

HUMAN'S BEST FRIENDS

Cats or Dogs ?



Sommaire :

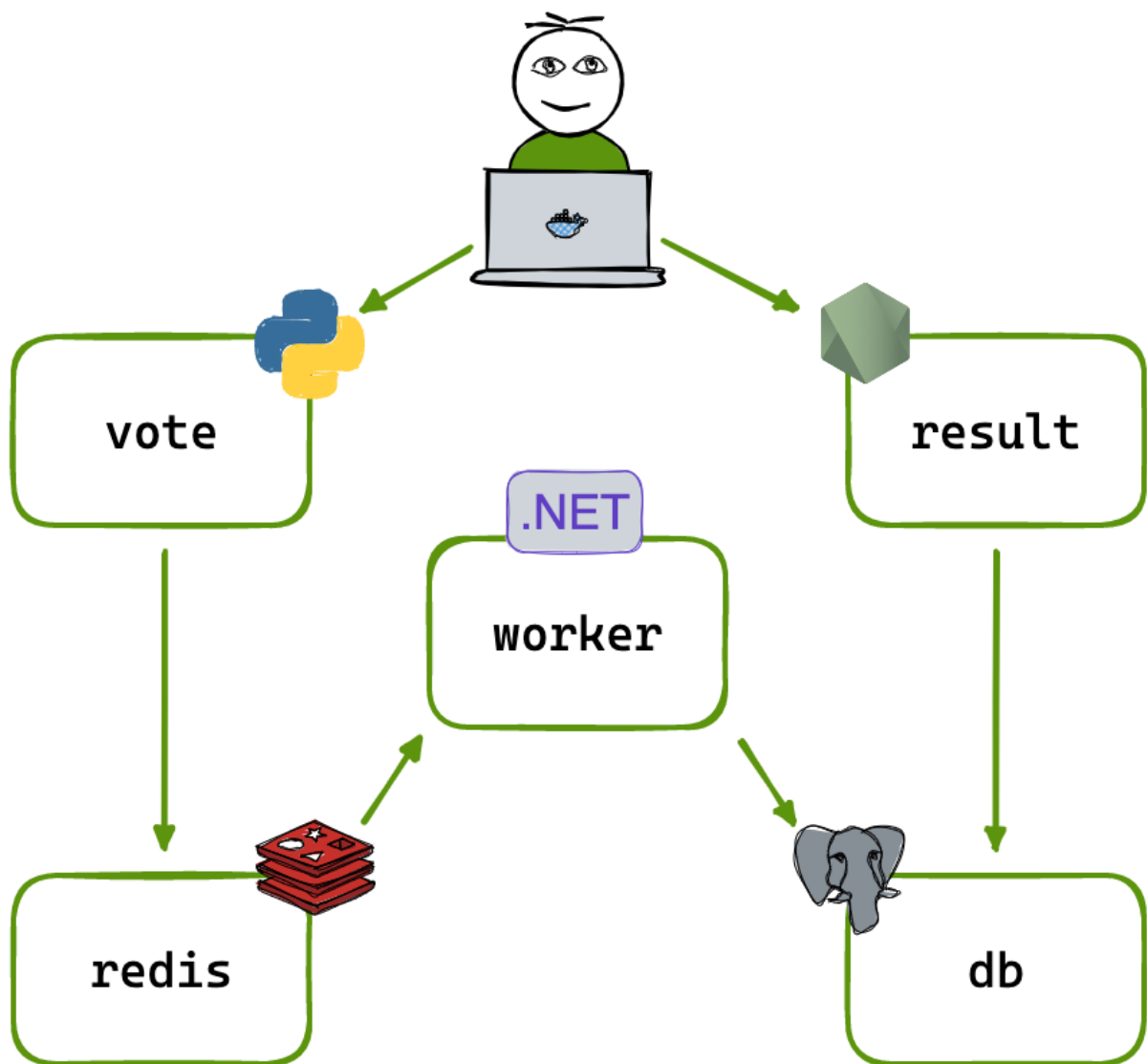
Introduction.....	3
Architecture de l'application.....	3
Docker :	4
Copie du git et creation des images.....	4
Création des fichier yml.....	6
Construction et démarrage de Docker.....	8
Lancement.....	9
Kubernetes	10
Installation et lancement de Kubernetes :	10
Installation et lancement de minikube :.....	10
Conclusion	11
Annexes :	12
Liens :	12

