



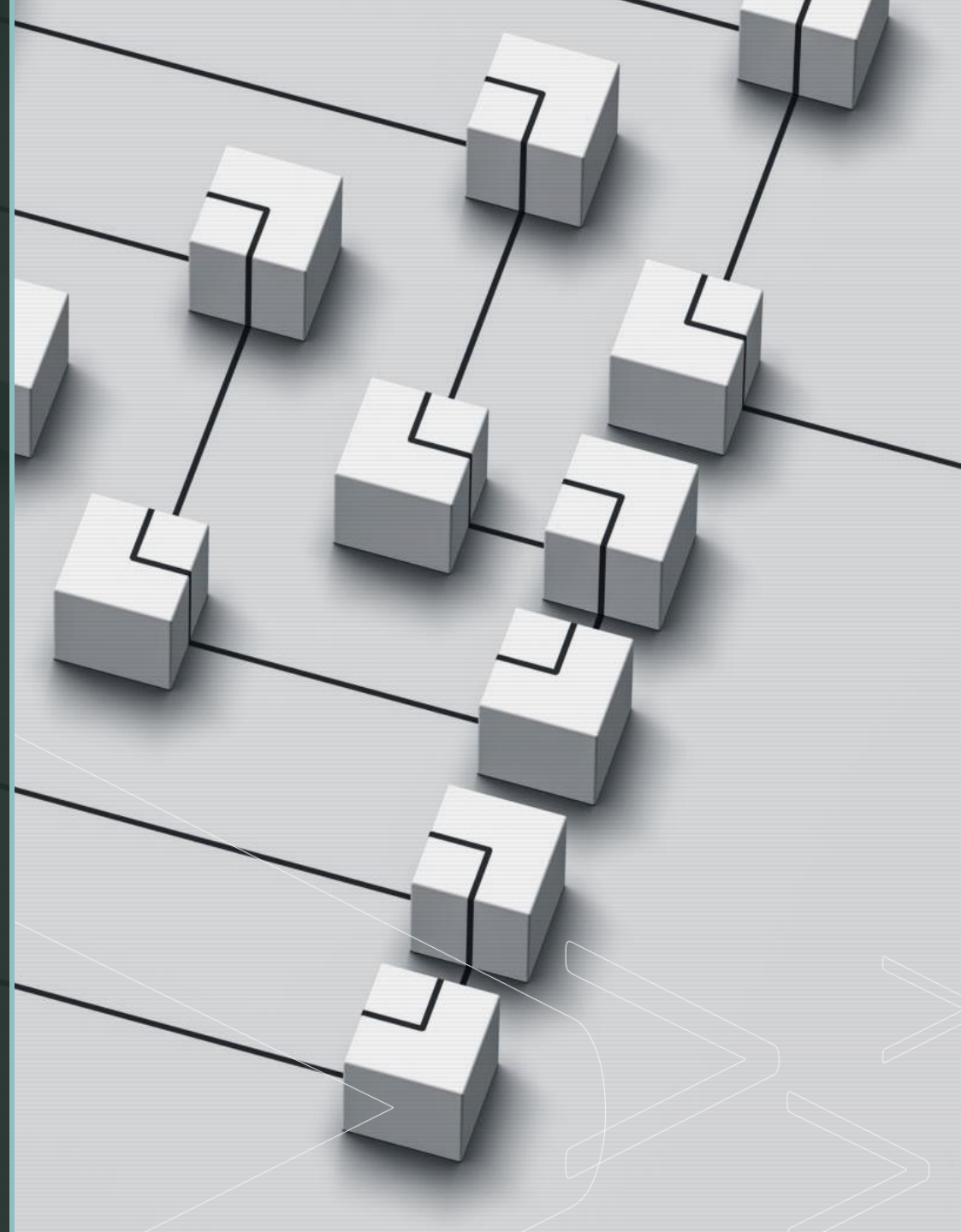
Exemple de Conteneurisation

Docker



Qu'est-ce que Docker ?

