TP-Docker

Table des matières

Résumé du Projet

Contexte :

Une application distribuée simple, "HumansBestFriend", utilisant plusieurs conteneurs Docker.

Le projet doit être réalisé dans une machine virtuelle équipée de Docker et Docker Compose.

Technologies Utilisées :

Python, Node.js, .NET

Redis pour la messagerie

Postgres pour le stockage

Tâches Principales

Fichier docker-compose.build.yml :

Créer ce fichier pour construire les images de l'application à partir des Dockerfiles fournis.

Service Worker :

Dépend de Redis et de la base de données (db).

Utilise l'image Docker de .NET pour la construction et l'exécution.

Service Vote :

Associer un volume au dossier /usr/local/app dans le conteneur.

Le service écoute sur le port 80, mais peut être exposé à l'extérieur (par exemple, sur le port 5002).

Utilise l'image Docker de Python.

Service Seed-Data :

Doit être dans le réseau front-tier.

Utilise l'image Docker de Python avec l'outil apache bench (ab).

Service Result :

Le conteneur interne utilise le port 80, exposé à l'extérieur sur le port 5001.

Utilise l'image Docker de Node.js.

Base de Données Postgres :

Utiliser l'image postgres:lastest.

Service Redis

Architecture

Application web frontale en Python pour voter entre deux options.

Redis pour la collecte des votes.

Worker .NET pour le traitement des votes et leur stockage dans une base de données Postgres.

Application web Node.js pour afficher les résultats des votes en temps réel.

Consignes Supplémentaires

L'application accepte un seul vote par navigateur client.

Le projet doit être documenté et soumis via un dépôt public GitHub, incluant un fichier SUBMISSION.md détaillé.

La mise en réseau des conteneurs doit être démontrée (par exemple, par un simple ping).

Date Limite

La soumission doit être complétée avant 18h00 aujourd'hui.

Initialisation de l'environnement

Connexion a la machine virtuelle :

```
C:\Users\HichemStinson13\Desktop\docker>ssh stinson@192.168.3.38
The authenticity of host '192.168.3.38 (192.168.3.38)' can't be established.
ECDSA key fingerprint is SHA256:Kjx3Rtn7YlRKmpIrMOkFZ6RUwExwATjysu5w3JON5J0.
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added '192.168.3.38' (ECDSA) to the list of known hosts.
stinson@192.168.3.38's password:
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-91-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of jeu. 21 déc. 2023 12:10:59 UTC

  System load:                    0.57421875
  Usage of /:                     43.3% of 9.75GB
  Memory usage:                   11%
  Swap usage:                     0%
  Processes:                      249
  Users logged in:                0
  IPv4 address for br-a03a0150ece7: 172.18.0.1
  IPv4 address for docker0:       172.17.0.1
  IPv4 address for ens33:         192.168.3.38

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.

   https://ubuntu.com/engage/secure-kubernetes-at-the-edge

La maintenance de sécurité étendue pour Applications n'est pas activée.

48 mises à jour peuvent être appliquées immédiatement.
4 de ces mises à jour sont des mises à jour de sécurité.
Pour afficher ces mises à jour supplémentaires, exécuter : apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


Last login: Thu Dec 21 12:10:59 2023
```

Cloner le git :

Cloner le git donné pour le tp

```
stinson@stinson:~$ git clone https://github.com/pascalito007/ynov-resources/
Cloning into 'ynov-resources'...
remote: Enumerating objects: 402, done.
remote: Counting objects: 100% (30/30), done.
remote: Compressing objects: 100% (22/22), done.
remote: Total 402 (delta 7), reused 22 (delta 4), pack-reused 372
Receiving objects: 100% (402/402), 11.55 MiB | 548.00 KiB/s, done.
Resolving deltas: 100% (59/59), done.
```

Accéder au dossier humans-best-friend :

