Groupe 47
Sébastien CARLUER & Edvans SOUPLET & Adrien VASSEUR

Virtualisation

Application de vote

Sommaire

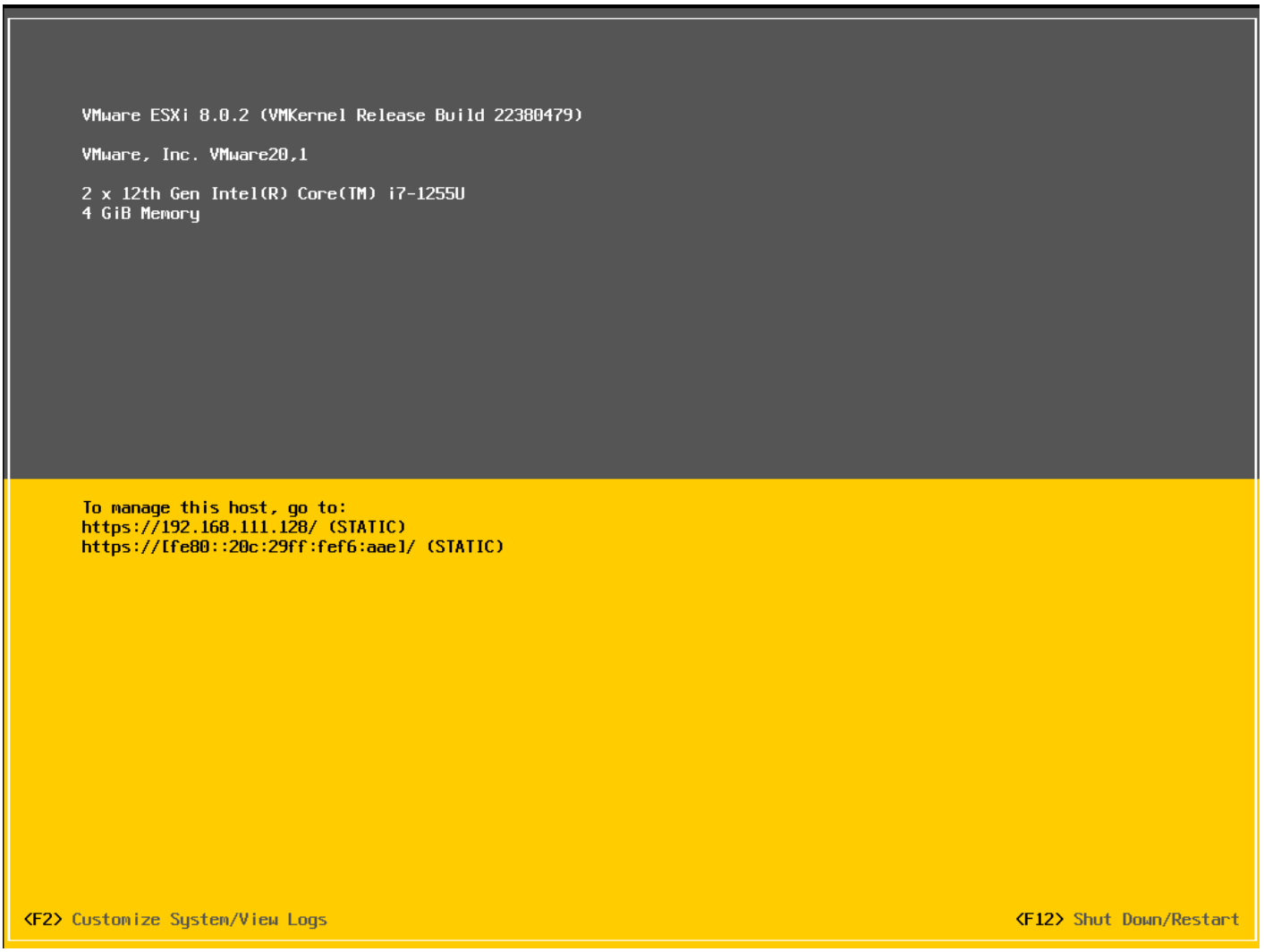
1. Introduction

Ce projet a été réalisé dans le cadre de nos cours à l'ESIEA en 4e année. L'objectif de celui-ci est de déployer une application de vote via docker. Pour cela nous avons utilisé un hyperviseur ESXi 8, ainsi qu'un serveur ubuntu 22.04