- Plate-forme open-source pour faciliter déploiement & gestion d'applications
- Encapsulation d'application + dépendances dans un conteneur
- Garantie de fonctionnement indépendamment de l'environnement



Conteneurs

- **Isolation** : isolation au niveau de l'application. Entité autonome, séparée des autres conteneurs et du système hôte. Garantit que les dépendances de l'application ne vont pas entrer en conflit avec celles d'autres applications.
- **Environnement cohérent** : dépendances requises par l'application, (système d'exploitation, bibliothèques, fichiers de configuration, etc.) encapsulées dans le conteneur. Pas de problèmes de compatibilité.
- **Portabilité** : exécutés de manière identique sur n'importe quelle machine ou infrastructure qui prend en charge la technologie de conteneurisation (comme Docker). Plus de problèmes liés à la différence d'environnements entre développement et production.
- **Léger** : par rapport aux machines virtuelles (VM). Même noyau du système d'exploitation que l'hôte, ce qui réduit la surcharge et permet de lancer rapidement des instances de conteneur.
- **Facilité de gestion** : faciles à créer, à distribuer et à gérer. Ils peuvent être construits à partir de fichiers de configuration (Dockerfiles) et partagés via des registres de conteneurs. La gestion de l'évolutivité est également simplifiée.
- **Sécurité** : Les conteneurs offrent un certain niveau de sécurité en limitant l'accès aux ressources système. Cependant, il est important de noter que la sécurité dépend de la configuration et des meilleures pratiques de l'utilisateur.
- **Interchangeabilité** : Les conteneurs sont interchangeables. Vous pouvez détruire un conteneur et en créer un nouveau en quelques secondes si nécessaire, ce qui simplifie la gestion des mises à jour et des correctifs.

Installation

- Installer Docker sur...
- Linux : `sudo apt update ; sudo apt install docker.io`
- Windows, macOS
 - Docker Desktop (inclut Docker Engine)
 - Cf : <https://www.docker.com/products/docker-desktop>
- Version : `docker --version`

Création d'un Dockerfile

- C'est un fichier texte avec les instructions pour construire une image Docker personnalisée
- Chaque Dockerfile commence par une image de base (FROM) et ajoute des couches d'instructions pour configurer l'environnement de l'application.

```
Microservices > Docker > Dockerfile > ...  
# Utilisez une image Python 3.11 comme image de base  
FROM python:3.11-alpine  
  
# Définit le répertoire de travail  
WORKDIR /app  
  
# Copie les fichiers de l'application dans le conteneur  
COPY . /app  
  
# Installe les dépendances Python définies dans requirements.txt  
RUN pip install -r requirements.txt  
  
# Expose le port 5000 pour l'application Flask  
EXPOSE 5000  
  
# Commande pour exécuter l'application  
CMD ["python", "sql_fast_api.py"]
```

Exercice 1

```
BUT3 > Microservices > SQLite > requirements.txt
1  Flask==2.1.2
2  waitress==2.1.2
```

- Reprenez l'API faite avec SQLite3
- Vérifiez bien que l'IP est `0.0.0.0`
(pas localhost pour Docker !)
- Créez un fichier `requirements.txt` des librairies de Python à installer
(installer `pip install pipreqs` puis `pipreqs . --force`)
- Créez le fichier Dockerfile pour lancer l'application
(FROM `python:3.11-alpine` car alpine très léger)

Construction de l'image Docker

- Commande docker build
- Syntaxe : `docker build -t nom_image:version .`
- Option `-t` : spécifier image et version (tag)
- Point final "." : si Dockerfile dans répertoire actuel

Exécution d'un conteneur

- Après le build
- Commande :
`docker run -p 5018:5009 nom_image:version`
- Option `-p` : mapper les ports hôte:conteneur (ex : 5018:5009)
- Image : `nom_image` (par défaut ``latest``)

Détruire / Arrêter les Conteneurs et Images

- Arrêter un conteneur :
`docker stop nom_conteneur`
- Supprimer le conteneur :
`docker rm nom_conteneur`
- Supprimer l'image :
`docker rmi nom_image:version`
- Lister les conteneurs et images :
`docker ps -a` # Liste les conteneurs + ceux arrêtés.
`docker images` # Liste les images Docker.