L'accès au dossier humans-best-friend pour travailler dedans

```
stinson@stinson:~/ynov-resources$ cd 2023/m2/dataeng/humans-best-friend/
stinson@stinson:~/ynov-resources/2023/m2/dataeng/humans-best-friend$
```

Création des fichiers :

Pour ce TP nous allons avoir besoins de créer plusieurs fichiers a savoir :

- docker-compose.build.yml
- compose.yml

Créer le docker-compose.build.yml :

Nous allons d'abord commencer par la creation du « docker-compose.build.yml » pour cela nous allons suivre les étapes suivantes :

```
stinson@stinson:~/ynov-resources/2023/m2/dataeng/humans-best-friend$ nano docker-compose.build.yml
```

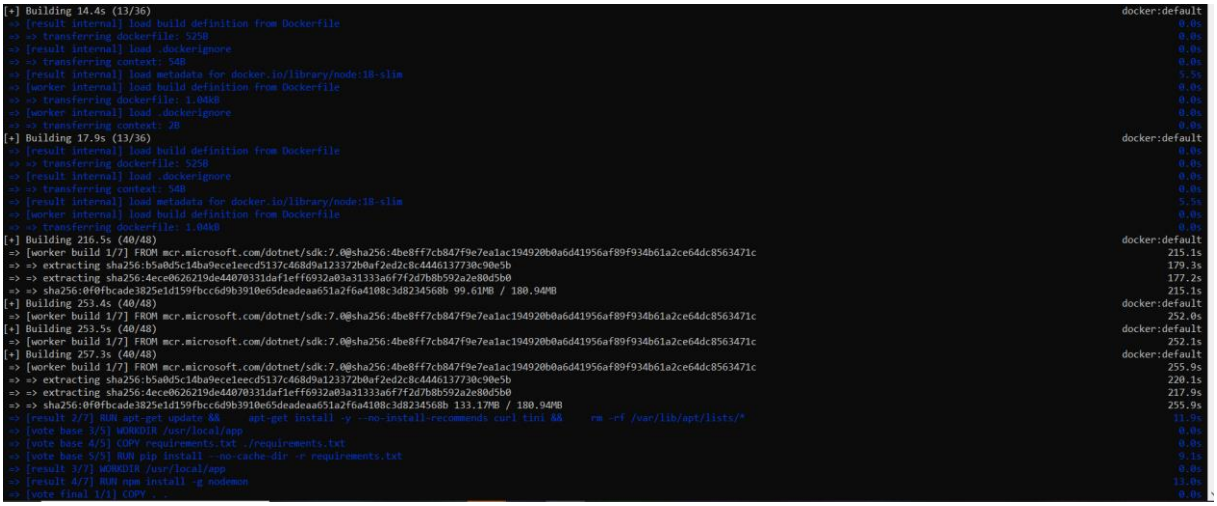
```yaml
version: '3'

services:
  worker:
    build:
      context: ./worker
      dockerfile: Dockerfile
    depends_on:
      redis:
        condition: service_healthy
      db:
        condition: service_healthy
    networks:
      - humansbestfriend-network
  vote:
    build:
      context: ./vote
      dockerfile: Dockerfile
    volumes:
      - ./vote:/usr/local/app
    networks:
      - humansbestfriend-network
  seed-data:
    build:
      context: ./seed-data
      dockerfile: Dockerfile
    depends_on:
      - vote
    restart: "no"
    networks:
      - humansbestfriend-network

  result:
    build:
      context: ./result
      dockerfile: Dockerfile
    depends_on:
      db:
        condition: service_healthy
    volumes:
      - ./result:/usr/local/app
    networks:
      - humansbestfriend-network

  redis:
    image: "redis:latest"
```

```yaml
  redis:
    image: "redis:latest"
    networks:
      - humansbestfriend-network

  db:
    image: "postgres:latest"
    networks:
      - humansbestfriend-network

networks:
  humansbestfriend-network:
```

