HUMAN'S BEST FRIENDS

Cats or Dogs ?

FISA 4A

MOSELLE Bermond VIRASSINGARAYAR Cédric NTAKIYIRUTA Jean-Luc

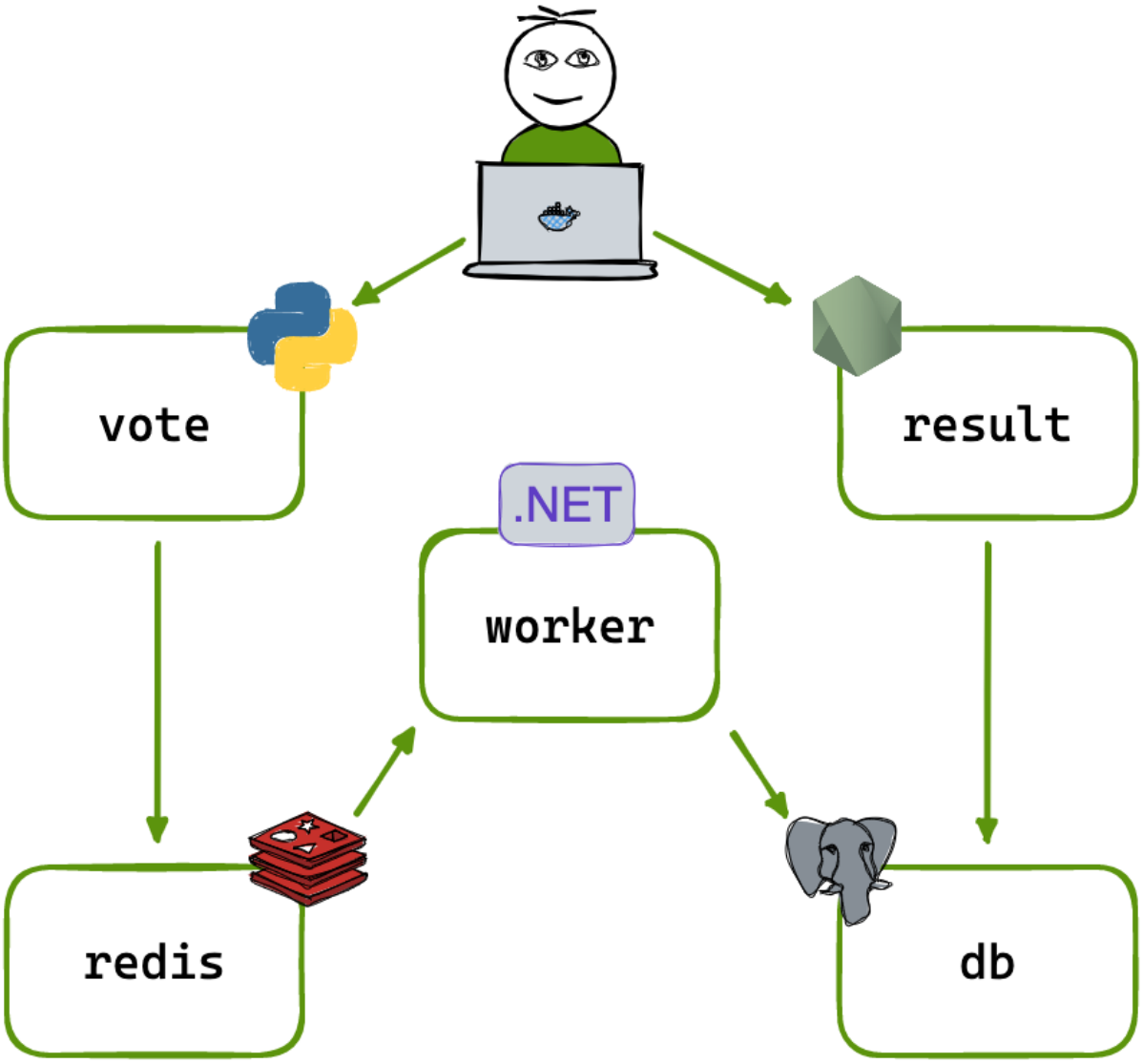
Sommaire :

