

Rapport Du Projet Docker

Fait le :21/12/2023

Par : Diallo Assane

Barbich Karima

Marzouk Karim

Introduction

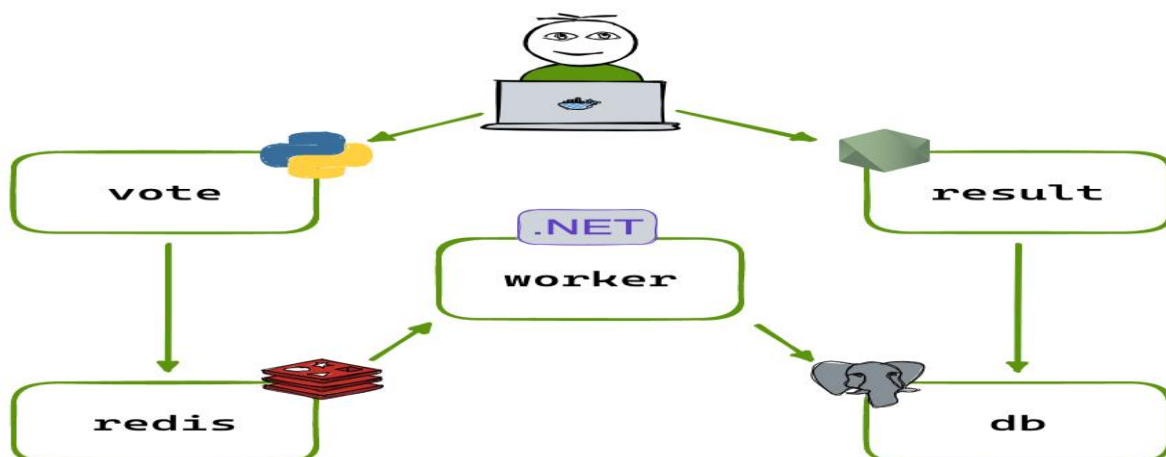
Contexte du Projet

Le monde moderne de l'ingénierie logicielle et des systèmes distribués présente des défis et des opportunités sans précédent. Dans ce contexte, le projet "HumansBestFriend", développé dans le cadre de notre programme cette année, se pose comme une réalisation exemplaire de l'intégration de technologies variées dans une application distribuée. Ce projet, accessible sur GitHub, est un terrain fertile pour explorer les aspects pratiques et théoriques du déploiement d'applications en conteneurs, de la gestion des bases de données, des files d'attente, et du traitement de données en temps réel.

Objectifs du Projet

Le projet "HumansBestFriend" vise à illustrer les concepts clés du génie logiciel et de l'architecture des systèmes dans un environnement Dockerisé. L'application, construite à l'aide de plusieurs langages de programmation tels que Python, Node.js et .NET, et exploitant des systèmes de gestion de base de données comme Redis et Postgres, offre une plateforme idéale pour comprendre la complexité et les nuances des applications distribuées.

Architecture globale



Technologies utilisées



Mise en Œuvre du Projet

Installation et Configuration

La mise en œuvre du projet a commencé par l'installation des prérequis nécessaires, notamment Docker et Docker Compose, pour assurer un environnement adéquat pour le déploiement des conteneurs de l'application.

Sur un système d'exploitation Ubuntu, les commandes suivantes ont été exécutées pour installer Docker Compose :

- Mise à jour des paquets existants :
`sudo apt install docker-compose`
- Installation de Docker Compose :
`sudo apt install docker-compose`

```
user@user:~/ynov-resources/2023/m2/dataeng/humans-best-friend$ sudo apt install docker-compose
[sudo] password for user:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
docker-compose is already the newest version (1.29.2-1).
0 upgraded, 0 newly installed, 0 to remove and 45 not upgraded.
user@user:~/ynov-resources/2023/m2/dataeng/humans-best-friend$ sudo apt update
Hit:1 http://ports.ubuntu.com/ubuntu-ports jammy InRelease
Get:2 http://ports.ubuntu.com/ubuntu-ports jammy-updates InRelease [119 kB]
Get:3 https://download.docker.com/linux/ubuntu jammy InRelease [48.8 kB]
Hit:4 http://ports.ubuntu.com/ubuntu-ports jammy-backports InRelease
Get:5 http://ports.ubuntu.com/ubuntu-ports jammy-security InRelease [110 kB]
Fetched 278 kB in 2s (129 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
45 packages can be upgraded. Run 'apt list --upgradable' to see them.
```

Préparation des Images

Une fois Docker Compose installé, nous avons procédé à la préparation des images Docker pour chaque service de l'application "HumansBestFriend". Pour cela, nous avons utilisé un fichier dédié nommé `docker-compose.build.yml`, qui contient les instructions nécessaires pour construire les images sans mettre en cache les étapes intermédiaires, permettant ainsi de s'assurer que les dernières modifications sont prises en compte.