Exécution du build pour le fichier docker-compose.build.yml

Ensuite nous utilisons la commande suivante a fin de build le fichier « docker-compose.build.yml »

```
stinson@stinson:~/ynov-resources/2023/m2/dataeng/humans-best-friend$    docker-compose -f docker-compose.build.yml build
```



Run la container registry pour mettre les autres images dedans

Pour cette étape nous allons run le container registry dans lequel nous allons push nos images afin de les utiliser par la suite

```
docker run -d -p 5000:5000 --restart always --name registry registry:latest
```

Taguer les images avec localhost :5000/ :

Une fois le container run nous allons taguer nos image d'un préfix : « localhost :5000 » grace aux commandes suivantes :

```
stinson@stinson:~/ynov-resources/2023/m2/dataeng/humans-best-friend$ docker tag humans-best-friend-worker localhost:5000/humans-best-friend-worker
stinson@stinson:~/ynov-resources/2023/m2/dataeng/humans-best-friend$ docker tag humans-best-friend-vote localhost:5000/humans-best-friend-vote
stinson@stinson:~/ynov-resources/2023/m2/dataeng/humans-best-friend$ docker tag humans-best-friend-seed-data localhost:5000/humans-best-friend-seed-data
stinson@stinson:~/ynov-resources/2023/m2/dataeng/humans-best-friend$ docker tag humans-best-friend-seed-result localhost:5000/humans-best-friend-result
Error response from daemon: No such image: humans-best-friend-seed-result:latest
stinson@stinson:~/ynov-resources/2023/m2/dataeng/humans-best-friend$ docker tag humans-best-friend-result localhost:5000/humans-best-friend-result
```

Push les images dans le localhost :5000 :

Une fois toutes les images taguées nous allons les push dans le registry comme suit :

```
stinson@stinson:~/ynov-resources/2023/m2/dataeng/humans-best-friend$ docker push localhost:5000/humans-best-friend-vote
Using default tag: latest
The push refers to repository [localhost:5000/humans-best-friend-vote]
2d1cef1697f4: Pushed
57d3d7544d8b: Pushed
9a3b4ba06c6d: Pushed
46cb6a26b99a: Pushed
d4937a0183da: Pushed
5a0cd8defe69: Pushed
32f021973527: Pushed
52ee4febd598: Pushed
384858ccd7ef: Pushed
7292cf786aa8: Pushed
latest: digest: sha256:576da06c9c740190aefbce39ecb14ac5742714fd7cd2178d01ce87fcf03171ec size: 2414
```

```
stinson@stinson:~/ynov-resources/2023/m2/dataeng/humans-best-friend$ docker push localhost:5000/humans-best-friend-worker
Using default tag: latest
The push refers to repository [localhost:5000/humans-best-friend-worker]
42ce18ae6d18: Pushed
561d248ee836: Pushed
5ee537aacfb3: Pushed
7c73e219acdd: Pushed
d47be6633206: Pushed
e6cd39d4cea4: Pushed
latest: digest: sha256:0a746750a586d4b55e618fb4ae82ac731feff3b138547415c9071d0200807241 size: 1577
```

```
stinson@stinson:~/ynov-resources/2023/m2/dataeng/humans-best-friend$ docker push localhost:5000/humans-best-friend-result
Using default tag: latest
The push refers to repository [localhost:5000/humans-best-friend-result]
93cab2b14d78: Pushed
de3d1871b364: Pushed
987a8a626050: Pushed
5f07ac4b084c: Pushed
1550928af8de: Pushed
cf705e065b95: Pushed
59ea00d1a724: Pushed
38a1fe82f870: Pushed
fa348aca89c0: Pushed
5a02db9fb904: Pushed
7292cf786aa8: Mounted from humans-best-friend-vote
latest: digest: sha256:687138f037445ebfd5bd1668ded273bd686dd65694629e67d094ee14c7a5202a size: 2625
stinson@stinson:~/ynov-resources/2023/m2/dataeng/humans-best-friend$ docker push localhost:5000/humans-best-friend-seed-data
Using default tag: latest
The push refers to repository [localhost:5000/humans-best-friend-seed-data]
c159f03014b3: Pushed
d22ce8898da1: Pushed
3da4bacad700: Pushed
875dce0b2c5c: Pushed
4cec408bacee: Pushed
a1e3c54d75a8: Pushed
561ecc6e457f: Pushed
384858ccd7ef: Mounted from humans-best-friend-vote
7292cf786aa8: Mounted from humans-best-friend-result
latest: digest: sha256:68bebedf5160b6cf4d275ca4c1223caa5d9872122865be923aeeda15f2ae68c1 size: 2203
```