2. Mise en place

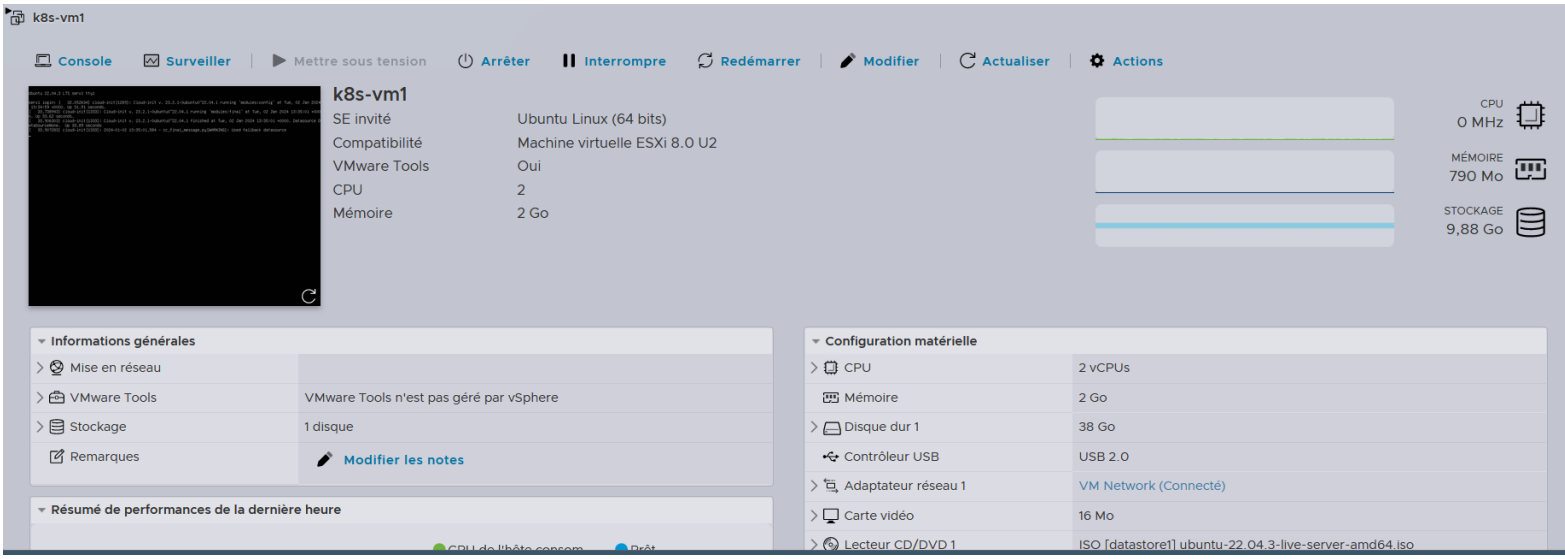
2.1. ESXI

La mise en place du projet de virtualisation a débuté par l'installation d'ESXi sur une machine virtuelle créée à l'aide de VMware Workstation, une plateforme de virtualisation. ESXi, le système d'exploitation de virtualisation de VMware, permet de créer et de gérer plusieurs machines virtuelles sur un seul serveur physique.
Cette approche nous offre un environnement sécurisé et isolé pour expérimenter la virtualisation, nous permettant ainsi d'explorer les fonctionnalités avancées de la virtualisation et de tester différents scénarios sans affecter l'infrastructure physique.



2.2. Machine virtuelle linux

Une fois ESXi installé et configuré avec succès, nous avons accédé à l'interface de gestion d'ESXi, accessible via un navigateur web, nous avons pu créer une nouvelle machine virtuelle.
Cette étape facilite la création d'un environnement virtuel adapté à nos besoins, offrant une flexibilité pour adapter les ressources matérielles selon les exigences du projet de virtualisation tels que la configuration matérielle (comme la quantité de mémoire RAM et le nombre de processeurs virtuels), le stockage de la machine virtuelle sur le serveur, ainsi que le choix du système d'exploitation Linux spécifique à installer.
Ici, nous avons déployé un serveur Ubuntu 22 avec comme adaptateur réseau VM Network, car le vSwitch rencontrait des anomalies.