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

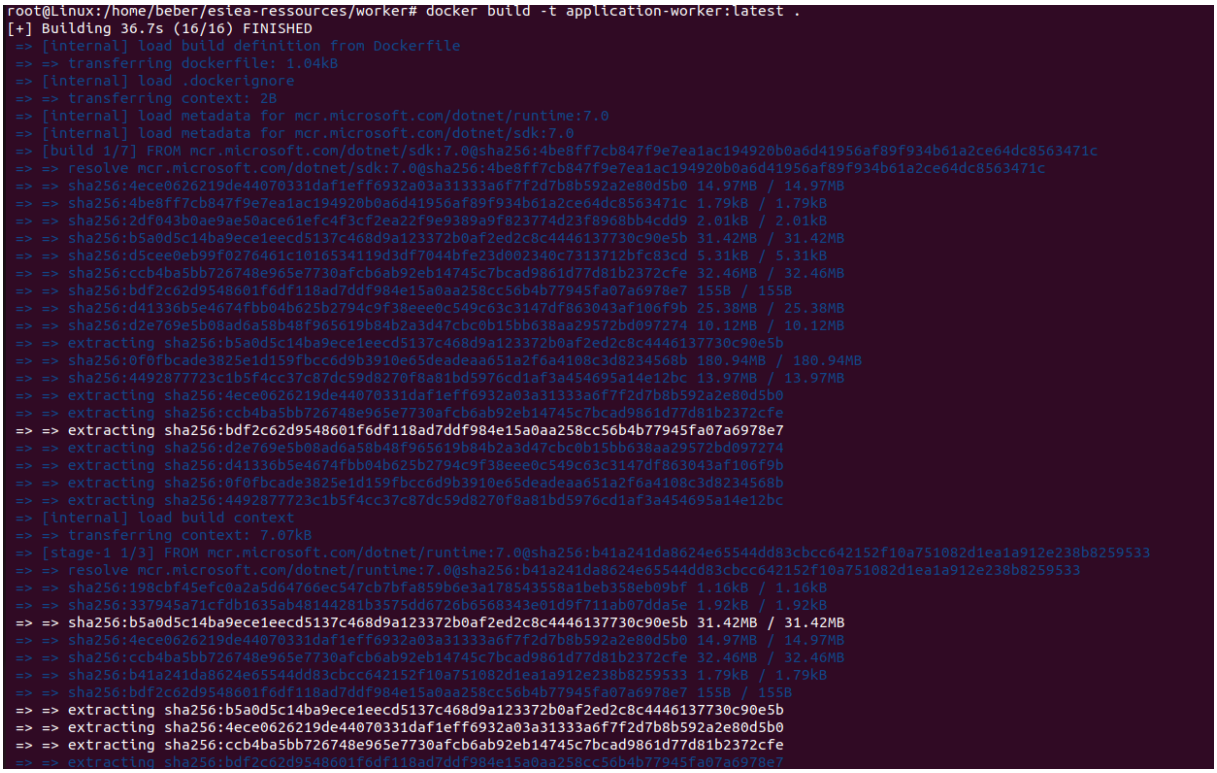
Après avoir copié le GIT, on se place dans le repertoire afin de créer et lancer les images : $ docker build -t vote-app:latest .

```
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@sha256:8f64a67710f3d981cf3008d6f9f1dbe61accd7927f165f4e37ea3f8b883ccc3f
 => [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
```