Verifier qu'elles sont bien de le registry

Une fois que c'est fait nous pouvons vérifier que nous avons bien nos image dans le catalog grace a la commande suivante :

```
stinson@stinson:~/ynov-resources/2023/m2/dataeng/humans-best-friend$ curl localhost:5000/v2/_catalog
{"repositories":["humans-best-friend-result","humans-best-friend-seed-data","humans-best-friend-vote","humans-best-friend-worker","nginx"]}
```

Créer le fichier compose.yml

Maintenant on passe a la creation du fichier « compose.yml » dans lequel nous allons utiliser les image stockées auparavant dans le localhost :5000 comme cela :

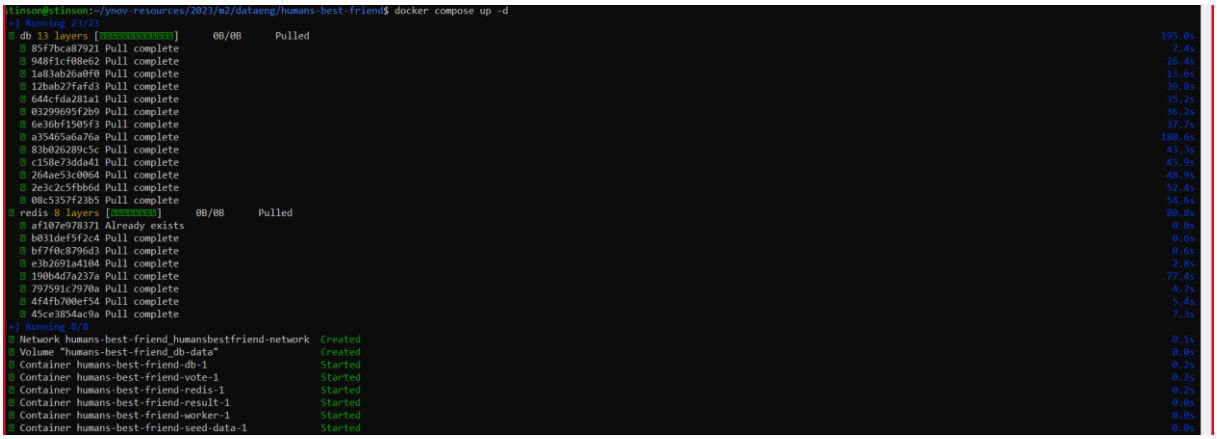
```
stinson@stinson:~/ynov-resources/2023/m2/dataeng/humans-best-friend$ nano compose.yml
```

```yaml
version: '3.8'

services:
  worker:
    image: localhost:5000/humans-best-friend_worker
    depends_on:
      - redis
      - db
    networks:
      - humansbestfriend-network

  vote:
    image: localhost:5000/humans-best-friend_vote
    volumes:
      - ./vote:/usr/local/app
    ports:
      - "5002:80"
    healthcheck:
      test: ["CMD", "curl", "-f", "http://localhost"]
      interval: 15s
      timeout: 5s
      retries: 3
      start_period: 10s
    networks:
      - humansbestfriend-network

  seed-data:
    image: localhost:5000/humans-best-friend_seed-data
    depends_on:
      - vote
    restart: "no"
    networks:
      - humansbestfriend-network

  result:
    image: localhost:5000/humans-best-friend_result
    depends_on:
      - db
    volumes:
      - ./result:/usr/local/app
    ports:
      - "5001:80"
      - "9229:9229"
    networks:
      - humansbestfriend-network
```

```yaml
  redis:
    image: redis:latest
    networks:
      - humansbestfriend-network

  db:
    image: postgres:latest
    networks:
      - humansbestfriend-network
    volumes:
      - "db-data:/var/lib/postgresql/data"
      - "./healthchecks:/healthchecks"
    healthcheck:
      test: /healthchecks/postgres.sh
      interval: "5s"

networks:
  humansbestfriend-network:

volumes:
  db-data:
```

