

VIRTUALISATION ET CONTENEUR

Projet HumansBestFriend - Application Distribuée avec Docker et Kubernetes

Ce projet a été réalisé par Anaïs DERAMCHIA et Romain FOURNET

INTRODUCTION

Le projet HumansBestFriend - Application Distribuée avec Docker et Kubernetes a pour objectif la création d'une application distribuée simple, appelée HumansBestFriend, qui permet aux utilisateurs de voter pour leur animal de compagnie préféré, qu'il s'agisse de chats ou de chiens. L'application sera déployée à l'aide de Docker et Kubernetes pour illustrer la gestion de conteneurs et d'orchestration dans un environnement distribué. Le projet sera réalisé à l'intérieur d'une machine virtuelle exécutant Docker et Docker Compose.

DÉROULEMENT

Dans le cadre de notre projet, notre première étape a été de mettre en place notre environnement ESXi et de lancer notre machine virtuelle. Une fois cette configuration réalisée, nous avons établi une connexion SSH entre le terminal de notre PC et notre machine virtuelle en utilisant la commande suivante :

SSH [romain@172.16.223.129](ssh:romain@172.16.223.129)

Afin de pouvoir exploiter les fichiers disponibles sur GitHub, nous avons procédé à l'installation de Docker en suivant les instructions détaillées sur le lien suivant : <https://docs.docker.com/engine/install/ubuntu/>

Cette première étape cruciale nous a permis de préparer notre environnement pour la suite du projet.

Suite à l'installation de Docker, nous avons également installé le plugin Compose en suivant les directives spécifiées dans la documentation : <https://docs.docker.com/compose/install/linux/#install-using-the-repository>

Cette étape a été essentielle pour la gestion des services et la configuration de l'ensemble de notre application.

De même, nous avons créé un répertoire et ensuite, nous avons procédé au clonage du projet à partir du référentiel GitHub du professeur. Le clonage a été réalisé en utilisant la commande git clone suivie de l'URL du référentiel.

Le cœur de notre travail a véritablement commencé avec la création de deux fichiers clés : "docker-compose.images.yml" et "docker-compose.yml".

Le premier fichier, "docker-compose.images.yml", était responsable de la construction des images de l'application à partir du contenu du fichier Docker fourni.

Le second fichier, "docker-compose.yml", était dédié au déploiement de l'application et à la gestion de tous les conteneurs nécessaires.

Lors de la création de ces fichiers, nous avons minutieusement suivi la procédure, en spécifiant notamment que le service de vote serait accessible sur le port localhost 5002 et les résultats du vote sur le port localhost 5001.

Une fois le code rédigé, nous avons exécuté la commande suivante pour lancer l'ensemble du système :

`sudo docker compose up` : Cette commande est utilisée pour lancer et démarrer les services définis dans un fichier de configuration Docker Compose

Enfin, pour visualiser les résultats de notre travail, nous nous sommes connectés à Internet en ouvrant différents onglets de navigateur et en accédant aux adresses suivantes :

172.16.223.129:5002 (service de vote)

172.16.223.129:5001 (résultats du vote)

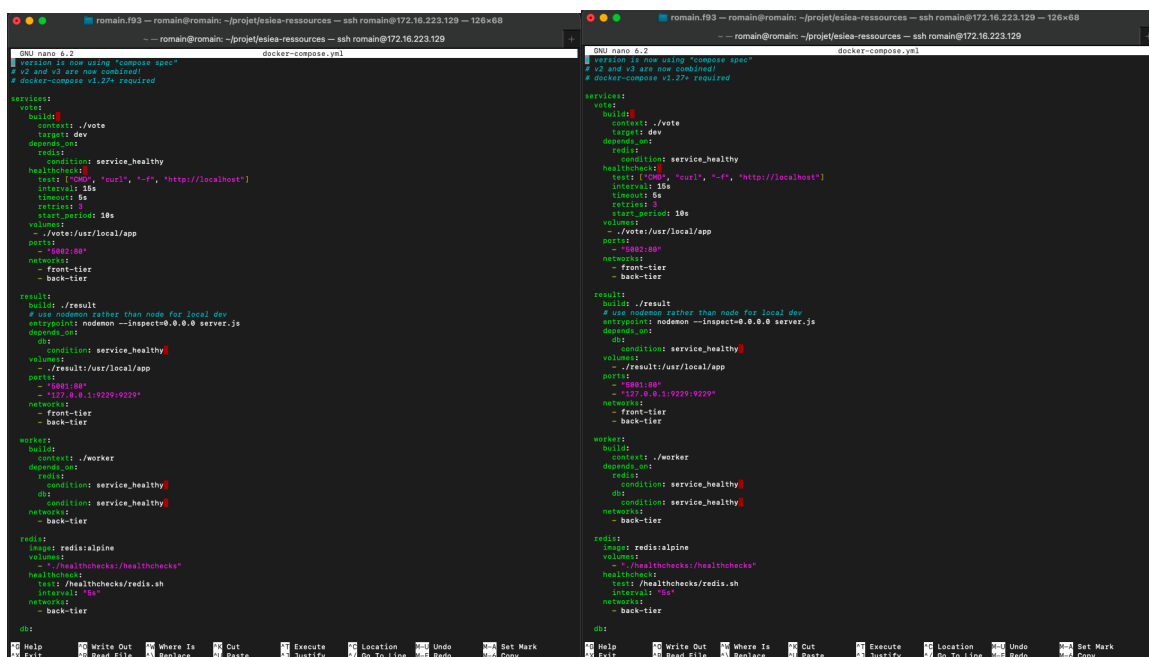
Cela nous a permis de vérifier le bon fonctionnement de notre application distribuée et de constater concrètement les résultats de notre travail sur ce projet.