Introduction

Dans le cadre de notre cursus consacré à la Virtualisation et à la Conteneurisation, notre équipe de trois membres s'est investie dans la conception d'une application intitulée "HumansBestFriend" dédiée au vote virtuel. Notre objectif central est d'explorer et de mettre en pratique les technologies de virtualisation et de conteneurisation en utilisant une machine virtuelle Ubuntu, exécutée via ESXi, tel que nous l'avons étudié au cours de nos travaux pratiques.

Notre démarche vise à mettre en œuvre les connaissances théoriques acquises dans le domaine de la virtualisation et de la conteneurisation, tout en offrant une expérience pratique et divertissante à travers une thématique aussi captivante et populaire que celle des meilleurs amis de l'homme.

Architecture de l'application



- Une application web frontale en Python qui vous permet de voter entre deux options
- Un Redis qui recueille de nouveaux votes Un worker .NET qui consomme les votes et les stocke
- Une base de données Postgres sauvegardée par un volume Docker
- Une application web Node.js qui affiche les résultats du vote en temps réel

Docker :

Copie du git et creation des images

\$ git clone <https://github.com/pascalito007/esiea-ressources.git>

Après avoir copié le GIT, on se place dans le repertoire afin de créer et lancer les images :

\$ docker build -t vote-app:latest .

```
root@Linux:/home/beber/esiea-ressources/vote# docker build -t application-vote:latest .
[+] Building 0.5s (11/11) FINISHED
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 1.09kB
=> [internal] load metadata for docker.io/library/python:3.11-slim
=> [base 1/5] FROM docker.io/library/python:3.11-slim@sha256:8f64a67710f3d981cf3008d6f9f1d6e61accd7927f165f4e37ea3f8b883ccc3f
=> [internal] load build context
=> => transferring context: 274B
=> CACHED [base 2/5] RUN apt-get update && apt-get install -y --no-install-recommends curl && rm -rf /var/lib/apt/lists/*
=> CACHED [base 3/5] WORKDIR /usr/local/app
=> CACHED [base 4/5] COPY requirements.txt ./requirements.txt
=> CACHED [base 5/5] RUN pip install --no-cache-dir -r requirements.txt
=> CACHED [final 1/1] COPY . .
=> exporting to image
=> => exporting layers
=> => writing image sha256:9abf8ba8d267fe58c433c743ed952f72675792d7bce3cdc3f0ce833784f9b3ba
=> => naming to docker.io/library/application-vote:latest
```