Exercice 2

Container memory usage ⓘ
382.37MB / 22.9GB

☰ Only show running containers

Image	Status	Port(s)	Last start...
fastapi_sqlite:latest	Running	5018:5009 ↗	20 minutes ago

🔍 Search

Name	Tag	Status	Created	Size	Actions
fastapi_sqlite	latest	In use	29 seconds ago	81.11 MB	⌵ ⌵ ⌵
abdbead6002a					🗑

🔍 Search

🔍 local: x | 📄 FastAPI: x | 📄 Meta: x | 🌐 local: x | +

🔍 localhost:5018/utilisateur/emprunts/2

🔍 Bookmarks 📄 Moodle 📄 Gmail 📄 PartageEmail 📄 GitHub

```
[{"id":3,"titre":"Germinal","pitch":"L'histoire d'une grève des mineurs dans le nord de la France et de leur lutte pour de meilleures conditions de travail.", "auteur_id":3,"date_public":"28/11/1885","emprunteur_id":2}, {"id":12,"titre":"La Métamorphose","pitch":"L'histoire de Gregor Samsa, un homme qui se réveille un matin transformé en un insecte géant et qui doit faire face à cette nouvelle réalité.", "auteur_id":11,"date_public":"15/10/1915","emprunteur_id":2}, {"id":18,"titre":"Les Trois Mousquetaires","pitch":"L'histoire de d'Artagnan, un jeune gascon qui quitte sa maison pour devenir mousquetaire et vivre de nombreuses aventures avec ses amis.", "auteur_id":14,"date_public":"14/03/1844","emprunteur_id":2}, {"id":23,"titre":"L'Assommoir","pitch":"L'histoire de Gervaise Macquart, une blanchisseuse qui lutte pour survivre dans les rues de Paris, tout en étant confrontée à l'alcoolisme et à la pauvreté.", "auteur_id":3,"date_public":"17/03/1877","emprunteur_id":2}]
```

- Construisez votre image Docker
- Vérifiez qu'elle est correctement construite

🔍 Search

Name	Tag	Status	Created	Size	Actions
fastapi_sqlite	latest	In use	29 seconds ago	81.11 MB	⌵ ⌵ ⌵
abdbead6002a					🗑

- Lancez une instance (port 5018 par exemple)
- Testez votre application

Volume : dossier partagé avec hôte

- **Problème** : Notre base de données est dans le conteneur
Si celui-ci est détruit, les modifications sont perdues !
- Définir un point de montage de volume (Dockerfile) :
`VOLUME /app/data`
- Puis monter le volume à l'exécution :
`docker run -v /chemin/vers/mon/dossier/local:/app/data
nom_image:version`
- **Meilleure solution** (en production) : Service persistant de data (AWS S3 ou autre...)

Exercice 3

POST /utilisateur/ajouter Ajouter Utilisateur

Parameters

No parameters

Request body required

```
{
  "nom": "Eléonore Attréides",
  "email": "elattr@ici.fr"
}
```

Curl

```
curl -X 'POST' \
  'http://localhost:5018/utilisateur/ajouter' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
    "nom": "Eléonore Attréides",
    "email": "elattr@ici.fr"
  }'
```

- Modifiez votre code d'API en changeant les références à votre database :

```
# chemin vers base de données
path_to_db = pathlib.Path(__file__).parent.absolute() /
"data" / "database.db"
```

- Dans le Dockerfile ajoutez le point de montage de volume correspondant
- Arrêtez, effacez conteneur et image
- Refaites le build et lancez votre conteneur avec le montage du volume
- Vérifiez qu'une modification de la database se fait bien sur le fichier de l'hôte

BUT3 > Microservices > Docker > data > database.db

Search tables...		Reset Filters	Records: 4
Tables (4)	id	nom	email
utilisateurs	Search column...	Search column...	Search column...
sqlite_sequence	1	1 Alice	nouveau.email.alice...
auteurs	2	2 John Doe	john.doe@example.c...
livres	3	3 Jane Smith	jane.smith@example...
	4	5 Eléonore Attréides	elattr@ici.fr

```

docker-compose.yml - The Compose specification establishes
version: '3.1'
services:
  front:
    image: frontservice
    build:
      context: ./FrontService
      dockerfile: Dockerfile
    ports:
      - "8089:8089"
    depends_on:
      - api
      - auth
    environment:
      - AUTH_SERVICE_URL=http://auth:8088
      - API_SERVICE_URL=http://api:5002
  api:
    image: exopython
    build:
      context: ./OpenAPI
      dockerfile: Dockerfile
    # expose:
    #   - "5002"
    ports:
      - "5002:5002"
    volumes:
      - ./OpenAPI/static/db:/app/static/db
    depends_on:
      - auth
    environment:
      - AUTH_SERVICE_URL=http://auth:8088
      - API_SERVICE_URL=http://api:5002
      - FRONT_SERVICE_URL=http://front:8089
  auth:
    image: authenticator
    build:
      context: ./OAuth
      dockerfile: Dockerfile
    ports:
      - "8088:8088"

