

---

# Virtualisation, conteneurisation



Alexis FERRO

Alexis GUIBON CLAVEL

Vincent GENIN

Guillaume POUVREAU

Semestre S7

2023-2024

---

## Table des matières

I.	Introduction.....	2
II.	Création du Docker-compose.yml .....	3
II.1	Difficulté rencontrée : configuration docker-compose.....	3
II.2	Difficulté rencontrée : configuration mot de passe.....	4
	Rendu de la page web après exécution.....	5
III.	Création de l'application de façon manuelle.....	5
IV.	Conclusion : les enseignements tirés .....	8

## I. Introduction

Dans le but d'explorer les avantages et les inconvénients de l'utilisation des conteneurs, nous avons suivi un projet de virtualisation au terme de ce premier semestre.

L'objectif était la virtualisation des services existants via des conteneurs Docker, de sorte à les orchestrer efficacement à l'aide de Docker Compose à l'intérieur d'une machine virtuelle VMWare EXSI.

À partir du dépôt GitHub de notre professeur contenant l'intégralité des fichiers nécessaires à l'accomplissement de ce projet, nous avons donc commencé par cloner le dépôt GitHub en utilisant la commande « *git clone URL* »

“ Une brève description de ce que représente Docker :

*Docker est une plate-forme logicielle qui vous permet de concevoir, tester et déployer des applications rapidement. Docker intègre les logiciels dans des unités normalisées appelées conteneurs, qui rassemblent tous les éléments nécessaires à leur fonctionnement, dont les*

*bibliothèques, les outils système, le code et l'environnement d'exécution.* ”



Dans un premier temps, traversons succinctement les quelques dossiers figurant dans ce GitHub et cherchons à bien comprendre la fonction de chacun :

### Healthchecks

- Cette instruction indique à Docker comment vérifier si notre container fonctionne toujours correctement tout au long de son utilisation. Elle est déclarée dans le fichier Dockerfile, lors de la création de l'image Docker.

### Result

- Le dossier contient les fichiers nécessaires à la gestion des résultats du projet. Il englobe deux sous-dossiers, "test" qui présente fichiers nécessaires pour l'implémentation et l'exécution de tests sur les différents services du projet et "views" qui gère les fichiers nécessaires à la gestion de l'interface utilisateur ainsi qu'à la présentation visuelle des résultats/données traitées par les différents services.

### Seed Data

- Dossier qui sert d'alimentation de la base de données et ajoute des données initiales, elle est réalisée à des fins de test ou de développement et permet de travailler avec un échantillon de données sans avoir à s'inquiéter de la corruption des données réelles.

## Vote

- Le dossier englobe les éléments liés à la gestion des votes dans le système. Il est chargé de traiter les votes des utilisateurs avec une mise en forme des votes et une présentation visuelle (styles.css)

## Worker

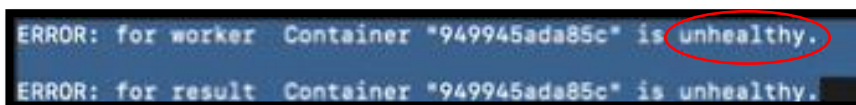
- Il est essentiel au bon fonctionnement de l'écosystème et au cœur du traitement asynchrone, de la gestion des tâches en arrière-plan. Cela offre une approche efficace pour traiter des opérations sans impacter la performance globale du système. Le dossier contient également le README qui fournit une documentation détaillée sur le fonctionnement du worker, les dépendances nécessaires, les étapes pour le déploiement ; l'image représentant une vue schématique de l'architecture du worker ; le docker-compose qui orchestre les conteneurs et définit la configuration et liens avec les autres services.

## II. Création du Docker-compose.yml

Une fois l'ensemble des fichiers récupérés, l'étape suivante était docker-compose.yml pour orchestrer l'ensemble des services individuels. Il est important de les lier aux réseaux spécifiques (networks) car cela favorise la communication entre les conteneurs. Chaque service devait donc être configuré avec les paramètres nécessaires de sorte à assurer une interconnexion optimale.