\$ docker build -t result-app:latest .

```
root@Linux:/home/beber/esiea-ressources/result# docker build -t application-result:latest .
[+] Building 15.8s (12/12) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 525B
=> [internal] load .dockerignore
=> => transferring context: 54B
=> [internal] load metadata for docker.io/library/node:18-slim
=> [1/7] FROM docker.io/library/node:18-slim@sha256:fe687021c06383a2bc5eafa6db29b627ed28a55f6bdfbcea108f0c624b783c37
=> => resolve docker.io/library/node:18-slim@sha256:fe687021c06383a2bc5eafa6db29b627ed28a55f6bdfbcea108f0c624b783c37
=> => sha256:ebd7ac832a7e4bd07f5ad6f4bbc6db1f5b02de7a8e07b40627142ef61a9b1b9d 3.36kB / 3.36kB
=> => sha256:9d5fc5b38df6ebdd70895cf78cd4e3de9aef9dc55ed5c79c2945066661a0e8e4 38.57MB / 38.57MB
=> => sha256:e4cb19787d8cf84786456582936c59a7e23c541a29061d728df3f6c8923dd739 2.67MB / 2.67MB
=> => sha256:fe687021c06383a2bc5eafa6db29b627ed28a55f6bdfbcea108f0c624b783c37 1.21kB / 1.21kB
=> => sha256:8e04602828dd8c394c701f265c048f2a7cf9cbf112635ba26cec2d06936f17b 1.37kB / 1.37kB
=> => sha256:d3cce7487840f2783532a377fca9e79c4d55fea3d5ce3caf7c145596fcc457f8e 7.62kB / 7.62kB
=> => extracting sha256:ebd7ac832a7e4bd07f5ad6f4bbc6db1f5b02de7a8e07b40627142ef61a9b1b9d
=> => sha256:6f59ea1fe6ede7c1fe4ad178ff96c351c02b84f6a0a7d4f38dcba13d4642298b 452B / 452B
=> => extracting sha256:9d5fc5b38df6ebdd70895cf78cd4e3de9aef9dc55ed5c79c2945066661a0e8e4
=> => extracting sha256:e4cb19787d8cf84786456582936c59a7e23c541a29061d728df3f6c8923dd739
=> => extracting sha256:6f59ea1fe6ede7c1fe4ad178ff96c351c02b84f6a0a7d4f38dcba13d4642298b
=> [internal] load build context
=> => transferring context: 278.98kB
=> [2/7] RUN apt-get update && apt-get install -y --no-install-recommends curl tini && rm -rf /var/lib/apt/lists/*
=> [3/7] WORKDIR /usr/local/app
=> [4/7] RUN npm install -g nodemon
=> [5/7] COPY package*.json ./
=> [6/7] RUN npm ci && npm cache clean --force && mv /usr/local/app/node_modules /node_modules
=> [7/7] COPY . .
=> exporting to image
=> => exporting layers
=> => writing image sha256:6d42de31503d852c70dbaa2d2123b325d2e36f5db6b1b65c0471b346b6302493
=> => naming to docker.io/library/application-result:latest
```