3. Déploiement manuel

Nous avons récupéré avec succès les fichiers Docker depuis le référentiel Git "esiea-ressources", à l'aide de la commande:

git clone https://github.com/pascalito007/esiea-ressources.git

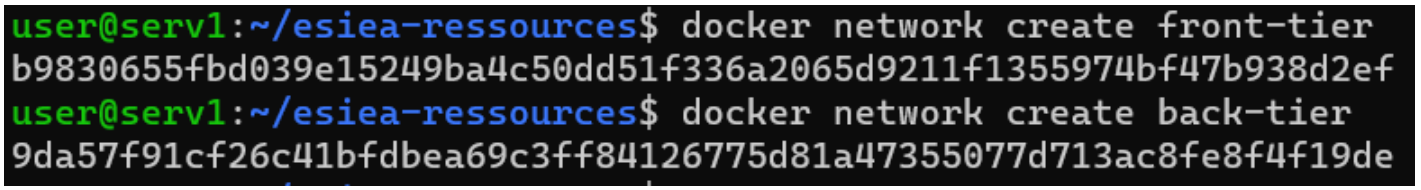
Ce référentiel contenait des instructions détaillées pour la configuration de différents services à l'aide de fichiers Docker.

```
user@serv1:~$ git clone https://github.com/pascalito007/esiea-ressources.git
Cloning into 'esiea-ressources'...
remote: Enumerating objects: 50, done.
remote: Counting objects: 100% (50/50), done.
remote: Compressing objects: 100% (41/41), done.
remote: Total 50 (delta 3), reused 46 (delta 2), pack-reused 0
Receiving objects: 100% (50/50), 228.19 KiB | 4.39 MiB/s, done.
Resolving deltas: 100% (3/3), done.
user@serv1:~$ ls
composetest  dockerfile  esiea-ressources  startbootstrap-grayscale
user@serv1:~$ cd esiea-ressources/
user@serv1:~/esiea-ressources$ ls
architecture.png  healthchecks  README-49-47.md  README.md  result  seed-data  vote  worker
user@serv1:~/esiea-ressources$
```

Pour répondre aux exigences du projet, deux réseaux distincts ont été créés :

1. front-tier: Ce réseau est destiné à exposer les services vers l'extérieur, permettant une accessibilité depuis l'extérieur du système. Les services comme le service Vote et le service Result font partie de ce réseau.

2. back-tier:Le réseau back-tier est réservé aux services qui ne nécessitent pas d'exposition directe à l'extérieur. Cela inclut des services tels que le service Worker, la base de données (db), Redis, etc.



Ces réseaux distincts permettent une isolation claire entre les services exposés et les services internes, assurant ainsi une sécurité et une gestion efficace des flux de données au sein de l'architecture distribuée.

La commande docker build -t worker est utilisée pour créer une nouvelle image Docker à partir d'un fichier Dockerfile présent dans le répertoire actuel. Voici ce que font les différents éléments de la commande :

- docker build: C'est la commande principale de Docker pour construire des images à partir des instructions spécifiées dans un fichier Dockerfile.
- -t worker|seed-data|vote|result: Ces argument permettent de taguer ou nommer l'image Docker qui sera créée.