Voici un aperçu de la manière dont nous avons configuré chaque service :

- Service Worker : Ce service utilise un environnement .NET pour les tâches de traitement en arrière-plan. Le contexte de build est spécifié comme `./worker`, indiquant que Docker doit utiliser le Dockerfile situé dans le répertoire `worker` pour construire l'image.
- Service Vote : L'interface de vote, construite avec Python, inclut une configuration healthcheck pour s'assurer que le service fonctionne correctement avant qu'il soit considéré comme sain.
- Service Seed Data : Responsable de l'initialisation de la base de données avec des données de départ, ce service est configuré avec un contexte de build et une politique de restart à `no`, signifiant qu'il ne doit pas redémarrer automatiquement s'il s'arrête.
- Service Result : Ce service rassemble et affiche les résultats des votes. Il est construit à partir du contexte fourni dans le répertoire `./result`.

- Base de Données (db) : Utilisant l'image postgres:15-alpine, le service de base de données comprend un script de vérification de santé situé à /healthchecks/postgres.sh, que Docker Compose exécutera à des intervalles spécifiés.
- Redis : Un service Redis est utilisé pour la mise en cache et la messagerie. Il utilise l'image par défaut redis et inclut une vérification de santé similaire à celle du service de base de données.

Commande de Construction

```
user@user:~/ynov-resources/2023/m2/dataeng/humans-best-friend$ vi docker-compose.build.yml
user@user:~/ynov-resources/2023/m2/dataeng/humans-best-friend$ docker-compose -f docker-compose.build.yml build --no-cache
db uses an image, skipping
redis uses an image, skipping
Building worker
[+] Building 128.9s (5/15)
=> [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@sha256:4be8ff7cb847f9e7e1ac194928b8a6d41956af89f934b61a2ce64dc8563471c
=> resolve mcr.microsoft.com/dotnet/sdk:7.0@sha256:4be8ff7cb847f9e7e1ac194928b8a6d41956af89f934b61a2ce64dc8563471c
=> sha256:4be8ff7cb847f9e7e1ac194928b8a6d41956af89f934b61a2ce64dc8563471c 1.79kB / 1.79kB
=> sha256:4297c75b7a51d8f6ab92eb814c26f99c765d6dcf638064196cd7a272c520a 2.81kB / 2.81kB
=> sha256:7e61b4d8e78c79e84f34f8995df5446a32856f5d381a9297412284f285eb 5.33kB / 5.33kB
=> sha256:2244784f264b3556a276558fbc21ada79a13dfff850e372b8f511318a7c93 4.19MB / 38.84MB
=> sha256:dec49b91a4ebc81feacfb3bd7a38f186d999a3f3f1a3e7f1541b8aa6a55ff0 4.19MB / 38.72MB
=> sha256:6454c385e5b7e25d529b148b273274d4166d6b580857a521af1ba36788123ab 8B / 14.92MB
=> [stage-1 1/3] FROM mcr.microsoft.com/dotnet/runtime:7.0@sha256:b41a241da8624e65544d83ccc642152f18a751082d1ea1a912e238b825953
=> [stage-1 1/3] FROM mcr.microsoft.com/dotnet/runtime:7.0@sha256:b41a241da8624e65544d83ccc642152f18a751082d1ea1a912e238b825953
```

La commande suivante a été exécutée dans le terminal pour construire les images :

```
docker-compose -f docker-compose.build.yml build --no-cache
```

Vérification des Images Docker

Après avoir construit les images Docker pour chaque service, nous avons procédé à une étape de vérification pour nous assurer que toutes les images étaient correctement créées et prêtes à être utilisées. La commande suivante a été exécutée pour lister toutes les images Docker disponibles sur le système :

docker images

```
>>> nothing to do, 'docker.io/library/humans-best-friend_worker'
a server@ubuntu: server:~/humansbestfriend/ynov-resources/2023/m2/dataeng/humans.best.friend$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
humans-best-friend_worker   latest             b52a0a06dced       14 seconds ago     194MB
humans-best-friend_result   latest             833b13dc864f       2 minutes ago      224MB
humans-best-friend_seed-data latest             d48f50f267b0       2 minutes ago      129MB
humans-best-friend_vote     latest             766c48e08963       3 minutes ago      154MB
registry              2                 909c3ff012b7       2 weeks ago        25.4MB
```