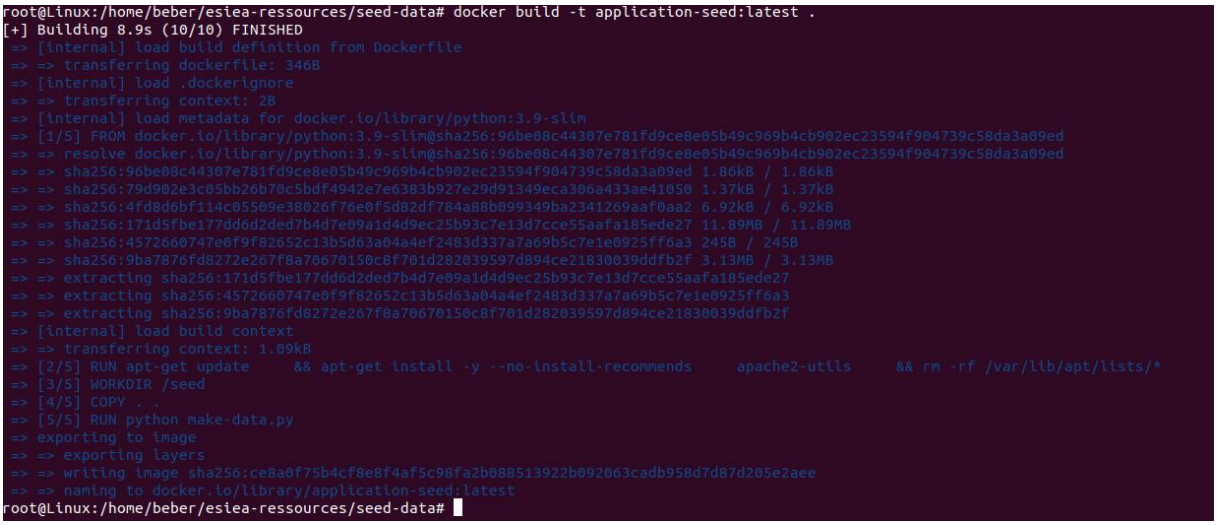
$ docker build -t result-app:latest .

```
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
 => => sha256:8e04602828ddd8c394c701f265c048f2a7cf9cbf112635ba26cec2d06936f17b 1.37kB / 1.37kB
 => => sha256:d3cce7487840f2783532a37fca9e79c4d55fea3d5ce3caf7c145596fcc457f8e 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 .



$ docker build -t seed-app:latest .



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/esiea-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 :

```yaml
version: '3.8'
services:
  worker:
    build:
      context: ./worker
      dockerfile: Dockerfile
  vote:
    build:
      context: ./vote
      dockerfile: Dockerfile
  seed-data:
    build:
      context: ./seed-data
      dockerfile: Dockerfile
  result:
    build:
      context: ./result
      dockerfile: Dockerfile
  db:
    image: postgres:15-alpine
    environment:
      POSTGRES_USER: "postgres"
      POSTGRES_PASSWORD: "postgres"
    volumes:
      - db-data:/var/lib/postgresql/data
      - "./healthchecks:/healthchecks"
    healthcheck:
      test: /healthchecks/postgres.sh
      interval: "5s"
    networks:
      - back-tier
      - cats-or-dogs-network
  redis:
    image: redis
networks:
  back-tier:
  cats-or-dogs-network:
volumes:
  db-data:
```

Création de docker-compose.yml :

```yaml
services:
  # Service for voting
  vote:
    build:
      context: ./vote
      target: dev
    depends_on:
      redis:
        condition: service_healthy
    healthcheck:
      test: ["CMD", "curl", "-f", "http://localhost"]
      interval: 15s
      timeout: 5s
      retries: 3
      start_period: 10s
    volumes:
     - ./vote:/usr/local/app
    ports:
      - "5001:80"
    networks:
      - front-tier
      - back-tier

  # Service for result handling
  result:
    build: ./result
    # use nodemon rather than node for local dev
    entrypoint: nodemon --inspect=0.0.0.0 server.js
    depends_on:
      db:
        condition: service_healthy
    volumes:
      - ./result:/usr/local/app
    ports:
      - "5002:80"
      - "127.0.0.1:9229:9229"
    networks:
      - front-tier
      - back-tier

  # Worker service
  worker:
    build:
      context: ./worker
    depends_on:
      redis:
        condition: service_healthy
      db:
        condition: service_healthy
    networks:
      - back-tier

  # Redis service
  redis:
    image: redis:alpine
    volumes:
      - "./healthchecks:/healthchecks"
    healthcheck:
      test: /healthchecks/redis.sh
```

```yaml
      interval: "5s"
    networks:
      - back-tier

  # Postgres database service
  db:
    image: postgres:15-alpine
    environment:
      POSTGRES_USER: "postgres"
      POSTGRES_PASSWORD: "postgres"
    volumes:
      - "db-data:/var/lib/postgresql/data"
      - "./healthchecks:/healthchecks"
    healthcheck:
      test: /healthchecks/postgres.sh
      interval: "5s"
    networks:
      - back-tier