L'exécution de cette commande dans le répertoire contenant le Dockerfile, permet à Docker de lire les instructions du Dockerfile pour créer une image Docker. Cette image sera nommée "worker" et pourra être utilisée pour créer des conteneurs docker basés sur cette image spécifique.

```
user@serv1:~/esiea-ressources/worker$ docker build -t worker .
[+] Building 14.2s (5/15)                                       docker:default
 => [internal] load .dockerignore                                          0.1s
 => => transferring context: 2B                                            0.0s
 => [internal] load build definition from Dockerfile                       0.1s
 => => transferring dockerfile: 1.04kB                                     0.0s
 => [internal] load metadata for mcr.microsoft.com/dotnet/sdk:7.0          9.2s
 => [internal] load metadata for mcr.microsoft.com/dotnet/runtime:7.0      9.2s
 => [build 1/7] FROM mcr.microsoft.com/dotnet/sdk:7.0@sha256:4be8ff7c      4.8s
 => => resolve mcr.microsoft.com/dotnet/sdk:7.0@sha256:4be8ff7cb847f9      0.0s
 => => sha256:2df043b0ae9ae50ace61efc4f3cf2ea22f9e938 2.01kB / 2.01kB      0.0s
 => => sha256:ccb4ba5bb726748e965e7730afcb6ab92eb14 32.46MB / 32.46MB      3.2s
 => => sha256:4be8ff7cb847f9e7ea1ac194920b0a6d41956af 1.79kB / 1.79kB      0.0s
 => => sha256:d5cee0eb99f0276461c1016534119d3df7044bf 5.31kB / 5.31kB      0.0s
 => => sha256:b5a0d5c14ba9ece1eecd5137c468d9a123372 31.42MB / 31.42MB      2.2s
 => => sha256:4ece0626219de44070331daf1eff6932a03a3 14.97MB / 14.97MB      0.8s
 => => sha256:bdf2c62d9548601f6df118ad7ddf984e15a0aa258cc 155B / 155B      2.1s
 => => sha256:d2e769e5b08ad6a58b48f965619b84b2a3d47 10.12MB / 10.12MB      3.2s
 => => sha256:d41336b5e4674fbb04b625b2794c9f38eee0c 16.20MB / 25.38MB      4.8s
 => => extracting sha256:b5a0d5c14ba9ece1eecd5137c468d9a123372b0af2ed      2.4s
 => => sha256:0f0fbcade3825e1d159fbcc6d9b3910e65de 13.63MB / 180.94MB      4.8s
 => => sha256:4492877723c1b5f4cc37c87dc59d8270f8a81 13.19MB / 13.97MB      4.8s
 => [stage-1 1/3] FROM mcr.microsoft.com/dotnet/runtime:7.0@sha256:b4      4.8s
 => => resolve mcr.microsoft.com/dotnet/runtime:7.0@sha256:b41a241da8      0.0s
 => => sha256:4ece0626219de44070331daf1eff6932a03a3 14.97MB / 14.97MB      0.8s
 => => sha256:ccb4ba5bb726748e965e7730afcb6ab92eb14 32.46MB / 32.46MB      3.2s
 => => sha256:b41a241da8624e65544dd83cbcc642152f10a75 1.79kB / 1.79kB      0.0s
 => => sha256:198cbf45efc0a2a5d64766ec547cb7bfa859b6e 1.16kB / 1.16kB      0.0s
 => => sha256:337945a71cfdb1635ab48144281b3575dd6726b 1.92kB / 1.92kB      0.0s
 => => sha256:b5a0d5c14ba9ece1eecd5137c468d9a123372 31.42MB / 31.42MB      2.2s
 => => sha256:bdf2c62d9548601f6df118ad7ddf984e15a0aa258cc 155B / 155B      2.1s
 => => extracting sha256:b5a0d5c14ba9ece1eecd5137c468d9a123372b0af2ed      2.4s
 => [internal] load build context                                          0.0s
 => => transferring context: 7.07kB                                        0.0s
```

```
 => exporting to image                                                     0.1s
 => => exporting layers                                                    0.1s
 => => writing image sha256:5aa9b2448c21e0dd710de3820ce28845d1d851e68      0.0s
 => => naming to docker.io/library/worker                                  0.0s
user@serv1:~/esiea-ressources/worker$
```

Construction de seed-data via le docker File :

```
user@serv1:~/esiea-ressources/seed-data$ docker build -t seed-data .
[+] Building 35.8s (10/10) FINISHED                              docker:default
 => [internal] load .dockerignore                                          0.1s
 => => transferring context: 2B                                            0.0s
 => [internal] load build definition from Dockerfile                       0.1s
 => => transferring dockerfile: 346B                                       0.0s
```

construction du dockerFIle pour vote :

```
user@serv1:~/esiea-ressources/vote$ docker build -t vote .
[+] Building 47.1s (11/11) FINISHED                              docker:default
 => [internal] load build definition from Dockerfile                       0.1s
 => => transferring dockerfile: 1.09kB                                     0.0s
 => [internal] load .dockerignore                                          0.0s
 => => transferring context: 2B                                            0.0s
 => [internal] load metadata for docker.io/library/python:3.11-slim        8.0s
 => [base 1/5] FROM docker.io/library/python:3.11-slim@sha256:8f64a67710   4.7s
 => => resolve docker.io/library/python:3.11-slim@sha256:8f64a67710f3d98   0.0s
 => => sha256:b05f8bf97bacb24c0f8ddcd5ea6764c49aff2a325c5007 241B / 241B   0.4s
 => => sha256:b5a30bc7f1f876ea3d8555c4c65117eee16c4e6224 3.39MB / 3.39MB   0.9s
 => => sha256:8f64a67710f3d981cf3008d6f9f1dbe61accd7927f 1.65kB / 1.65kB   0.0s
 => => sha256:233e4c53aebbd019c06f6d927b93de5292cfc5091a 1.37kB / 1.37kB   0.0s
 => => sha256:dd150e5400f1a756b7752931d3064a57f529325b74 6.93kB / 6.93kB   0.0s
 => => sha256:6ac7ac839d9ddb5b76a12d1c4e07d66079748ed2 12.84MB / 12.84MB   0.9s
 => => extracting sha256:6ac7ac839d9ddb5b76a12d1c4e07d66079748ed2dcdc51a   2.3s
 => => extracting sha256:b05f8bf97bacb24c0f8ddcd5ea6764c49aff2a325c50071   0.0s
 => => extracting sha256:b5a30bc7f1f876ea3d8555c4c65117eee16c4e62241a1a6   1.1s
 => [internal] load build context                                          0.0s
 => => transferring context: 6.46kB                                        0.0s
 => [base 2/5] RUN apt-get update &&     apt-get install -y --no-instal   18.5s
 => [base 3/5] WORKDIR /usr/local/app                                      0.1s
 => [base 4/5] COPY requirements.txt ./requirements.txt                    0.1s
 => [base 5/5] RUN pip install --no-cache-dir -r requirements.txt         14.8s
 => [final 1/1] COPY . .                                                   0.2s
 => exporting to image                                                     0.5s
 => => exporting layers                                                    0.5s
 => => writing image sha256:064b94a7f7a7b33f89a03eccaa5cbfef962c6c2e1c45   0.0s
 => => naming to docker.io/library/vote                                    0.0s
```