Résultats de la Commande

La sortie de la commande a affiché une liste des images récemment construites, y compris :

humans-best-friend_worker

humans-best-friend_seed-data

humans-best-friend_result

humans-best-friend_vote

Préparation pour le Registre Docker

Suite à la vérification des images Docker construites, l'étape suivante fut de les taguer en préparation pour leur envoi vers notre registre Docker local. Cette étape est essentielle pour organiser les images et les préparer pour le déploiement.

Taguage des Images Docker

Nous avons utilisé la commande suivante pour taguer l'image du service worker :

`docker tag humans-best-friend_worker localhost:5000/worker:v1`

```
u-server@ubuntu-server:~/humansbestfriend/ynov-resources/2023/m2/dataeng/humans-best-friend$ docker tag humans-best-friend_worker localhost:5000/r_worker:v1
u-server@ubuntu-server:~/humansbestfriend/ynov-resources/2023/m2/dataeng/humans-best-friend$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
humans-best-friend_worker   latest             b52a0a66ced        3 minutes ago      194MB
localhost:5000/r_worker   v1                b52a0a66ced        3 minutes ago      194MB
humans-best-friend_result   latest             833b13dc864f       5 minutes ago      224MB
humans-best-friend_seed-data latest             d48f50f267b0       6 minutes ago      129MB
humans-best-friend_vote     latest             f66c46e08963       6 minutes ago      154MB
registry              2                 909c3ff012b7       2 weeks ago        25.4MB
```

Envoi des Images au Registre Docker

Poussée de l'Image vers le Registre Local

Une fois les images correctement taguées, nous avons entrepris de les pousser vers notre registre Docker local. Cette opération permet de stocker les images dans un registre centralisé, facilitant le déploiement et le partage entre différents environnements de développement et de production. La commande exécutée a été :

`docker push localhost:5000/worker:v1`

```
u-server@ubuntu-server:~/humansbestfriend/ynov-resources/2023/m2/dataeng/humans-best-friend$ docker push localhost:5000/r_worker:v1
The push refers to repository [localhost:5000/r_worker:v1]
ec1648bf4d1b: Pushed
7cc2871b85cc: Pushed
5ee537aacf03: Pushed
7c73e219acdd: Pushed
d47be6633206: Pushed
ebcd30d4cca4: Pushed
v1: digest: sha256:e4ae1e024534c87b726fd5219322ab1a8f9e5f1b28548dbcbffa3960f2d7d6be size: 1577
```

La sortie de la commande confirme que les différentes couches de l'image ont été poussées avec succès vers le registre local

Consultation des Images dans le Registre

Pour confirmer que les images ont été correctement poussées et répertoriées dans le registre Docker local, la commande suivante a été exécutée :

`curl localhost:5000/v2/_catalog`

Cette requête à l'API du registre Docker retourne la liste des dépôts d'images disponibles. La sortie montre que l'image du service worker est désormais disponible dans le registre.

```
u-server@ubuntu-server:~/humansbestfriend/ynov-resources/2023/m2/dataeng/humans-best-friend$ curl localhost:5000/v2/_catalog
{"repositories":["r_result","r_worker"]}
u-server@ubuntu-server:~/humansbestfriend/ynov-resources/2023/m2/dataeng/humans-best-friend$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED    STATUS    PORTS
3256b31cc495   registry:2 "/entrypoint.sh /etc " 3 hours ago Up 3 hours 0.0.0.0:5000->5000/tcp, :::5000->5000/tcp   NAMES registry
u-server@ubuntu-server:~/humansbestfriend/ynov-resources/2023/m2/dataeng/humans-best-friend$ docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED    STATUS    PORTS
3256b31cc495   registry:2 "/entrypoint.sh /etc " 3 hours ago Up 3 hours 0.0.0.0:5000->5000/tcp, :::5000->5000/tcp   NAMES registry
u-server@ubuntu-server:~/humansbestfriend/ynov-resources/2023/m2/dataeng/humans-best-friend$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
humans-best-friend_worker   latest             b52a0a66ced        11 minutes ago      194MB
localhost:5000/r_worker   v1                b52a0a66ced        11 minutes ago      194MB
humans-best-friend_result   latest             833b13dc864f       13 minutes ago      224MB
localhost:5000/r_result   v1                833b13dc864f       13 minutes ago      224MB
humans-best-friend_seed-data latest             d48f50f267b0       13 minutes ago      129MB
humans-best-friend_vote     latest             f66c46e08963       14 minutes ago      154MB
registry              2                 909c3ff012b7       2 weeks ago        25.4MB
```