```

Docker-compose

- Plusieurs conteneurs en tant que services
- Un fichier de configuration pour tous les services
- Format : compose.yaml ou docker-compose.yml (old)
- Lancer les services :
`docker-compose up -d`
- <https://docs.docker.com/compose/features-uses/>
- <https://docs.docker.com/compose/compose-file/>

Docker-compose Structure

- Top-level :
version, name, services, networks,
volumes, configs, secrets
- Version et Name : seulement informatifs
- Configs : monter les fichiers de config
- Services : cf. diapo suivante

```
services:  
  redis:  
    image: redis:latest  
    configs:  
      - my_config  
      - my_other_config  
configs:  
  my_config:  
    file: ./my_config.txt  
  my_other_config:  
    external: true
```

Docker-compose Services

- Services :
build, depends_on,
configs, entrypoint,
environment, expose,
ports, secrets, volumes
- Build : context & dockerfile
- Ports : comme expose
- Depends_on : ordre de build
- Volumes : montage de dossier
- <https://docs.docker.com/compose/compose-file/05-services/>

```
services:
  front:
    image: frontservice
    build:
      context: ./front
      dockerfile: Dockerfile
    ports:
      - "8092:8092"
    depends_on:
      - api
    volumes:
      - ./front/data:/app/data
    environment:
      - API_URL=http://api:5009
```

Docker-compose Volumes

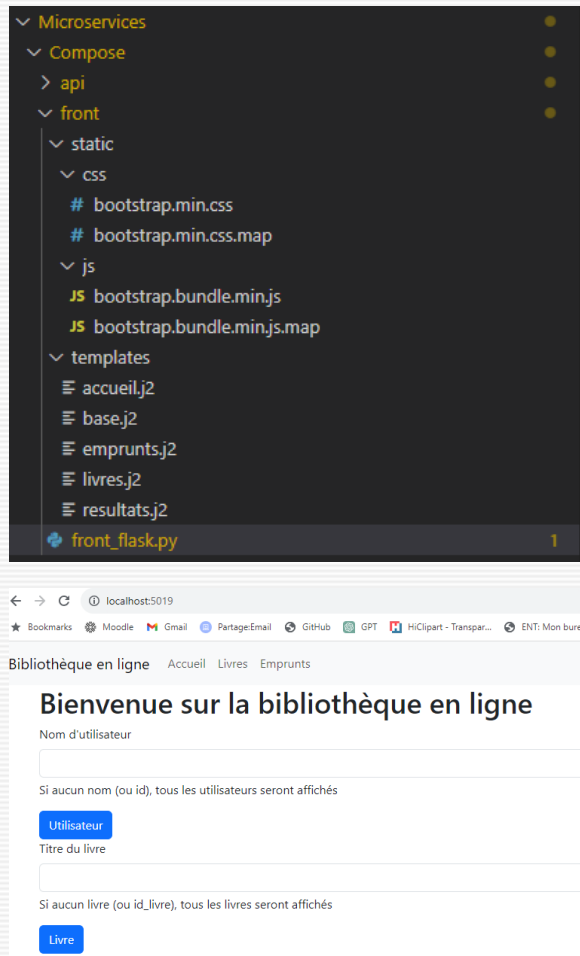
- Volumes :
données persistantes
(tous services)
- Définis dans les services
- Echanges de données

```
services:
  backend:
    image: example/database
    volumes:
      - db-data:/etc/data

  backup:
    image: backup-service
    volumes:
      - db-data:/var/lib/backup/data

volumes:
  db-data:
```

Exercice 4



- Ajoutez une application Flask pour le front-end et un docker-compose pour lier les services
- Créez 4 endpoints basés sur base.j2 pour consulter la bibliothèque : accueil, emprunts, livres et resultats
- Sur la page d'accueil, un formulaire avec deux boutons `submit` pour chaque champ (utilisateur ou livre) qui permet de faire une recherche par mot-clé (ou sur id si un entier est entré)
- Endpoint `/livres` : pour tester. Faites l'appel par `requests` à l'URL définie par

```
api_service_url = os.environ.get("API_SERVICE_URL", "http://api:5009")
URL_EXO = api_service_url + "/"
```