\$ docker build -t worker-app:latest .

```
root@Linux:/home/beber/eslea-ressources/worker# docker build -t application-worker:latest .
[+] Building 36.7s (16/16) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 1.04kB
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for mcr.microsoft.com/dotnet/runtime:7.0
=> [internal] load metadata for mcr.microsoft.com/dotnet/sdk:7.0
=> [build 1/7] FROM mcr.microsoft.com/dotnet/sdk:7.0@sha256:4be8ff7cb847f9e7ea1ac194920b0a6d41956af89f934b61a2ce64dc8563471c
=> => resolve mcr.microsoft.com/dotnet/sdk:7.0@sha256:4be8ff7cb847f9e7ea1ac194920b0a6d41956af89f934b61a2ce64dc8563471c
=> => sha256:4ece0626219de44070331daf1eff6932a03a31333a6f7f2d7b8b592a2e80d5b0 14.97MB / 14.97MB
=> => sha256:4be8ff7cb847f9e7ea1ac194920b0a6d41956af89f934b61a2ce64dc8563471c 1.79kB / 1.79kB
=> => sha256:2df043b0a9e5a0ace1efc4f3cf2ea22f9e9389a9f823774d23f8968bb4cdd9 2.01kB / 2.01kB
=> => sha256:b5a0d5c14ba9ece1eecd5137c468d9a123372b0af2ed2c8c4446137730c90e5b 31.42MB / 31.42MB
=> => sha256:d5cee0eb99f0276461c1016534119d3df7044bfe23d002340c7313712bfc83cd 5.31kB / 5.31kB
=> => sha256:ccb4ba5bb726748e965e7730afc6ab92eb14745c7bcad9861d77d81b2372cfe 32.46MB / 32.46MB
=> => sha256:bd2c62d9548601f6df118ad7dd984e15a0aa258cc56b4b77945fa07a6978e7 155B / 155B
=> => sha256:d41336b5e4674fbb04b625b2794c9f38eeec549c63c3147df863043af106f9b 25.38MB / 25.38MB
=> => sha256:d2e769e5b08ada6a58b48f965619b84b2a3d47cbc0b15bb638aa29572bd097274 10.12MB / 10.12MB
=> => extracting sha256:b5a0d5c14ba9ece1eecd5137c468d9a123372b0af2ed2c8c4446137730c90e5b
=> => sha256:0f0fbcade3825e1d159fbc6d9b3910e65deadeaa651a2f6a4108c3d8234568b 180.94MB / 180.94MB
=> => sha256:4492877723c1b5f4cc37c87dc59d8270f8a81bd5976cd1af3a454695a14e12bc 13.97MB / 13.97MB
=> => extracting sha256:4ece0626219de44070331daf1eff6932a03a31333a6f7f2d7b8b592a2e80d5b0
=> => extracting sha256:ccb4ba5bb726748e965e7730afc6ab92eb14745c7bcad9861d77d81b2372cfe
=> => extracting sha256:bd2c62d9548601f6df118ad7dd984e15a0aa258cc56b4b77945fa07a6978e7
=> => extracting sha256:d2e769e5b08ada6a58b48f965619b84b2a3d47cbc0b15bb638aa29572bd097274
=> => extracting sha256:d41336b5e4674fbb04b625b2794c9f38eeec549c63c3147df863043af106f9b
=> => extracting sha256:0f0fbcade3825e1d159fbc6d9b3910e65deadeaa651a2f6a4108c3d8234568b
=> => extracting sha256:4492877723c1b5f4cc37c87dc59d8270f8a81bd5976cd1af3a454695a14e12bc
=> [internal] load build context
=> => transferring context: 7.07kB
=> [stage-1 1/3] FROM mcr.microsoft.com/dotnet/runtime:7.0@sha256:b41a241da8624e65544dd83bcc642152f10a751082d1ea1a912e238b8259533
=> => resolve mcr.microsoft.com/dotnet/runtime:7.0@sha256:b41a241da8624e65544dd83bcc642152f10a751082d1ea1a912e238b8259533
=> => sha256:198cbf45efc0a25d64766ec547cb7fa859b0e3a178543558a1beb350eb09bf 1.16kB / 1.16kB
=> => sha256:337945a71cfdb1635ab48144281b3575dd6726b6568343e91d0f711ab07dda5e 1.92kB / 1.92kB
=> => sha256:b5a0d5c14ba9ece1eecd5137c468d9a123372b0af2ed2c8c4446137730c90e5b 31.42MB / 31.42MB
=> => sha256:4ece0626219de44070331daf1eff6932a03a31333a6f7f2d7b8b592a2e80d5b0 14.97MB / 14.97MB
=> => sha256:ccb4ba5bb726748e965e7730afc6ab92eb14745c7bcad9861d77d81b2372cfe 32.46MB / 32.46MB
=> => sha256:b41a241da8624e65544dd83bcc642152f10a751082d1ea1a912e238b8259533 1.79kB / 1.79kB
=> => sha256:bd2c62d9548601f6df118ad7dd984e15a0aa258cc56b4b77945fa07a6978e7 155B / 155B
=> => extracting sha256:b5a0d5c14ba9ece1eecd5137c468d9a123372b0af2ed2c8c4446137730c90e5b
=> => extracting sha256:4ece0626219de44070331daf1eff6932a03a31333a6f7f2d7b8b592a2e80d5b0
=> => extracting sha256:ccb4ba5bb726748e965e7730afc6ab92eb14745c7bcad9861d77d81b2372cfe
=> => extracting sha256:bd2c62d9548601f6df118ad7dd984e15a0aa258cc56b4b77945fa07a6978e7
```