État des Conteneurs Docker

En parallèle, pour surveiller l'état actuel des conteneurs Docker, les commandes suivantes ont été utilisées :

`docker ps`

`docker ps -a`

```

u-server@ubuntu-server:~/humansbestfriend/ynov-resources/2023/m2/dataeng/humans-best-friend$ docker tag humans-best-friend_worker localhost:5000/r_worker:v1
u-server@ubuntu-server:~/humansbestfriend/ynov-resources/2023/m2/dataeng/humans-best-friend$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
humans-best-friend_worker   latest             b52aa0a66ced       3 minutes ago      194MB
localhost:5000/r_worker     v1                 b52aa0a66ced       3 minutes ago      194MB
humans-best-friend_result   latest             833b13dc864f       5 minutes ago      224MB
humans-best-friend_seed-data latest             d48f50f267b0       6 minutes ago      129MB
humans-best-friend_vote     latest             f66c46e08963       6 minutes ago      154MB
registry              2                  909c3ff012b7       2 weeks ago        25.4MB
u-server@ubuntu-server:~/humansbestfriend/ynov-resources/2023/m2/dataeng/humans-best-friend$ docker push localhost:5000/r_worker:v1
The push refers to repository [localhost:5000/r_worker]
ec1648bf4d1b: Pushed
7cc2871b85cc: Pushed
5ee37aacfb3: Pushed
7c73e219acd: Pushed
d47be6633206: Pushed
ebcd39d4cea4: Pushed
v1: digest: sha256:e4ae1e024554c87b726fd5219322ab1a8f9e5f1b28548dbcffa3960f2d7d6be size: 1577
u-server@ubuntu-server:~/humansbestfriend/ynov-resources/2023/m2/dataeng/humans-best-friend$
u-server@ubuntu-server:~/humansbestfriend/ynov-resources/2023/m2/dataeng/humans-best-friend$
u-server@ubuntu-server:~/humansbestfriend/ynov-resources/2023/m2/dataeng/humans-best-friend$
u-server@ubuntu-server:~/humansbestfriend/ynov-resources/2023/m2/dataeng/humans-best-friend$ curl localhost:5000/v2/_catalog
{"repositories":["r_worker"]}

```

Envoi de l'Image du Service result

Après avoir traité l'image du service worker, nous avons répété le processus pour le service result. La commande suivante a été exécutée pour taguer et pousser l'image :

```
docker push localhost:5000/r_result:v1
```

Cette action a permis d'envoyer l'image result vers le registre, où elle sera stockée aux côtés de l'image worker précédemment poussée.

Vérification du Registre

Pour confirmer que le registre Docker local contenait désormais les deux images nécessaires, nous avons utilisé la commande :

```
curl localhost:5000/v2/_catalog
```

```

u-server@ubuntu-server:~/humansbestfriend/ynov-resources/2023/m2/dataeng/humans-best-friend$ docker tag humans-best-friend_result localhost:5000/r_result:v1
u-server@ubuntu-server:~/humansbestfriend/ynov-resources/2023/m2/dataeng/humans-best-friend$ docker push localhost:5000/r_result:v1
The push refers to repository [localhost:5000/r_result]
48040607423d: Pushed
c039489fd83: Pushed
8c8354d5c36a: Pushed
3d0888c721a0: Pushed
449afa8df174: Pushed
d267bed80f24: Pushed
68ea00da724: Pushed
88a1fe82f870: Pushed
fa348aca89c0: Pushed
6a02db9fb904: Pushed
7292cf786aa8: Pushed
v1: digest: sha256:8d896afedab20f2d8941f8aab9914f8986bb20be2a5e73bbdb976ed2a7accba size: 2625
u-server@ubuntu-server:~/humansbestfriend/ynov-resources/2023/m2/dataeng/humans-best-friend$ curl localhost:5000/v2/_catalog
{"repositories":["r_result","r_worker"]}

```

Finalisation du Registre des Images Docker

Avec les services worker et result déjà poussés vers notre registre Docker local, nous avons poursuivi avec le service de seed-data.

