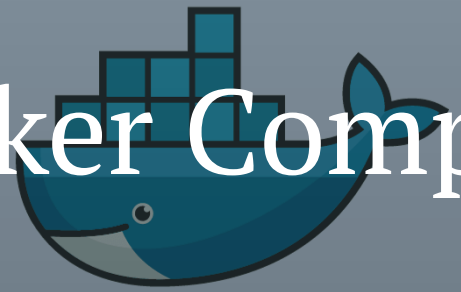
Docker Compose



Plan de cette partie

Objectifs :

- Comprendre l’intérêt de Docker Compose

Table des matières :

Pourquoi Docker Compose ?

- Commandes docker rapidement nombreuses et peu lisibles. Docker compose fourni un format lisible et compact
- pemet de décrire un système de containers dans un fichier et de le versionner (Infrastructure as Code)

Exemple :

```yaml
services:
  web:
    image: nginx:alpine
    ports:
      - "8000:80"
    volumes:
      - ./app:/usr/share/nginx/html
```

Format YAML

YAML (YAML Ain’t Markup Language)

Format basé sur l’indentation, exemple :

```yaml
personne:
  nom: Martin
  age: 42
  ville: Lyon
  competences:
    - programmation
    - gestion de projet
    - communication
  projets_actifs: 3 # Commentaire
```

Inclut la syntaxe JSON :

```yaml
personne:
  nom: Martin
  age: 42
  ville: Lyon
  competences: ['programmation', 'gestion de projet', 'communication']
  projets_actifs: 3 # Commentaire
```

Exemple Simple : Nginx

- Avec Docker CLI

```
docker run -name web -itd -p 8000:80 nginx:alpine
```

- Avec montage de volume

```
docker run --name web -itd -p 8000:80 -v $(pwd)/app:/usr/share/nginx/html nginx:alpine
```

- La version Docker Compose

```
services:
  web:
    image: nginx:alpine
    ports:
      - "8000:80"
    volumes:
      - ./app:/usr/share/nginx/html
```

Commandes Docker Compose Essentielles

Commande	Description
`docker compose build`	Construction des images
`docker compose up`	Lancement des services
`docker compose up -d`	Lancement en arrière-plan
`docker compose ps`	Liste des containers
`docker compose logs`	Visualisation des logs

Commandes Docker Compose Avancées

Commande	Description
`docker compose pause`	Pause des services
`docker compose unpause`	Reprise des services
`docker compose stop`	Arrêt (conserve les données)
`docker compose down`	Arrêt et suppression
`docker compose down --rmi all`	Nettoyage complet

Exemple Multi-Containers : Flask et Redis

```
from flask import Flask
from redis import Redis

app = Flask(__name__)
redis = Redis(host='redis', port=6379)

@app.route('/')
def hello():
    count = redis.incr('hits')
    return 'Hello for the {} time !\n'.format(count)

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

Dockerfile et docker-compose.yml

- Dockerfile pour Flask

```
FROM python:3.11-slim
ADD . /app
WORKDIR /app
RUN pip install -r requirements.txt
EXPOSE 5000
CMD ["python", "run.py"]
```

- Docker Compose

```
services:
  web:
    build: .
    ports:
      - "4000:5000"
  redis:
    image: "redis"
```

Variables d’environnement

- Gestion des secrets stockés dans un fichier .env. Exemple :

```
DB_HOST=localhost
API_KEY=123456
```

- Injection dans les conteneurs :

```
services:
  app:
    image: exemple:12
    environment:
      - DB_HOST=${DB_HOST}
      - API_KEY=${API_KEY}
```

Volumes et réseaux

```yaml
services:
  web:
    image: nginx
    networks:
      - back
    volumes:
      - ./app:/usr/share/nginx/html
  app:
    image: myapp
    networks:
      - database
      - back
  db:
    image: postgres
    networks:
      - database
    volumes:
      - db_data:/var/lib/postgresql/data
volumes:
  db_data:

networks:
  database:
  back:
```

Autres points "standards"

Dependences

```yaml
services:
  web:
    image: nginx
    depends_on:
      - app
  app:
    image: myapp
```

Options de build

```yaml
services:
  web:
    build:
      context: .
      dockerfile: Dockerfile.dev
```

Profiles

```yaml
services:
  appdev:
    image: myapp
    volumes:
      - ./src:/usr/share/persistence
    profiles:
      - dev
  appprod:
    image: myapp
    profiles:
      - prod
```

```
docker compose --profile dev up
```

À vous