\$ docker build -t seed-app:latest .

```
root@Linux:/home/beber/eslea-ressources/seed-data# docker build -t application-seed:latest .
[+] Building 8.9s (10/10) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 346B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/python:3.9-slim
=> [1/5] FROM docker.io/library/python:3.9-slim@sha256:96be08c44307e781fd9ce8e05b49c969b4cb902ec23594f904739c58da3a09ed
=> => resolve docker.io/library/python:3.9-slim@sha256:96be08c44307e781fd9ce8e05b49c969b4cb902ec23594f904739c58da3a09ed
=> => sha256:96be08c44307e781fd9ce8e05b49c969b4cb902ec23594f904739c58da3a09ed 1.86kB / 1.86kB
=> => sha256:79d902e3c05bb2b670c5bdf4942e7e6383b927e2d9d91349eca306aa33ae41050 1.37kB / 1.37kB
=> => sha256:4fd8d6bf114c0509e38026f76e0f5d82df784a88b099349ba2341269aaf0aa2 6.92kB / 6.92kB
=> => sha256:171d5fbeb17dd6d2ded7b4d7e09a1d4d9ec25b93c7e13d7cce55aafa185ede27 11.89MB / 11.89MB
=> => sha256:4572660747e0f9f82652c13b5d63a04a4ef2483d337a7a69b5c7e1e0925ff6a3 245B / 245B
=> => sha256:9ba7876fd8272e267f8a70670150c8f701d282039597d894ce21830039ddfbb2f 3.13MB / 3.13MB
=> => extracting sha256:171d5fbeb17dd6d2ded7b4d7e09a1d4d9ec25b93c7e13d7cce55aafa185ede27
=> => extracting sha256:4572660747e0f9f82652c13b5d63a04a4ef2483d337a7a69b5c7e1e0925ff6a3
=> => extracting sha256:9ba7876fd8272e267f8a70670150c8f701d282039597d894ce21830039ddfbb2f
=> [internal] load build context
=> => transferring context: 1.09kB
=> [2/5] RUN apt-get update && apt-get install -y --no-install-recommends apache2-utils && rm -rf /var/lib/apt/lists/*
=> [3/5] WORKDIR /seed
=> [4/5] COPY .
=> [5/5] RUN python make_data.py
=> => exporting to image
=> => exporting layers
=> => writing image sha256:ce8a0f75b4cf8e8f4f5c98fa2b088513922b092063cadb958d7d87d205e2aee
=> => naming to docker.io/library/application-seed:latest
root@Linux:/home/beber/eslea-ressources/seed-data#
```

Ensuite le lancement des images :

\$ docker run -d -p 5001:80 --name vote-container vote-app:latest

\$ docker run -d -p 5002:80 --name result-container result-app:latest

\$ docker run -d --name worker-container worker-app:latest

\$ docker run -d --name seed-container seed-app:latest

Et enfin verification de l'ensemble :

\$ docker ps

```
root@Linux:/home/beber/eslea-ressources/seed-data# docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
1091d84d4156	application-worker:latest	"dotnet Worker.dll"	About a minute ago	Up About a minute		worker-container
f130be056bff	application-result:latest	"/usr/bin/tini -- no..."	2 minutes ago	Up 2 minutes	0.0.0.0:5002->80/tcp, :::5002->80/tcp	result-container
108e5c71dbd0	application-vote:latest	"gunicorn app:app -b..."	4 minutes ago	Up 4 minutes	0.0.0.0:5001->80/tcp, :::5001->80/tcp	vote-container

Création des fichier yml

Création de docker-compose.build.yml :