Construction du DockerFile pour result :

```
user@serv1:~/esiea-ressources/result$ docker build -t result .
[+] Building 140.9s (12/12) FINISHED                             docker:default
 => [internal] load build definition from Dockerfile                       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            8.1s
 => [1/7] FROM docker.io/library/node:18-slim@sha256:fe687021c06383a      11.1s
 => => resolve docker.io/library/node:18-slim@sha256:fe687021c06383a2      0.0s
 => => sha256:8e04602828ddd8c394c701f265c048f2a7cf9cb 1.37kB / 1.37kB      0.0s
 => => sha256:d3cce7487840f2783532a37fca9e79c4d55fea3 7.62kB / 7.62kB      0.0s
 => => sha256:ebd7ac832a7e4bd07f5ad6f4bbc6db1f5b02de7 3.36kB / 3.36kB      0.4s
 => => sha256:9d5fc5b38df6ebdd70895cf78cd4e3de9aef9 38.57MB / 38.57MB      5.1s
 => => sha256:e4cb19787d8cf84786456582936c59a7e23c541 2.67MB / 2.67MB      0.8s
 => => sha256:fe687021c06383a2bc5eafa6db29b627ed28a55 1.21kB / 1.21kB      0.0s
 => => extracting sha256:ebd7ac832a7e4bd07f5ad6f4bbc6db1f5b02de7a8e07      0.0s
 => => sha256:6f59ea1fe6ede7c1fe4ad178ff96c351c02b84f6a0a 452B / 452B      1.5s
 => => extracting sha256:9d5fc5b38df6ebdd70895cf78cd4e3de9aef9dc55ed5      4.9s
 => => extracting sha256:e4cb19787d8cf84786456582936c59a7e23c541a2906      0.4s
 => => extracting sha256:6f59ea1fe6ede7c1fe4ad178ff96c351c02b84f6a0a7      0.0s
 => [internal] load build context                                          0.1s
 => => transferring context: 278.98kB                                      0.0s
 => [2/7] RUN apt-get update &&     apt-get install -y --no-install-   14.5s
```

La commande docker run est utilisée pour exécuter un conteneur Docker à partir d'une image spécifique.

- --name db: donne un nom au conteneur.
- -e POSTGRES_PASSWORD=password: définit une variable d'environnement dans le conteneur PostgreSQL. Dans ce cas, elle définit la variable d'environnement POSTGRES_PASSWORD avec la valeur "password", ce qui sera utilisé comme mot de passe pour l'utilisateur administrateur par défaut dans PostgreSQL.
- -d: indique à Docker de lancer le conteneur en arrière-plan (mode détaché).
- postgres:15-alpine: Argument final de la commande docker run. Il spécifie l'image à partir de laquelle le conteneur sera lancé. Plus précisément, c'est l'image postgres avec la version 15-alpine provenant du Docker Hub. Cette image contient PostgreSQL version 15 basée sur Alpine Linux.