Faire le docker compose up du compose.yml

Une fois cela fait on exécute la commande suivant pour le fichier « compose.yml »

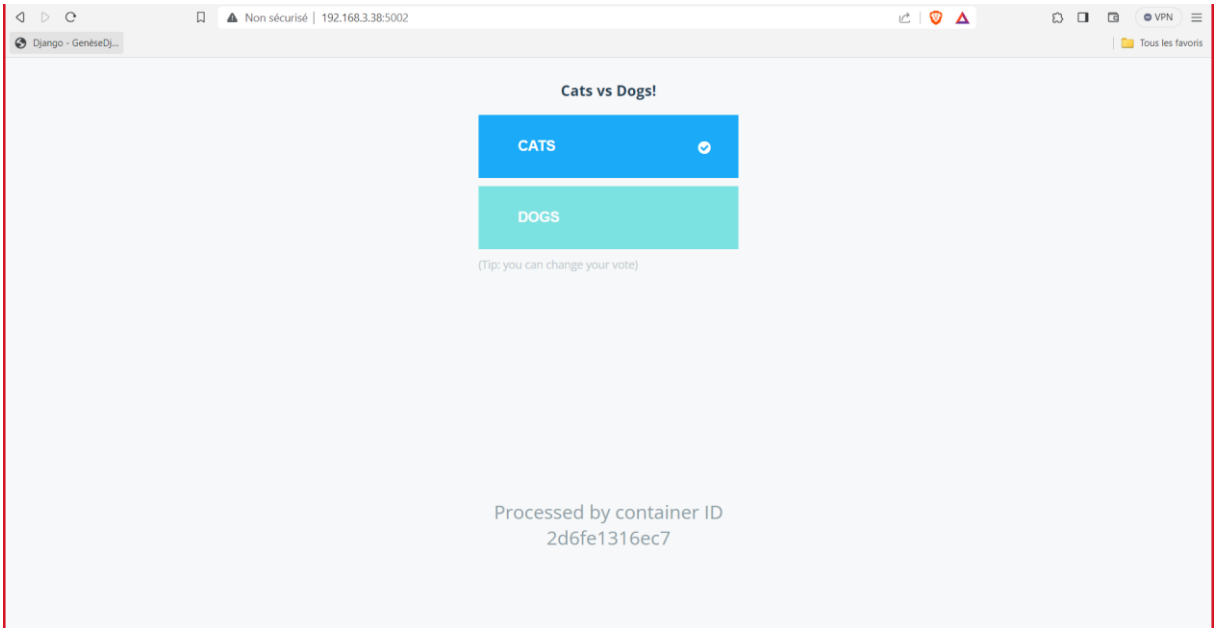


Resultats

Nous pouvons alors maintenant aller sur un navigateur afin de vérifier les résultats qu'on a obtenue