CONCLUSION

En conclusion, le projet HumansBestFriend a constitué une expérience enrichissante dans la conception et le déploiement d'une application distribuée. En suivant rigoureusement les étapes, depuis la configuration initiale de l'environnement ESXi jusqu'à la mise en œuvre des fichiers Docker Compose, notre équipe a acquis une compréhension approfondie de la gestion de conteneurs à l'aide de Docker et de l'orchestration avec Kubernetes.

En ce qui concerne les difficultés rencontrées, elles furent minimales (type problème de droit ou d'installation) mais dans l'ensemble tout était très bien indiqué et détaillé dans les consignes.

ANNEXE



```
version: '3.8'
services:
  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:
      - "5002:80"
    networks:
      - front-tier
      - back-tier

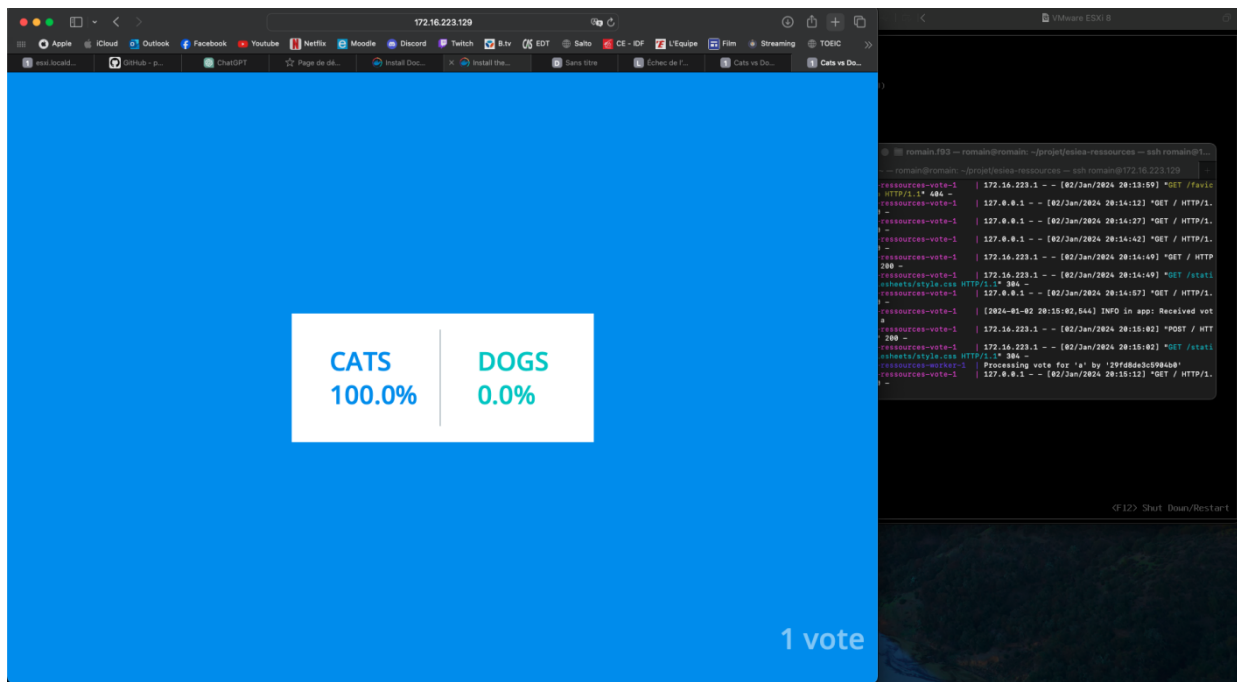
  result:
    build: ./result
    # use nodemon rather than node for local dev
    entrypoint: nodemon --inspect=0.0.0 server.js
    depends_on:
      db:
        condition: service_healthy
    volumes:
      - ./result:/usr/local/app
    ports:
      - "5001:80"
      - "127.0.0.1:9229:9229"
    networks:
      - front-tier
      - back-tier

  worker:
    build:
      context: ./worker
    depends_on:
      redis:
        condition: service_healthy
    healthcheck:
      test: ["CMD", "redis-cli ping"]
      interval: 10s
      timeout: 5s
      retries: 3
      start_period: 10s
    volumes:
      - ./worker:/usr/local/app
    ports:
      - "5003:80"
    networks:
      - front-tier
      - back-tier

  redis:
    image: redis:alpine
    healthcheck:
      test: ["CMD", "redis-cli ping"]
      interval: 10s
      timeout: 5s
      retries: 3
      start_period: 10s
    volumes:
      - ./redis:/usr/local/app
    ports:
      - "6379:6379"
    networks:
      - front-tier
      - back-tier

networks:
  front-tier:
  back-tier:
  db:
    image: postgres:13
    volumes:
      - ./db:/var/lib/postgresql/data
    ports:
      - "5432:5432"
    networks:
      - front-tier
      - back-tier
```

Création des fichiers "docker-compose.images.yml" et "docker-compose.yml"



Visualisation des résultats de notre travail