### II.1 Difficulté rencontrée : configuration docker-compose

Nous avons fait face à une difficulté majeure liée à la connexion à la base de données. En effet, nous ne pensions pas qu'il était impératif de se connecter à la base de données pour garantir le bon fonctionnement du service. De ce fait, nous avons obtenus des erreurs dans le healthcheck des conteneurs lors de l'exécution se manifestant par le message "ERROR: for worker Container "949945ada85c" is unhealthy"



```
ERROR: for worker Container "949945ada85c" is unhealthy.  
ERROR: for result Container "949945ada85c" is unhealthy.
```

en dépit du "healthy" que nous aurions dû avoir



```
✓ Container esiea-ressources-db-1 Healthy  
✓ Container esiea-ressources-result-1 Started
```

en découlait seulement deux « done » sur trois et une incohérence.

```
Recreating esiea-ressources_db_1 ... ✗
Recreating esiea-ressources_db_1 ... done
Starting esiea-ressources_seed-data_1 ... done
```

Le healthcheck n'était en fait pas en mesure de se connecter à la base de données ce qui a entraîné une évaluation négative du conteneur "worker". L'absence de a conduit à des erreurs dans la gestion de la santé du conteneur. C'est pourquoi nous nous sommes rendus sur les ressources disponibles du Docker Hub, accessibles depuis le README de Postgres fourni par le professeur.

Ainsi, en ajoutant ces lignes :

... via **docker-compose** or **docker stack deploy**

Example docker-compose.yml for postgres :

```
# Use postgres/example user/password credentials
version: '3.1'

services:
  db:
    image: postgres
    restart: always
    environment:
      POSTGRES_PASSWORD: example

  adminer:
    image: adminer
```

**environment:**  
**POSTGRES\_PASSWORD: postgres**

Nous avons permis au conteneur worker de se connecter correctement à la base de données, éliminant ainsi les erreurs de « unhealthy ».