- Idéalement créer un tableau HTML en Jinja2 en page `/livres` (cf. Captures page suivante)
- Rem : requirements (sinon bug Flask) **Werkzeug==2.3.7**

Endpoint /livres



The screenshot shows a web browser window with the address bar displaying 'localhost:8092/livres'. The browser's bookmark bar includes links to Bookmarks, Moodle, Gmail, Partage:Email, GitHub, GPT, HiClipart - Transpar..., and ENT: Mon bureau. The page content features a navigation bar with 'Bibliothèque en ligne', 'Accueil', 'Livres', and 'Emprunts'. Below this is a heading 'Bienvenue sur la bibliothèque en ligne' followed by 'Liste des livres'. A table lists ten books with their titles, authors (represented by numbers 1-9), publication years, and availability status ('Oui' or 'Non').

Titre	Auteur	Année de publication	Dispo ?
Les Misérables	1	14/04/1862	Oui
Madame Bovary	2	01/10/1856	Oui
Germinal	3	28/11/1885	Non
L'Étranger	4	01/06/1942	Oui
La Peste	4	16/06/1947	Non
Les Fleurs du mal	5	23/06/1857	Oui
Les Liaisons dangereuses	6	23/03/1782	Oui
Voyage au bout de la nuit	7	04/10/1932	Oui
Les Choses	8	01/03/1965	Oui
Les Cerfs-volants	9	29/05/2003	Oui

Exercice 5

localhost:8092

Bookmarks Moodle Gmail Partage:Email GitHub GPT HiClipart - Transpar... ENT: M

Bibliothèque en ligne Accueil Livres Emprunts

Bienvenue sur la bibliothèque en ligne

Nom d'utilisateur

Si aucun nom (ou id), tous les utilisateurs seront affichés

Utilisateur

Titre du livre

mo

Si aucun livre (ou id_livre), tous les livres seront affichés

Livre

Résultats de la requête

Titre

La Métamorphose

Le Comte de Monte-Cristo













Les Trois Mousquetaires

La Vie mode d'emploi

L'Assommoir

- Détruire les containers et les images
- Dans l'API ajouter une route `livres_full` qui donne une liste complète des livres avec les noms des auteurs (en jointure)
- Ajouter l'endpoint associée au formulaire
- Relancer le docker-compose up
- Tester l'API cliente ("front") en association avec l'API

Search Only show running containers

<input type="checkbox"/>	Name	Image	Status	Port(s)	Last start... ↓	CPU (%)	Actions
<input type="checkbox"/>	 biblio_compose		Running (2/2)		4 minutes ago	0.19%	  
<input type="checkbox"/>	 front-1 7df635393592	frontservice	Running	8092:8092	4 minutes ago	0.01%	  
<input type="checkbox"/>	 api-1 a9c2f6c3d0a0	sqlservice	Running	5009:5009	5 minutes ago	0.18%	  

Endpoint /livres

- Modifiée avec jointure sur les noms des auteurs

Bibliothèque en ligne Accueil Livres Emprunts			
Bienvenue sur la bibliothèque en ligne			
Liste des livres			
Titre	Auteur	Année de publication	Dispo ?
Les Misérables	Victor Hugo	14/04/1862	Oui
Madame Bovary	Gustave Flaubert	01/10/1856	Oui
Germinal	Émile Zola	28/11/1885	Non
L'Étranger	Albert Camus	01/06/1942	Oui
La Peste	Albert Camus	16/06/1947	Non
Les Fleurs du mal	Charles Baudelaire	23/06/1857	Oui
Les Liaisons dangereuses	Pierre Choderlos de Laclos	23/03/1782	Oui
Voyage au bout de la nuit	Louis-Ferdinand Céline	04/10/1932	Oui

Endpoint /results

Bibliothèque en ligne Accueil Livres Emprunts

Bienvenue sur la bibli

Nom d'utilisateur

jo

Si aucun nom (ou id), tous les utilisateurs seront aff

Utilisateur

Résultats de la requête

Nom: John Doe

Email: john.doe@example.com

Emprunts ci-dessous :

Titre

Germinal

La Métamorphose

Les Trois Mousquetaires

L'Assommoir