Taguage et Envoi de l'Image du Service seed-data

Pour le service de seed-data, qui est crucial pour l'initialisation de notre base de données avec des données de départ, nous avons exécuté la commande suivante pour le taguage et la poussée de l'image :

```
docker tag humans-best-friend_seed-data localhost:5000/r_seed:v1
```

```
docker push localhost:5000/r_seed:v1
```

```

u-server@ubuntu-server:~/humansbestfriend/ynov-resources/2023/m2/dataeng/humans-best-friend$ docker tag humans-best-friend_seed-data localhost:5000/r_seed:v1
u-server@ubuntu-server:~/humansbestfriend/ynov-resources/2023/m2/dataeng/humans-best-friend$ docker push localhost:5000/r_seed:v1
The push refers to repository [localhost:5000/r_seed]
468ddb8797e2: Pushed
00b4dd19425d: Pushed
05f1b8eda60c: Pushed
1add4ecfaa91: Pushed
4cec408bace: Pushed
a1e3c54d75a8: Pushed
661ecc6e457f: Pushed
384858cc0ef: Pushed
7292cf786aa8: Mounted from r_result
v1: digest: sha256:0645ce0582a9102d1de1d5451bd29dcb5a6f8275c755bd679cb5634a8845 size: 2203
u-server@ubuntu-server:~/humansbestfriend/ynov-resources/2023/m2/dataeng/humans-best-friend$ curl localhost:5000/v2/_catalog
{"repositories":["r_result","r_seed","r_worker"]}

```

Mise en Registre de l'Image du Service de Vote

Avec les services antérieurs déjà en place, nous avons procédé à la même opération pour le service de vote de notre application, qui est un composant clé pour l'interface utilisateur.

Taguage et Envoi de l'Image vote

La commande utilisée pour taguer et pousser l'image vote vers le registre Docker local a été :

```
docker tag humans-best-friend_vote localhost:5000/r_vote:v1
```

```
docker push localhost:5000/r_vote:v1
```

```
q-server@ubuntu-server:~/humansbestfriend/ynov-resources/2023/m2/dataeng/humans-best-friend$ docker tag humans-best-friend_vote localhost:5000/r_vote:v1
q-server@ubuntu-server:~/humansbestfriend/ynov-resources/2023/m2/dataeng/humans-best-friend$ docker push localhost:5000/r_vote:v1
The push refers to repository [localhost:5000/r_vote]
c2481e293519: Pushed
ef76d4d9a9db: Pushed
d6827379ed87: Pushed
d8be5fb7c521: Pushed
796fe640de72: Pushed
3a0c08defe69: Pushed
8af0219f3527: Pushed
62ee4feb598: Pushed
384858ccd7ef: Mounted from r_seed
7292c7786aa8: Mounted from r_seed
v1: digest: sha256:adf74dcdd39ee406b9495b5692450595ff8c94c8335c7fe422ecf3d64ec2ed21 size: 2414
q-server@ubuntu-server:~/humansbestfriend/ynov-resources/2023/m2/dataeng/humans-best-friend$ docker images
REPOSITORY          TAG          IMAGE ID       CREATED        SIZE
humans-best-friend_worker   latest       b52aa0a66ced   18 minutes ago  194MB
localhost:5000/r_worker     v1          b52aa0a66ced   18 minutes ago  194MB
humans-best-friend_result   latest       833b13dc864f   20 minutes ago  224MB
localhost:5000/r_result     v1          833b13dc864f   20 minutes ago  224MB
humans-best-friend_seed-data latest       d48f50f267b0   21 minutes ago  129MB
localhost:5000/r_seed       v1          d48f50f267b0   21 minutes ago  129MB
humans-best-friend_vote     latest       f66c46e08963   21 minutes ago  154MB
localhost:5000/r_vote       v1          f66c46e08963   21 minutes ago  154MB
registry                2           909c3ff012b7   2 weeks ago    25.4MB
q-server@ubuntu-server:~/humansbestfriend/ynov-resources/2023/m2/dataeng/humans-best-friend$
```

Ces commandes ont assigné le tag v1 à notre image vote et l'ont transférée vers le registre Docker local. La sortie de la commande push a confirmé que les couches de l'image ont été correctement poussées vers le registre.