Cette commande lance donc un conteneur Docker à partir de l'image PostgreSQL 15 Alpine en utilisant un mot de passe spécifié pour l'utilisateur administrateur de la base de données PostgreSQL. Une base de données PostgreSQL fonctionnelle serait démarrée dans ce conteneur, accessible localement à l'aide du nom du conteneur ou via le port spécifié par PostgreSQL, généralement le port par défaut 5432.

creation de la base de donnée via DockerHub (https://hub.docker.com/_/postgres) :

```
user@serv1:~/esiea-ressources/result$ docker run --name db -e POSTGRES_PASSW
ORD=password -d postgres:15-alpine
Unable to find image 'postgres:15-alpine' locally
15-alpine: Pulling from library/postgres
661ff4d9561e: Already exists
e4a3f96ea8e5: Pull complete
0c1e2e159ea1: Pull complete
26c071a8426e: Pull complete
e9a1ba05d22c: Pull complete
efc39a79d7dc: Pull complete
72124e665f9e: Pull complete
aa569f3e770e: Pull complete
86d5fe07cb37: Pull complete
Digest: sha256:e2a22801fcab638f9491039f8257e9f719ab02e8c78c6a6f2c0349505f92d
c35
Status: Downloaded newer image for postgres:15-alpine
113499bf565c2c524f4ff9718b538c2491040b75cffa1a066a7f1961bb4b3768
user@serv1:~/esiea-ressources/result$
```

connecter la base de donnée à notre réseau back-tier:
docker network connect back-tier db
il faut également le déconnecter du bridge :
docker network disconnect bridge db

Pour toutes les commandes docker run suivante la même structure est appliquée. On crée un conteneur Docker utilisant différentes images.

- docker run: lancer un conteneur à partir d'une image Docker.
- --name redis|worker|vote|result: permet de nommer le conteneur créé.
- -d: exécute le conteneur en arrière-plan.
- 'argument final': spécifie l'image avec laquelle le conteneur sera lancé.
- -p 5002:80: mappe le port 5002 de l'hôte au port 80 du conteneur. Cela signifie que les requêtes envoyées à notre machine sur le port 5002 seront redirigées vers le port 80 du conteneur.

installation et mise en réseau de Redis :

```
user@serv1:~/esiea-ressources/result$ docker run --name redis -d redis
Unable to find image 'redis:latest' locally
latest: Pulling from library/redis
af107e978371: Already exists
b031def5f2c4: Pull complete
bf7f0c8796d3: Pull complete
e3b2691a4104: Pull complete
190b4d7a237a: Pull complete
797591c7970a: Pull complete
4f4fb700ef54: Pull complete
45ce3854ac9a: Pull complete
Digest: sha256:a7cee7c8178ff9b5297cb109e6240f5072cdaaafd775ce6b586c3c704b064
58e
Status: Downloaded newer image for redis:latest
872d4e4e685f9e9324b782aff5ebe9b09c360af0fac8bd79f4ccbf24a53a73f0
user@serv1:~/esiea-ressources/result$ docker net^C
user@serv1:~/esiea-ressources/result$ docker network disconnect bridge redis
user@serv1:~/esiea-ressources/result$ docker network connect back-tier redis
user@serv1:~/esiea-ressources/result$
```

lancement et mise en réseau de worker :

```
user@serv1:~/esiea-ressources/result$ docker run --name worker -d worker
e7a6e58c94360b573ecd8589fa8523fade351a3c64034d201e016f1ee2b22820
user@serv1:~/esiea-ressources/result$
user@serv1:~/esiea-ressources/result$ docker network connect back-tier worker
user@serv1:~/esiea-ressources/result$ docker network disconnect bridge worker
```

lancement et mise en réseau de vote :