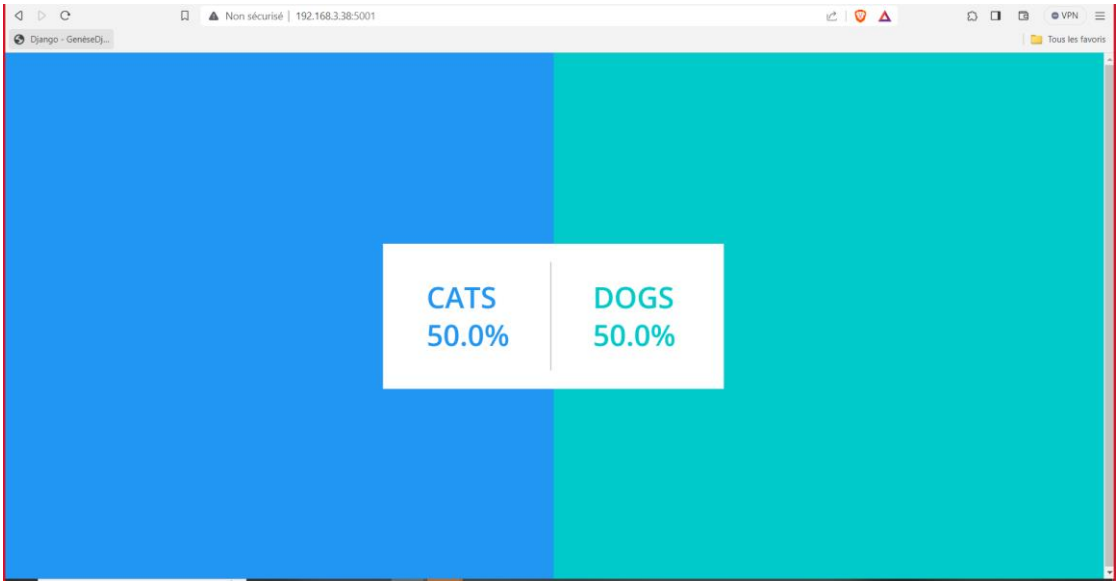
Resultat obtenue quand on lance dans le navigateur 192.168.3.38 :5002 :

On peut aller a l'adresse : « 192.168.3.38 :5002 » pour voter



Resultat obtenue quand on lance dans le navigateur 192.168.3.38 :5001

Puis on va à l'adresse : « 192.168.3.38 :5001 » pour voir le resultat des votes



Verifier le ping entre deux container :

Enfin nous avons vérifié la communication entre les containers comme suit :

Acceder au containers et installer iputils-ping dans tout les containers en accèdant au container avec la commande commande dans l'exemple suivant :

docker exec -it ebfdd9021f04 bash

puis en utiilsant les commande suivant :

apt-get update

apt-get install iputils-ping

```
stinson@stinson:~/ynov-resources/2023/m2/dataeng/humans-best-friend$ docker exec -it 0f621970f5b9 ping -c 5 ebfdd9021f04
PING ebfdd9021f04 (172.19.0.6) 56(84) bytes of data.
64 bytes from humans-best-friend-worker-1.humans-best-friend_humansbestfriend-network (172.19.0.6): icmp_seq=1 ttl=64 time=0.105 ms
64 bytes from humans-best-friend-worker-1.humans-best-friend_humansbestfriend-network (172.19.0.6): icmp_seq=2 ttl=64 time=0.070 ms
64 bytes from humans-best-friend-worker-1.humans-best-friend_humansbestfriend-network (172.19.0.6): icmp_seq=3 ttl=64 time=0.071 ms
64 bytes from humans-best-friend-worker-1.humans-best-friend_humansbestfriend-network (172.19.0.6): icmp_seq=4 ttl=64 time=0.066 ms
64 bytes from humans-best-friend-worker-1.humans-best-friend_humansbestfriend-network (172.19.0.6): icmp_seq=5 ttl=64 time=0.069 ms

--- ebfdd9021f04 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4103ms
rtt min/avg/max/mdev = 0.066/0.076/0.105/0.014 ms
```