  # Service to seed the database with votes
  # Will run only with the "seed" profile
  # docker compose --profile seed up -d
  seed:
    build: ./seed-data
    profiles: ["seed"]
    depends_on:
      vote:
        condition: service_healthy
    networks:
      - front-tier
    restart: "no"

volumes:
  db-data:

networks:
  front-tier:
  back-tier:
  cats-or-dogs-network:
    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:b41a241da8624e65544dd83cbcc642152f10a751082d1ea1a912e238b825953
 => [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 sha256:9cb8f2a74b60d85e2376084683Sdeafb5953b6f9857d8ef86a7199f427c26fba
 => => naming to docker.io/library/esiea-ressources_worker
```

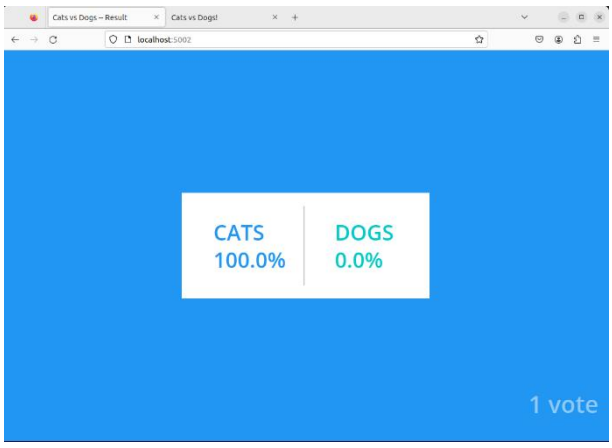
$ docker-compose -f docker-compose.build.yml up

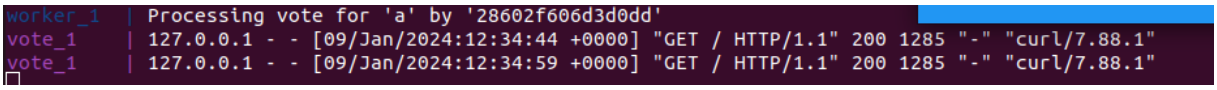
```
root@Linux:/home/beber/esiea-ressources# docker-compose -f docker-compose.build.yml up
Creating network "esiea-ressources_default" with the default driver
Creating network "esiea-ressources_back-tier" with the default driver
Creating network "esiea-ressources_cats-or-dogs-network" with the default 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 * oO0OoO0OoO0Oo Redis is starting oO0OoO0
```





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

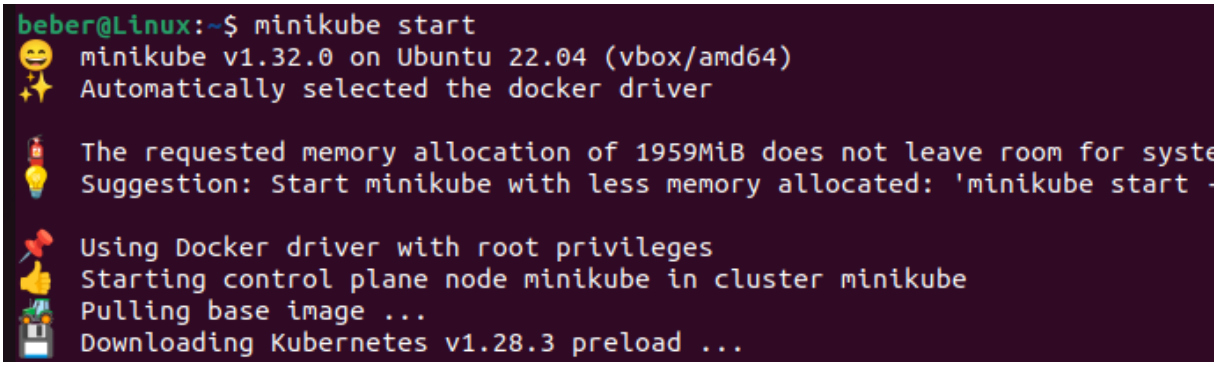
Installation et lancement de minikube :

$ 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/



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