## II.2 Difficulté rencontrée : configuration mot de passe

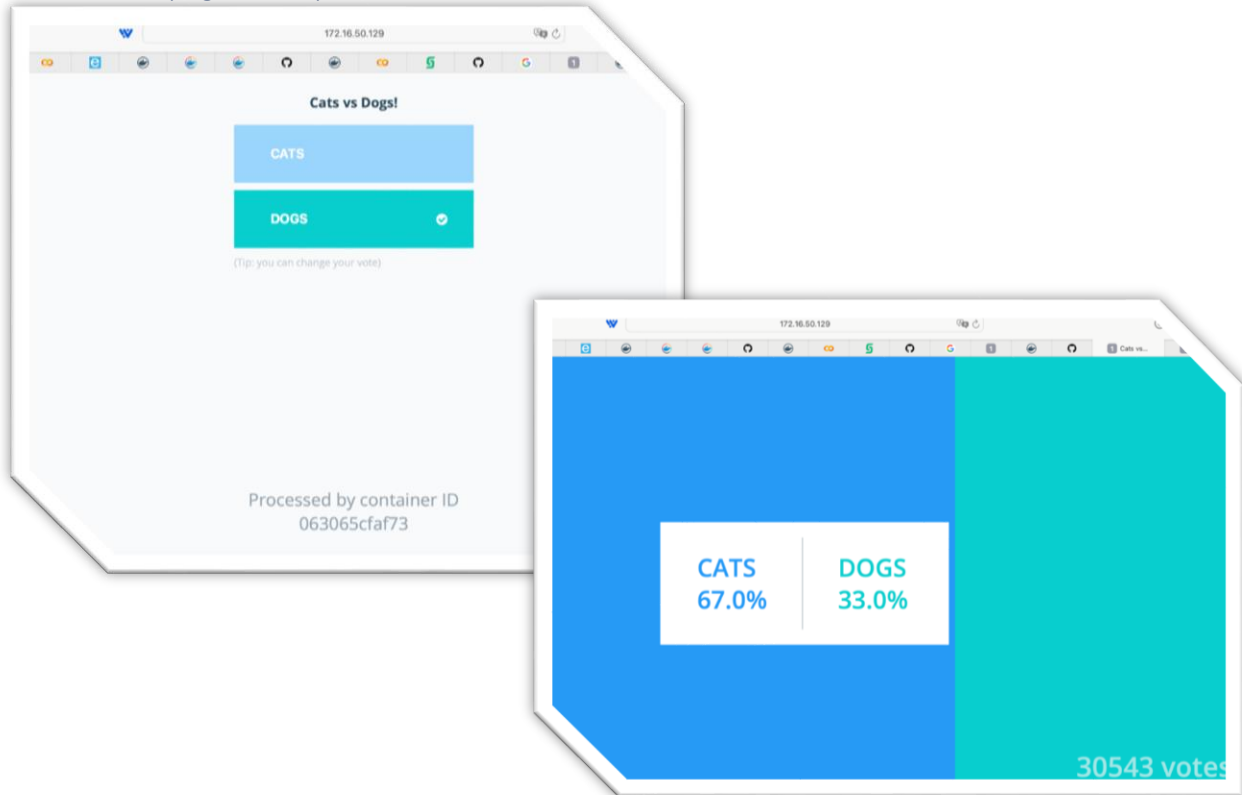
De plus, nous avons rencontré un problème avec la gestion du mot de passe de la base de données. Après avoir premièrement configuré un mot de passe, le déploiement ne se faisait pas correctement. Nous avons donc éliminé la ligne et configuré un nouveau mot de passe mais cela n'a pas résolu le problème. Après plusieurs tentatives non réussies, nous avons découvert que le premier le mot de passe configuré restait celui de référence peu importe les modifications. La solution était donc de réintroduire un nouveau mot de passe après avoir supprimé l'intégralité des configurations précédemment faites.

```
alexisfr7@alex:~/esiea-ressources$ docker rm esiea-ressources-db-1
esiea-ressources-db-1
alexisfr7@alex:~/esiea-ressources$ docker volume rm esiea-ressources_db-data
esiea-ressources_db-data
```

Puis le souci fût complètement réglé.

```
✓ Volume "esiea-ressources_db-data" Created
✓ Container esiea-ressources-db-1 Created
✓ Container esiea-ressources-vote-1 Created
✓ Container esiea-ressources_seed-data-1 Created
✓ Container esiea-ressources_redis-1 Created
Attaching to esiea-ressources-db-1 esiea-ressources-vote-1 esiea-ressources_seed-data-1 esiea-ressources_redis-1
```

Rendu de la page web après exécution



### III. Création de l'application de façon manuelle

- Tout d'abord, nous avons dupliqué les ressources GitHub grâce la commande `cp` avant d'entamer la création du réseau docker « *cats-or-dogs-network* » avec le sous réseau 172.23.0.0/24.

```
alexisfr7@alex:~/esiea-manuelle$ docker network create --driver=bridge cats-or-dogs-network --subnet=172.23.0.0/24
2b6406b1541521f5b0381ce364c13db01961072c2e6654e6b4bc09710113dc61
alexisfr7@alex:~/esiea-manuelle$ docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
dbc043c11fe4        bridge             bridge              local
2b6406b15415        cats-or-dogs-network bridge              local
```

- Nous avons créé la base de données qui est un volume docker appelé : « *db-data* ».

```
alexisfr7@alex:~/esiea-manuelle$ docker volume create db-data
db-data
```

- Nous avons ensuite construit les images Docker à partir des fichiers Docker file situés dans les répertoires respectifs :