Intégration des Services de Base de Données et de Cache

Taguage et Poussée des Images de Redis et PostgreSQL

En complément des services applicatifs précédemment établis, nous avons assuré que nos services de cache et de base de données étaient également prêts pour un déploiement cohérent. Pour ce faire, nous avons procédé au taguage et à la poussée des images Docker officielles de Redis et PostgreSQL :

Pour Redis, la commande suivante a été exécutée :

```
q-server@ubuntu-server:~/humansbestfriend/ynov-resources/2023/m2/dataeng/humans-best-friend$ docker tag redis:alpine localhost:5000/r_redis:v1
q-server@ubuntu-server:~/humansbestfriend/ynov-resources/2023/m2/dataeng/humans-best-friend$ docker images
REPOSITORY          TAG          IMAGE ID       CREATED        SIZE
humans-best-friend_worker   latest       b52aa0a66ced   2 hours ago    194MB
localhost:5000/r_worker     v1          b52aa0a66ced   2 hours ago    194MB
humans-best-friend_result   latest       833b13dc864f   2 hours ago    224MB
localhost:5000/r_result     v1          833b13dc864f   2 hours ago    224MB
humans-best-friend_seed-data latest       d48f50f267b0   2 hours ago    129MB
localhost:5000/r_seed       v1          d48f50f267b0   2 hours ago    129MB
humans-best-friend_vote     latest       f66c46e08963   2 hours ago    154MB
localhost:5000/r_vote       v1          f66c46e08963   2 hours ago    154MB
postgres              15-alpine    c043c2b0d5ee   7 days ago     240MB
redis                  alpine       d2d4688f8cbe   13 days ago    41MB
localhost:5000/r_redis     v1          d2d4688f8cbe   13 days ago    41MB
registry                2           909c3ff012b7   2 weeks ago    25.4MB
q-server@ubuntu-server:~/humansbestfriend/ynov-resources/2023/m2/dataeng/humans-best-friend$ docker push localhost:5000/r_redis:v1
The push refers to repository [localhost:5000/r_redis]
97d8a180f428: Pushed
5f70bf18a086: Pushed
ac9404cab767: Pushed
cafe03ef2e40: Pushed
8ac59f7a1ab9: Pushed
b5fe2163aadd: Pushed
c2292eddb031: Pushed
3af4f8f59b76: Pushed
v1: digest: sha256:2c8f6ecce767cd325d3bd6390c56bab7d4ead99245c7f1bd607bb2b26079 size: 1989
q-server@ubuntu-server:~/humansbestfriend/ynov-resources/2023/m2/dataeng/humans-best-friend$ docker tag postgres:15-alpine localhost:5000/r_postgres:v1
q-server@ubuntu-server:~/humansbestfriend/ynov-resources/2023/m2/dataeng/humans-best-friend$ docker push localhost:5000/r_postgres:v1
The push refers to repository [localhost:5000/r_postgres]
dc78fc030a07: Pushed
431de0137131: Pushed
67793a05c4c0: Pushed
470a3180ab3d: Pushed
343a4310f73: Pushed
5e62c73ab93f: Pushed
5df9a38e251: Pushed
d43ab3a9972: Pushed
3af4f8f59b76: Mounted from r_redis
v1: digest: sha256:aeldf3e40fe311f7ac36037e6996305c97fc083a1918bd34935eab824507d3 size: 2192
q-server@ubuntu-server:~/humansbestfriend/ynov-resources/2023/m2/dataeng/humans-best-friend$ curl localhost:5000/v2/_catalog
{"repositories":["r_postgres","r_redis","r_result","r_seed","r_vote","r_worker"]}
```

```
docker tag redis:alpine localhost:5000/r_redis:v1
```

```
docker push localhost:5000/r_redis:v1
```


Pour PostgreSQL, nous avons utilisé :

```
docker tag postgres:15-alpine localhost:5000/r_postgres:v1
```

```
docker push localhost:5000/r_postgres:v1
```

Démarrage de l'Application avec Docker Compose

Après avoir préparé et vérifié toutes les images Docker nécessaires, nous sommes passés à l'étape finale de mise en œuvre du projet "HumansBestFriend": le lancement des services via Docker Compose.