```

1  version: '3.8'
2  services:
3    worker:
4      build:
5        context: ./worker
6        dockerfile: Dockerfile
7    vote:
8      build:
9        context: ./vote
10     dockerfile: Dockerfile
11   seed-data:
12     build:
13       context: ./seed-data
14       dockerfile: Dockerfile
15   result:
16     build:
17       context: ./result
18       dockerfile: Dockerfile
19   db:
20     image: postgres:15-alpine
21     environment:
22       POSTGRES_USER: "postgres"
23       POSTGRES_PASSWORD: "postgres"
24     volumes:
25       - db-data:/var/lib/postgresql/data
26       - "./healthchecks:/healthchecks"
27     healthcheck:
28       test: /healthchecks/postgres.sh
29       interval: "5s"
30     networks:
31       - back-tier
32       - cats-or-dogs-network
33   redis:
34     image: redis
35   networks:
36     back-tier:
37     cats-or-dogs-network:
38   volumes:
39     db-data:

```

Création de docker-compose.yml :

```
1 services:
2   # Service for voting
3   vote:
4     build:
5       context: ../vote
6       target: dev
7     depends_on:
8       redis:
9         condition: service_healthy
10    healthcheck:
11      test: ["CMD", "curl", "-f", "http://localhost"]
12      interval: 15s
13      timeout: 5s
14      retries: 3
15      start_period: 10s
16    volumes:
17      - ../vote:/usr/local/app
18    ports:
19      - "5001:80"
20    networks:
21      - front-tier
22      - back-tier
23
24   # Service for result handling
25   result:
26     build: ../result
27     # use nodemon rather than node for local dev
28     entrypoint: nodemon --inspect=0.0.0.0 server.js
29     depends_on:
30       db:
31         condition: service_healthy
32     volumes:
33       - ../result:/usr/local/app
34     ports:
35       - "5002:80"
36       - "127.0.0.1:9229:9229"
37     networks:
38       - front-tier
39       - back-tier
40
41   # Worker service
42   worker:
43     build:
44       context: ../worker
45     depends_on:
46       redis:
47         condition: service_healthy
48       db:
49         condition: service_healthy
50     networks:
51       - back-tier
52
53   # Redis service
54   redis:
55     image: redis:alpine
56     volumes:
57       - ../healthchecks:/healthchecks
58     healthcheck:
59       test: /healthchecks/redis.sh
```

```

60     interval: "5s"
61     networks:
62     - back-tier
63
64     # Postgres database service
65     db:
66         image: postgres:15-alpine
67         environment:
68             POSTGRES_USER: "postgres"
69             POSTGRES_PASSWORD: "postgres"
70         volumes:
71             - "db-data:/var/lib/postgresql/data"
72             - "../healthchecks:/healthchecks"
73         healthcheck:
74             test: /healthchecks/postgres.sh
75             interval: "5s"
76         networks:
77         - back-tier
78
79     # Service to seed the database with votes
80     # Will run only with the "seed" profile
81     # docker compose --profile seed up -d
82     seed:
83         build: ./seed-data
84         profiles: ["seed"]
85         depends_on:
86             vote:
87                 condition: service_healthy
88         networks:
89         - front-tier
90         restart: "no"
91
92     volumes:
93     db-data:
94
95     networks:
96     front-tier:
97     back-tier:
98     cats-or-dogs-network:
99     driver: bridge

```

Construction et démarrage de Docker

\$ docker-compose -f docker-compose.build.yml build