#### - vote

```
alexisfr7@alex:~/esiea-manuelle$ docker build -t alexisfr-virtu-vote:cmd ./vote/
[+] Building 0.5s (11/11) FINISHED
=> [internal] load .dockerignore                                docker:default 0.1s
=> => transferring context: 2B                                  0.0s
=> [internal] load build definition from Dockerfile              0.1s
=> => transferring dockerfile: 1.09kB                            0.0s
=> [internal] load metadata for docker.io/library/python:3.11-slim 0.0s
=> [base 1/5] FROM docker.io/library/python:3.11-slim           0.0s
=> [internal] load build context                                0.0s
=> => transferring context: 356B                                   0.0s
=> CACHED [base 2/5] RUN apt-get update && apt-get install -y --no-install-recommends curl && rm -rf /var/lib 0.0s
=> CACHED [base 3/5] WORKDIR /usr/local/app                      0.0s
=> CACHED [base 4/5] COPY requirements.txt ./requirements.txt    0.0s
=> CACHED [base 5/5] RUN pip install --no-cache-dir -r requirements.txt 0.0s
=> CACHED [final 1/1] COPY . .                                    0.0s
=> exporting to image                                           0.0s
=> exporting layers                                             0.0s
=> writing image sha256:5600fa2b9c58b0a1be406a9e9721c254c2bf72bcd3b592c387ea5107d963e2a 0.0s
=> naming to docker.io/library/alexisfr-virtu-vote:cmd          0.0s
```

#### - seed-data

```
alexisfr7@alex:~/esiea-manuelle$ docker build -t alexisfr-virtu-seed-data:cmd ./seed-data/
[+] Building 1.7s (10/10) FINISHED
=> [internal] load build definition from Dockerfile              docker:default 0.1s
=> => transferring dockerfile: 346B                              0.0s
=> [internal] load .dockerignore                                0.1s
=> => transferring context: 2B                                  0.0s
=> [internal] load metadata for docker.io/library/python:3.9-slim 1.3s
=> [1/5] FROM docker.io/library/python:3.9-slim@sha256:96be08c44307e781fd9ce8e05b49c969b4cb902ec23594f904739c58da3a09 0.0s
=> [internal] load build context                                0.0s
=> => transferring context: 101B                                   0.0s
=> CACHED [2/5] RUN apt-get update && apt-get install -y --no-install-recommends apache2-utils && rm -rf 0.0s
=> CACHED [3/5] WORKDIR /seed                                   0.0s
=> CACHED [4/5] COPY . .                                        0.0s
=> CACHED [5/5] RUN python make-data.py                         0.0s
=> exporting to image                                           0.0s
=> exporting layers                                             0.0s
=> writing image sha256:0bc8b85fd50ac6c0a24a939997fdb91b283f8ee3fcc78becd5f6611571ae0e44 0.0s
=> naming to docker.io/library/alexisfr-virtu-seed-data:cmd     0.0s
```

#### - result

```
alexisfr7@alex:~/esiea-manuelle$ docker build -t alexisfr-virtu-result:cmd ./result/
[+] Building 2.0s (12/12) FINISHED
=> [internal] load build definition from Dockerfile              docker:default 0.1s
=> => transferring dockerfile: 525B                              0.0s
=> [internal] load .dockerignore                                0.1s
=> => transferring context: 54B                                  0.0s
=> [internal] load metadata for docker.io/library/node:18-slim 1.4s
=> [1/7] FROM docker.io/library/node:18-slim@sha256:fe687021c06383a2bc5eafa6db29b627ed28a55f6bdfbcea108f0c624b783c37 0.0s
=> [internal] load build context                                0.1s
=> => transferring context: 593B                                   0.0s
=> CACHED [2/7] RUN apt-get update && apt-get install -y --no-install-recommends curl tini && rm -rf /var/lib 0.0s
=> CACHED [3/7] WORKDIR /usr/local/app                          0.0s
=> CACHED [4/7] RUN npm install -g nodemon                      0.0s
=> CACHED [5/7] COPY package*.json ./                          0.0s
=> CACHED [6/7] RUN npm ci && npm cache clean --force && mv /usr/local/app/node_modules /node_modules 0.0s
=> CACHED [7/7] COPY . .                                        0.0s
=> exporting to image                                           0.0s
=> exporting layers                                             0.0s
=> writing image sha256:e3837101ed6d57b97d6dcd001280db69cb424762a5a8cb05624339084106db37 0.0s
=> naming to docker.io/library/alexisfr-virtu-result:cmd        0.0s
```