Exécution de Docker Compose

La commande suivante a été utilisée pour démarrer tous les services définis dans notre fichier `docker-compose.yml` :

```
docker compose up
```

```

➤ docker-compose up --server-- --humanusbestfriend/ynow-resources/2023/mz/dataeng/humans-best-friend docker compose up
➤ Running 7/7
➤ Network humans-best-friend_humansbestfriend-network Created 1.1s
➤ Volume "humans-best-friend_db-data" Created 0.0s
➤ Container humans-best-friend-db-1 Created 0.1s
➤ Container humans-best-friend-redis-1 Created 0.1s
➤ Container humans-best-friend-result-1 Created 0.1s
➤ Container humans-best-friend-vote-1 Created 0.2s
➤ Container humans-best-friend-worker-1 Created 0.2s
➤ Container humans-best-friend-worker-1 Created 0.1s
attaching to humans-best-friend-db-1, humans-best-friend-redis-1, humans-best-friend-result-1, humans-best-friend-vote-1, humans-best-friend-worker-1
humans-best-friend-redis-1 | 1:~ 21 Dec 2023 14:29:48.658 # warning: Memory overcommit must be enabled without it, a background save or replication may fail under low memory condition. If
ing disabled, it can also cause failures without low memory condition, see https://github.com/jmahl/jmahl/issues/1328. To fix this issue add 'vm.overcommit_memory = 1' to /etc/sysct
l.conf and then reboot or run the command 'sysctl vm.overcommit_memory=1' for this take effect.
humans-best-friend-redis-1 | 1:~ 21 Dec 2023 14:29:48.658 # oob000000000 Redis is starting oob000000000
humans-best-friend-redis-1 | 1:~ 21 Dec 2023 14:29:48.658 # Redis version=7.2.3, bits=64, commit=00000000, modified=0, pid=1, just started
humans-best-friend-redis-1 | 1:~ 21 Dec 2023 14:29:48.658 # warning: no config file specified, using the default config. In order to specify a config file use redis-server /path/to/redis
.conf
humans-best-friend-redis-1 | 1:~ 21 Dec 2023 14:29:48.660 # monotonic clock: POSIX clock_gettime
humans-best-friend-redis-1 | 1:~ 21 Dec 2023 14:29:48.661 # Running modestandalone, port=6379.
humans-best-friend-redis-1 | 1:~ 21 Dec 2023 14:29:48.661 # Server initialized
humans-best-friend-redis-1 | 1:~ 21 Dec 2023 14:29:48.662 # Ready to accept connections tcp
humans-best-friend-db-1 | The files belonging to this database system will be owned by user "postgres".
humans-best-friend-db-1 | This user must also own the server process.
humans-best-friend-db-1 | The database cluster will be initialized with locale "en_US.utf8".
humans-best-friend-db-1 | The default database encoding has accordingly been set to "UTF8".
humans-best-friend-db-1 | The default text search configuration will be set to "english".
humans-best-friend-db-1 | Data page checksums are disabled.
humans-best-friend-db-1 | fixing permissions on existing directory /var/lib/postgresql/data ... ok
humans-best-friend-db-1 | creating subdirectories ... ok
humans-best-friend-db-1 | selecting dynamic shared memory implementation ... posix
humans-best-friend-db-1 | selecting default max_connections ... 100
humans-best-friend-db-1 | selecting default shared_buffers ... 128MB
humans-best-friend-db-1 | selecting default time zone ... UTC
humans-best-friend-db-1 | creating configuration files ... ok
humans-best-friend-db-1 | running bootstrap script ... ok
humans-best-friend-db-1 | sh: locale not found
humans-best-friend-db-1 | 2023-12-21 14:29:33.361 UTC [29] WARNING: no usable system locales were found
humans-best-friend-db-1 | 2023-12-21 14:29:33.361 UTC [29] INFO: Starting ganonic 21.2.0
humans-best-friend-db-1 | 2023-12-21 14:29:34.000000 [1] [INFO] Listening at: http://0.0.0.0:80 (1)
humans-best-friend-vote-1 | 2023-12-21 14:29:54.000000 [1] [INFO] Using worker: sync
humans-best-friend-vote-1 | 2023-12-21 14:29:54.000000 [2] [INFO] Booting worker with pid: 2
humans-best-friend-vote-1 | 2023-12-21 14:29:54.000000 [6] [INFO] Booting worker with pid: 6
humans-best-friend-vote-1 | 2023-12-21 14:29:54.000000 [9] [INFO] Booting worker with pid: 9
humans-best-friend-vote-1 | 2023-12-21 14:29:55.000000 [10] [INFO] Booting worker with pid: 10
humans-best-friend-worker-1 | waiting for db

