiers à créer :

- `app.py`
- `requirements.txt`
- `Dockerfile`

Étapes :

1. Créer le fichier `app.py` :

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello():
    return "Hello Docker avec Flask!"

@app.route('/health')
def health():
    return "OK"

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000, debug=True)
```

1. Créer le fichier `requirements.txt` :

```
Flask==2.3.3
```

1. Créer le fichier `Dockerfile` :

```
FROM python:3.9-slim
WORKDIR /app
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt
COPY app.py .
EXPOSE 5000
CMD ["python", "app.py"]
```

1. Construire l'image :

```
docker build -t mon-app-flask .
```

1. Lancer l'application :

```
docker run -d --name flask-app -p 5000:5000 mon-app-flask
```

Tester : http://localhost:5000

Exercice 3 : Pousser sur Docker Hub

Objectifs : Publier une image Docker, lancer un conteneur depuis le registry

Étapes :

1. Créer un token d'accès personnel sur Docker Hub
2. Se connecter à Docker Hub

```
docker login -u votre_nom_utilisateur
```

1. Taguer l'image

```
docker tag mon-app-flask votre_nom_utilisateur/flask-app:latest
```

1. Pousser l'image

```
docker push votre_nom_utilisateur/flask-app:latest
```

1. Lancer l'image depuis Docker Hub

```
docker run -d --name flask-from-hub -p 5001:5000 votre_nom_utilisateur/flask-
app:latest
```

Tester : http://localhost:5001

Exercice 4 : Volume persistant

Objectifs : Créer et utiliser un volume Docker

Étapes :

1. Créer un volume

```
docker volume create mon-volume
```

1. Lancer un conteneur avec volume

```
docker run -d --name nginx-volume -p 8080:80 -v mon-volume:/usr/share/nginx/
html nginx
```

1. Inspecter le volume

```
docker volume inspect mon-volume
```

1. Ajouter des données persistantes

```
# Accéder au conteneur
docker exec -it nginx-volume bash
# Créer un fichier dans le volume
echo "<h1>Données persistantes</h1>" > /usr/share/nginx/html/index.html
exit
```

1. Arrêter / redémarrer

```
docker stop nginx-volume
docker start nginx-volume
```

1. Vérifier persistance

```
curl http://localhost:8080
```

Exercice 5 : Docker Compose avec Network

Objectifs : Comprendre Docker Compose, lancer une application multi-conteneurs

Fichiers à créer :

- `nginx.conf`
- `docker-compose.yml`

Contenu de `nginx.conf` :

```
events {}
http {
    upstream flask-app {
        server flask-app:5000;
    }

    server {
        listen 80;
        location / {
```

```
            proxy_pass http://flask-app;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
        }
    }
}
```

Contenu de `docker-compose.yml` :

```
version: '3.8'

services:
  flask-app:
    build: .
    ports:
      - "5000:5000"
    environment:
      - FLASK_ENV=development
    volumes:
      - ./app:/app
    networks:
      - app-network
    depends_on:
      - mysql

  mysql:
    image: mysql:8.0
    environment:
      MYSQL_ROOT_PASSWORD: rootpassword
      MYSQL_DATABASE: flaskapp
      MYSQL_USER: appuser
      MYSQL_PASSWORD: apppassword
    volumes:
      - mysql-data:/var/lib/mysql
    networks:
      - app-network
    ports:
      - "3306:3306"

  nginx:
    image: nginx:latest
    ports:
      - "80:80"
    volumes:
      - ./nginx.conf:/etc/nginx/nginx.conf
    depends_on:
      - flask-app
    networks:
      - app-network
```

```
volumes:
  mysql-data:

networks:
  app-network:
    driver: bridge
```

Commandes :

```
docker-compose up -d
docker-compose logs
docker-compose down
docker-compose ps
docker network ls
docker network inspect <nom-projet>_app-network
```

Exercice 6 : GitLab CI/CD avec Docker (Solution 1 - Artifacts)

Objectifs : Automatiser build et push d'une image Docker

Variables GitLab :

- `DOCKERHUB_USERNAME` → votre nom d'utilisateur Docker Hub
- `DOCKERHUB_TOKEN` → token d'accès Docker Hub

Fichier à créer :

- `.gitlab-ci.yml`

Contenu de `.gitlab-ci.yml` :

```
stages:
  - build
  - push

variables:
  IMAGE_NAME: "$DOCKERHUB_USERNAME/flask_app"
  IMAGE_TAG: "latest"

build_image:
  stage: build
  image: docker:24.0.5
  services:
    - docker:24.0.5-dind
  variables:
    DOCKER_TLS_CERTDIR: "/certs"
  script:
```

```
    - echo "🔧 Construction de l'image Docker..."
    - docker build -t $IMAGE_NAME:$IMAGE_TAG .
    - docker save -o image.tar $IMAGE_NAME:$IMAGE_TAG
    - docker images
  artifacts:
    paths:
      - image.tar
    expire_in: 1 hour
  only:
    - main

push_image:
  stage: push
  image: docker:24.0.5
  services:
    - docker:24.0.5-dind
  variables:
    DOCKER_TLS_CERTDIR: "/certs"
  before_script:
    - echo "$DOCKERHUB_TOKEN" | docker login -u "$DOCKERHUB_USERNAME" --
password-stdin
  script:
    - echo "🗜️ Chargement de l'image..."
    - docker load -i image.tar
    - echo "🗜️ Poussée de l'image vers Docker Hub..."
    - docker push $IMAGE_NAME:$IMAGE_TAG
  dependencies:
    - build_image
  only:
    - main
```

Pipeline :

Suivre dans CI/CD → Pipelines Pusher le travail dans ton projet GitLab

---

Compétences acquises

- 🔗Manipulation de conteneurs
- 🔗Conteneurisation d'applications
- 🔗Construction et publication