```

root@Linux:/home/beber/esiea-ressources# docker-compose -f docker-compose.build.yml build
db uses an image, skipping
redis uses an image, skipping
Building worker
[+] Building 0.4s (16/16) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 1.04kB
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for mcr.microsoft.com/dotnet/runtime:7.0
=> [internal] load metadata for mcr.microsoft.com/dotnet/sdk:7.0
=> [build 1/7] FROM mcr.microsoft.com/dotnet/sdk:7.0@sha256:4be8ff7cb847f9e7ea1ac194920b0a6d41956af89f934b61a2ce64dc8563471c
=> [stage-1 1/3] FROM mcr.microsoft.com/dotnet/runtime:7.0@sha256:b41a241da8624e65544dd83cbcc642152f10a751082d1ea1a912e238b8259533
=> [internal] load build context
=> => transferring context: 95B
=> CACHED [stage-1 2/3] WORKDIR /app
=> CACHED [build 2/7] RUN echo "I am running on linux/amd64, building for linux/amd64"
=> CACHED [build 3/7] WORKDIR /source
=> CACHED [build 4/7] COPY *.csproj .
=> CACHED [build 5/7] RUN dotnet restore -a amd64
=> CACHED [build 6/7] COPY . .
=> CACHED [build 7/7] RUN dotnet publish -c release -o /app -a amd64 --self-contained false --no-restore
=> CACHED [stage-1 3/3] COPY --from=build /app .
=> exporting to image
=> => exporting layers
=> => writing image sha256:9cb8f2a74b60d85e23760846835deafb5953b6f9857d8ef86a7199f427c26fba
=> => naming to docker.io/library/esiea-ressources_worker

```

\$ docker-compose -f docker-compose.build.yml up


```

root@Linux:/home/beber/esiea-ressources# docker-compose -f docker-compose.build.yml up
Creating network "esiea-ressources_default" with the default driver
Creating network "esiea-ressources_back-tier" with the default driver
Creating network "esiea-ressources_cats-or-dogs-network" with the default driver
Creating volume "esiea-ressources_db-data" with default driver
Pulling db (postgres:15-alpine)...
15-alpine: Pulling from library/postgres
661ff4d9561e: Pull complete
51985e4f4706: Pull complete
047846c02df4: Pull complete
02927b05f2c8: Extracting [=====] 84.67MB/91.4MB
a223aae49cc3: Download complete
218b46dad3a9: Download complete
d50f691fa37b: Download complete
eca62a4d01cd: Download complete
e8186d2cb497: Download complete

```

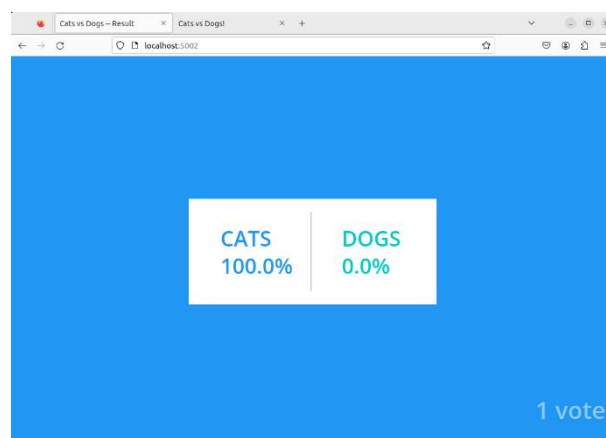
Lancement

\$ docker-compose up

```

root@Linux:/home/beber/esiea-ressources# docker-compose up
WARNING: Some networks were defined but are not used by any service: cats-or-dogs-
WARNING: Found orphan containers (esiea-ressources_seed-data_1) for this project.
ean it up.
Recreating esiea-ressources_redis_1 ... done
Recreating esiea-ressources_db_1 ... done
Recreating 6ef8092a2aab_esiea-ressources_vote_1 ... done
Recreating esiea-ressources_worker_1 ... done
Recreating d2bfd442c3c2_esiea-ressources_result_1 ... done
Attaching to esiea-ressources_redis_1, esiea-ressources_db_1, esiea-ressources_vot
db_1 |
db_1 | PostgreSQL Database directory appears to contain a database; Skipping
db_1 |
db_1 | 2024-01-09 12:33:07.120 UTC [1] LOG: starting PostgreSQL 15.5 on x86_
db_1 | 2024-01-09 12:33:07.120 UTC [1] LOG: listening on IPv4 address "0.0.0
db_1 | 2024-01-09 12:33:07.120 UTC [1] LOG: listening on IPv6 address ":::",
db_1 | 2024-01-09 12:33:07.139 UTC [1] LOG: listening on Unix socket "/var/r
db_1 | 2024-01-09 12:33:07.143 UTC [23] LOG: database system was shut down a
db_1 | 2024-01-09 12:33:07.148 UTC [1] LOG: database system is ready to acce
redis_1 | 1:C 09 Jan 2024 12:33:06.995 # WARNING Memory overcommit must be enabl
ilures without low memory condition, see https://github.com/jemalloc/jemalloc/issu
m.overcommit_memory=1' for this to take effect.
redis_1 | 1:C 09 Jan 2024 12:33:06.995 * oO00oO00oO00o Redis is starting oO00oO0
redis_1 | 1:C 09 Jan 2024 12:33:06.995 * Redis running 7.0.3 bits 64 commit 00

```