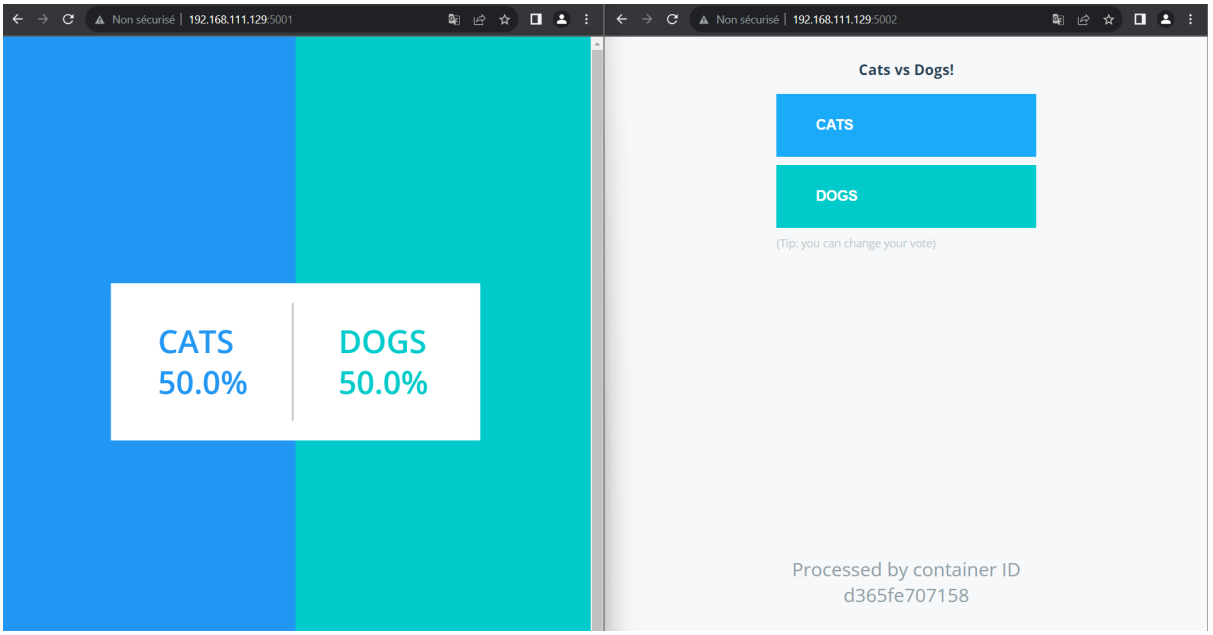
```
user@serv1:~/esiea-ressources/vote$ docker run --name vote -p 5002:80 -d vote
d365fe7071588d1b5f4dc7cf56215c09ee165af78c8467099ee599a540b07299
user@serv1:~/esiea-ressources/vote$ docker network connect front-tier vote
user@serv1:~/esiea-ressources/vote$ docker network connect back-tier vote
user@serv1:~/esiea-ressources/vote$ docker network disconnect bridge vote
```

lancement et mise en reseau du result :

```
user@serv1:~/esiea-ressources/vote$ docker run --name results -p 5001:80 -d res
ult
cffed8c6abc2a5159eab4482166f2b8b4b2d5a0b7975b6d0ef3d97dde42a8e6d
user@serv1:~/esiea-ressources/vote$ docker network disconnect bridge results
user@serv1:~/esiea-ressources/vote$ docker network connect back-tier results
```

Liste des dockers présents :

```
user@serv1:~/esiea-ressources/vote$ docker ps
CONTAINER ID   IMAGE                COMMAND                  CREATED          STATUS          PORTS                                       NAMES
d365fe707158   vote                 "gunicorn app:app -b…"   53 seconds ago   Up 52 seconds   0.0.0.0:5002->80/tcp, :::5002->80/tcp       vote
cffed8c6abc2   result               "/usr/bin/tini -- no…"   4 minutes ago    Up 3 minutes    0.0.0.0:5001->80/tcp, :::5001->80/tcp       results
e7a6e58c9436   worker               "dotnet Worker.dll"      17 minutes ago   Up 17 minutes                                               worker
872d4e4e685f   redis                "docker-entrypoint.s…"   20 minutes ago   Up 20 minutes   6379/tcp                                    redis
113499bf565c   postgres:15-alpine   "docker-entrypoint.s…"   30 minutes ago   Up 30 minutes   5432/tcp                                    db
user@serv1:~/esiea-ressources/vote$
```



4. Déploiement avec Docker Compose

Déployer avec Docker Compose, en utilisant la commande docker-compose up -d, permet de lancer des applications multi-conteneurs définies dans un fichier YAML appelé "docker-compose.yml". Cette commande démarre tous les services spécifiés dans ce fichier.

Docker Compose est un outil qui simplifie le déploiement d'applications composées de plusieurs conteneurs Docker interagissant entre eux.