#### - worker

```
alexisfr7@alex:~/esiea-manuelle$ docker build -t alexisfr-virtu-worker:cmd ./worker/
[+] Building 0.4s (16/16) FINISHED
=> [internal] load build definition from Dockerfile              docker:default 0.1s
=> => transferring dockerfile: 1.04kB                            0.0s
=> [internal] load .dockerignore                                0.0s
=> => transferring context: 2B                                  0.0s
=> [internal] load metadata for mcr.microsoft.com/dotnet/runtime:7.0 0.2s
=> [internal] load metadata for mcr.microsoft.com/dotnet/sdk:7.0 0.2s
=> [build 1/7] FROM mcr.microsoft.com/dotnet/sdk:7.0@sha256:4be8ff7cb847f9e7ea1ac194920b0a6d41956af89f934b61a2ce64dc8 0.0s
=> [stage-1 1/3] FROM mcr.microsoft.com/dotnet/runtime:7.0@sha256:b41a241da8624e6554dd83cbcc642152f10a751082d1ea1a91 0.0s
=> [internal] load build context                                0.0s
=> => transferring context: 95B                                   0.0s
=> CACHED [stage-1 2/3] WORKDIR /app                             0.0s
=> CACHED [build 2/7] RUN echo "I am running on linux/amd64, building for linux/amd64" 0.0s
=> CACHED [build 3/7] WORKDIR /source                           0.0s
=> CACHED [build 4/7] COPY *.csproj .                            0.0s
=> CACHED [build 5/7] RUN dotnet restore -a amd64                0.0s
=> CACHED [build 6/7] COPY . .                                    0.0s
=> CACHED [build 7/7] RUN dotnet publish -c release -o /app -a amd64 --self-contained false --no-restore 0.0s
=> CACHED [stage-1 3/3] COPY --from=build /app .                 0.0s
=> exporting to image                                           0.0s
=> exporting layers                                             0.0s
=> writing image sha256:5ab2fa9d987697a46f2c8106c811625e4e3442bed5cf913fa130fbd90ab1e61f 0.0s
=> naming to docker.io/library/alexisfr-virtu-worker:cmd        0.0s
```



- Enfin, nous avons créé et exécuté les conteneurs grâce à la commande « `run` », qui a construit et lancé les conteneurs Docker.

```
alexisfr7@alex:~/esiea-manuelle$ docker run \
--network=cats-or-dogs-network \
--ip 172.23.0.2 \
--name vote \
-v "$(pwd)/vote:/usr/local/app" \
-p 5002:80 \
-d \
alexisfr-virtu-vote:cmd
16e8d15e2cac52fa3c0dd03bcff27b3f99b470fa86449d96e585482388fbf11c
```

**vote**

```
alexisfr7@alex:~/esiea-manuelle$ docker run \
--network=cats-or-dogs-network \
--ip 172.23.0.6 \
--name worker \
-d \
alexisfr-virtu-worker:cmd
f75d2fa9d5f939b0a28261463cbc828e0cd8547660b6d184f05417bc9182eab5
```