```
worker_1 | Processing vote for 'a' by '28602f606d3d0dd'  
vote_1 | 127.0.0.1 - - [09/Jan/2024:12:34:44 +0000] "GET / HTTP/1.1" 200 1285 "-" "curl/7.88.1"  
vote_1 | 127.0.0.1 - - [09/Jan/2024:12:34:59 +0000] "GET / HTTP/1.1" 200 1285 "-" "curl/7.88.1"
```

Kubernetes

Installation et lancement de Kubernetes :

```
curl -LO https://dl.k8s.io/release/\$\(curl -L -s https://dl.k8s.io/release/stable.txt\)/bin/linux/amd64/kubectl*
```

```
curl -LO https://dl.k8s.io/release/\$\(curl -L -s https://dl.k8s.io/release/stable.txt\)/bin/linux/amd64/kubectl.sha256
```

```
echo "$(cat kubectl.sha256) kubectl" | sha256sum -check
```

```
sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
```

Installation et lancement de minikube :

```
$ curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linuxamd64
```

```
$ sudo install minikube-linux-amd64 /usr/local/bin/minikube
```

```
$ minikube start
```

```
$ kubectl create -f k8s-specifications/
```

```
beber@Linux:~$ minikube start  
😊 minikube v1.32.0 on Ubuntu 22.04 (vbox/amd64)  
✨ Automatically selected the docker driver  
  
💡 The requested memory allocation of 1959MiB does not leave room for system  
Suggestion: Start minikube with less memory allocated: 'minikube start -m 2048m'  
  
🔧 Using Docker driver with root privileges  
👍 Starting control plane node minikube in cluster minikube  
📡 Pulling base image ...  
📦 Downloading Kubernetes v1.28.3 preload ...
```

Conclusion

Le défi de développer et de déployer une application en utilisant la virtualisation et la conteneurisation s'est avéré être une véritable épreuve. Initialement, nous avons rencontré des difficultés à relier les concepts enseignés en cours à leur application pratique. Ensuite, la configuration des machines virtuelles s'est avérée être un défi majeur, nécessitant une persévérance soutenue sur une période de plus de huit heures.

Malheureusement, malgré nos efforts considérables, le résultat final s'est soldé par un échec en raison de la non-acceptation de la virtualisation sur les ordinateurs de chaque membre du groupe. Cette expérience a mis en lumière la complexité intrinsèque de cette tâche et son aspect chronophage, nous incitant à reconsidérer notre approche pour assurer une meilleure efficacité à l'avenir. Cependant, malgré ces obstacles, nous avons persisté, car l'approche présentée dans le sujet détaillait clairement nos objectifs.

En explorant d'autres alternatives, nous avons finalement donné naissance à l'application HumansBestFriend. Cette situation a été une leçon importante, soulignant à quel point la virtualisation et la conteneurisation peuvent être exigeantes, mais également combien elles sont essentielles et stimulantes pour notre apprentissage.

Annexes :

Liens :

<https://github.com/Beber328/project-human-best-friend>