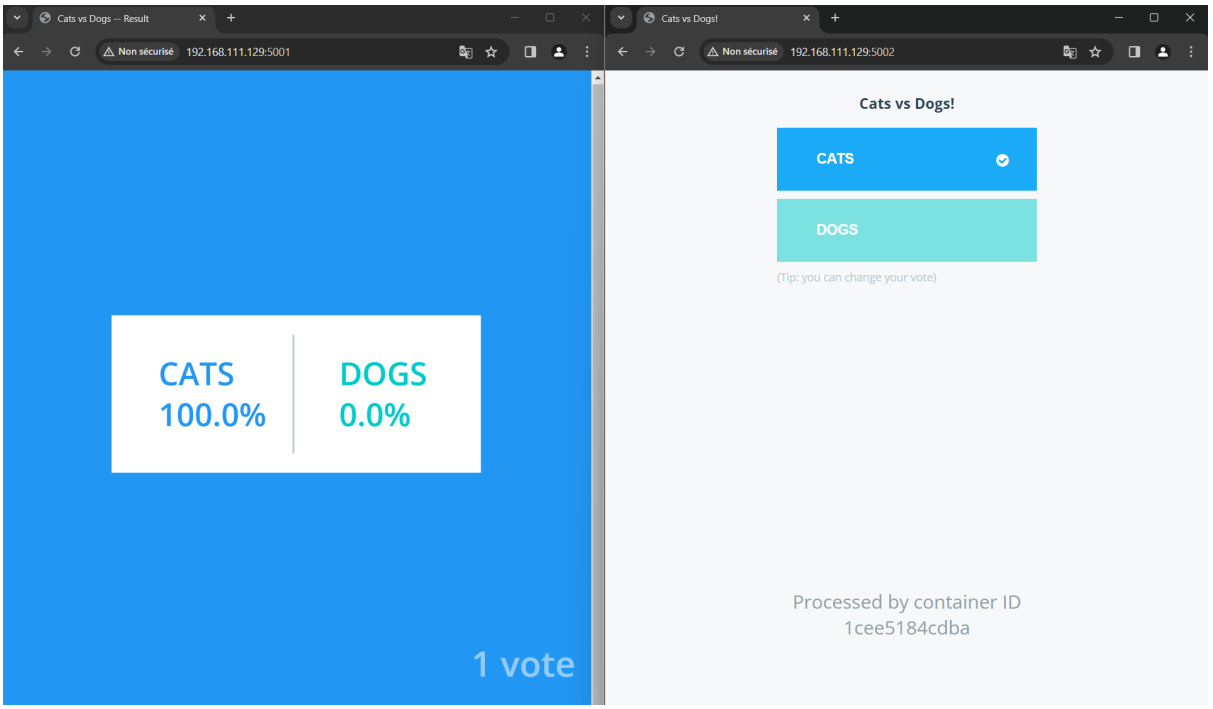
Le fichier "docker-compose.yml" décrit la configuration des services nécessaires pour votre application, y compris les images Docker, les ports exposés, les volumes partagés, les variables d'environnement, etc.

A l'exécution, Docker Compose lit le fichier de configuration, crée les réseaux et les volumes nécessaires, puis lance les conteneurs pour chaque service spécifié dans le fichier. Les conteneurs sont lancés en arrière-plan, donc ne bloquent pas le terminal.

Cela simplifie grandement le déploiement d'applications complexes, car au lieu de gérer chaque conteneur séparément, Docker Compose permet de gérer et de coordonner plusieurs conteneurs en une seule commande.

L'application déployée via Docker Compose

```
user@serv1:~/esiea-ressources$ docker compose up -d
[+] Running 6/6
 ✓ Network esiea-ressources_cats-or-dogs-network  Created
 ✓ Container esiea-ressources-redis-1             Healthy
 ✓ Container esiea-ressources-db-1                Healthy
 ✓ Container esiea-ressources-vote-1              Started
 ✓ Container esiea-ressources-worker-1            Started
 ✓ Container esiea-ressources-result-1            Started
```




```
user@serv1:~/esiea-ressources$ docker ps
CONTAINER ID   IMAGE                    COMMAND                  CREATED          STATUS                    PORTS
               NAMES
1cee5184cdba   esiea-ressources-vote     "python app.py"          34 seconds ago   Up 31 seconds (healthy)   0.0.0.0:5002->80/tcp, :::5002->80/tcp
               esiea-ressources-vote-1
2159b4b29713   esiea-ressources-result   "nodemon --inspect=0…"   3 minutes ago    Up 3 minutes              127.0.0.1:9229->9229/tcp, 0.0.0.0:5001->80/tcp,
 :::5001->80/tcp   esiea-ressources-result-1
a3b769697d68   esiea-ressources-worker   "dotnet Worker.dll"      3 minutes ago    Up 3 minutes
               esiea-ressources-worker-1
5bb17dfeaaa7   redis:alpine             "docker-entrypoint.s…"   3 minutes ago    Up 3 minutes (healthy)    6379/tcp
               esiea-ressources-redis-1
3ebe7263bdf5   postgres:15-alpine       "docker-entrypoint.s…"   3 minutes ago    Up 3 minutes (healthy)    5432/tcp
               esiea-ressources-db-1
user@serv1:~/esiea-ressources$
```

On peut voir que l'ID du conteneur afficher sur le site et dans la VM sont identiques
On utilise bien le port 5002 et 5001. On est également bien sur un réseau unique cats-or-dogs-network

Le lien de notre projet github : https://github.com/EdvansSPT/ESIEA-cat_or_dogs-SOUPLET-CARLUER-VASSEUR