**worker**

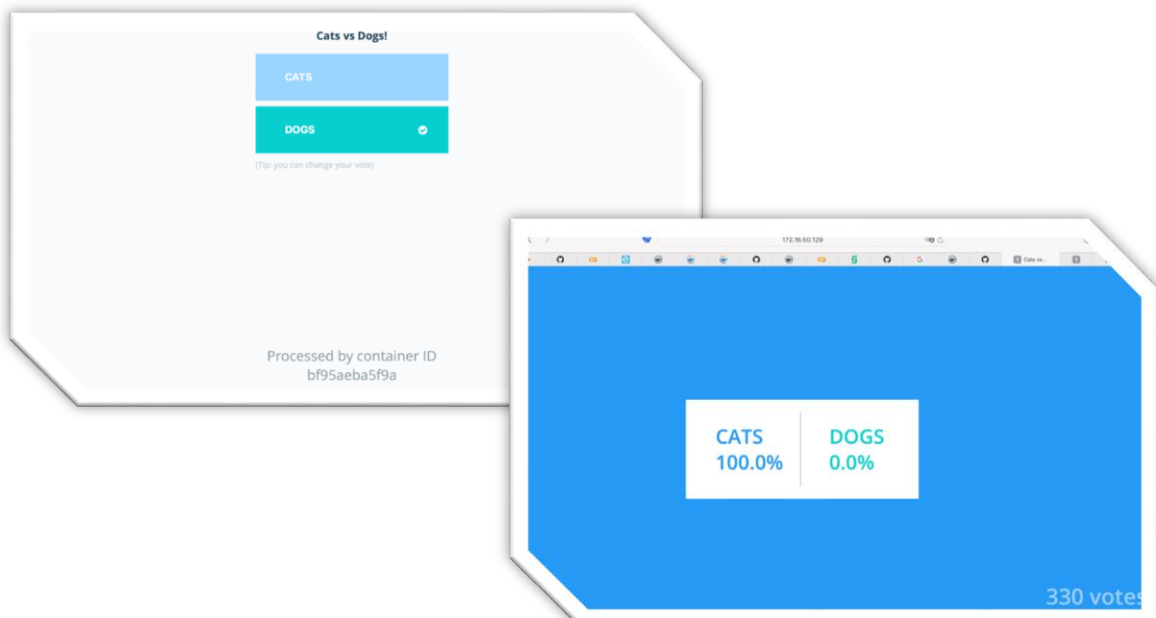
```
alexisfr7@alex:~/esiea-manuelle$ docker run \
--network=cats-or-dogs-network \
--ip 172.23.0.5 \
--name result \
--health-cmd="curl -f http://localhost/ || exit 1" \
--health-interval="30s" \
-p 5001:80 \
-p 127.0.0.1:9229:9229 \
-d \
alexisfr-virtu-result:cmd
d17fd202a1a143634a3c48b9b0a440b25f7fc808cfe851387b00903391fef249
```

**result**

```
alexisfr7@alex:~/esiea-manuelle$ docker run \
--network=cats-or-dogs-network \
--ip 172.23.0.7 \
--name seed-data \
-d \
alexisfr-virtu-seed-data:cmd
74b017f272f38427217059e05c4745318099b4fd7d57a2349a744e6aa21e4731
```

**seed-data**

- L’affichage de l’application sur internet est la même que pour la version avec Docker file



- Pour stopper l'exécution des conteneurs, il suffit simplement d'utiliser la commande « *docker stop* » pour arrêter tous les services en cours.

```
alexisfr7@alex:~/esiea-manuelle$ docker stop db redis result vote worker
db
redis
result
vote
worker
```

#### IV. Conclusion : les enseignements tirés

Ce projet nous a permis d'explorer les avantages et les inconvénients de l'utilisation des conteneurs Docker pour la virtualisation de services. Nous avons pu constater que cette technologie offre une solution efficace pour orchestrer les différents services à l'aide de Docker Compose. Les difficultés rencontrées au cours du projet nous ont notamment aidé à mieux comprendre les dépendances et surtout l'importance des bonnes configurations lors de la communication avec les bases de données. Un défaut d'inattention ou un souci dans l'organisation peut vite mener à l'erreur. C'est pourquoi la documentation fut un point fondamental, que ce soit sur Internet ou bien dans les READ ME/Docker Hub, ces ressources étaient indispensables à la résolution de nos problèmes spécifiques. Il est normal de ne pas parvenir à la réalisation parfaite d'un déploiement du premier coup et ces expériences ont souligné l'importance de la curiosité, la compréhension d'un problème, mais également de la méthodologie dans le processus de débogage. Ce projet nous a finalement permis d'acquérir de nouvelles compétences en matière de virtualisation et de développement d'applications web, ce qui nous sera certainement utile dans notre futur professionnel.