```

Résultats du Démarrage

La sortie du terminal a indiqué la création réussie des éléments suivants :

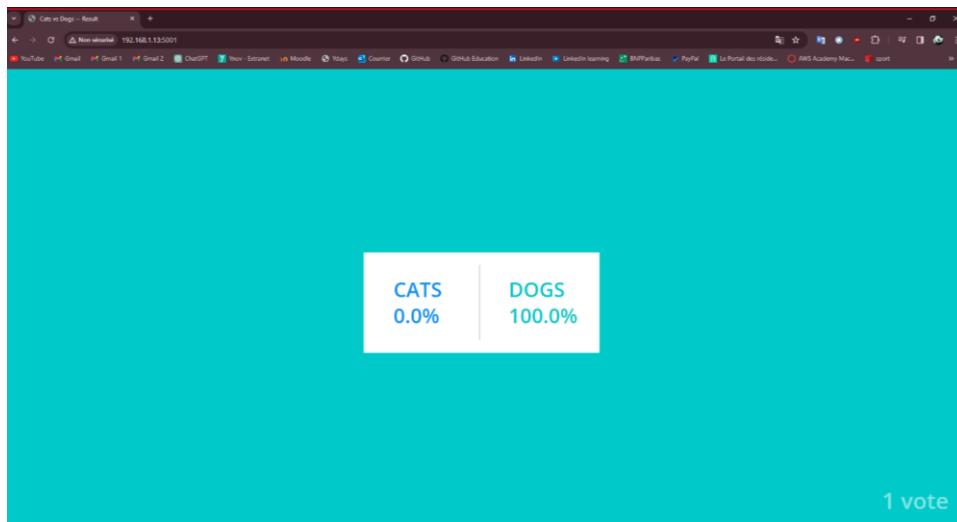
Réseau : Un réseau Docker nommé `humans-best-friend_humansbestfriend-network` a été créé pour faciliter la communication entre les conteneurs.

Volumes : Un volume Docker pour la persistance des données de la base de données Postgres a été établi.

Conteneurs : Plusieurs conteneurs pour nos services, y compris pour les bases de données, Redis, et les services applicatifs (db, redis, vote, worker, et result), ont été créés et démarrés.

Validation Fonctionnelle de l'Application

Après le démarrage de l'ensemble des services, nous avons procédé à la validation de l'application "HumansBestFriend" en accédant à l'interface de résultats du vote.



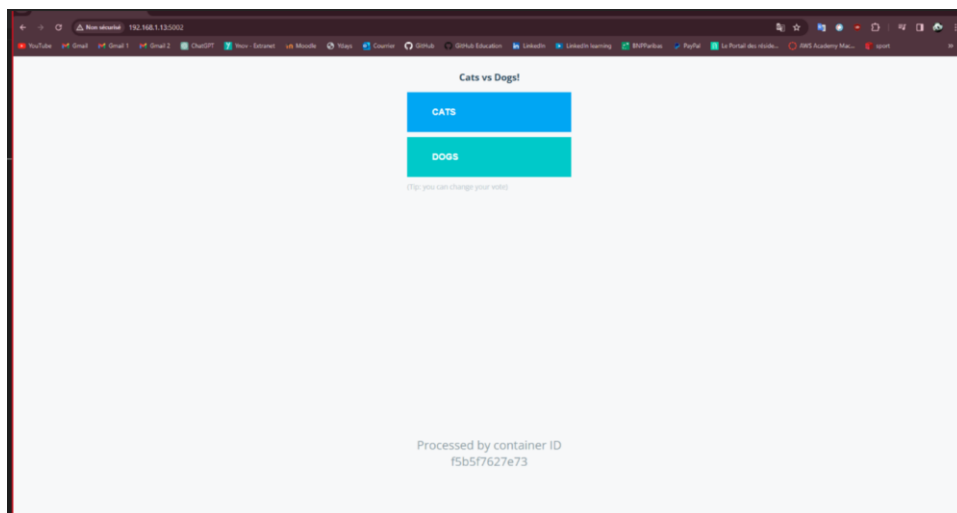
Interface Utilisateur du Vote

En ouvrant notre navigateur, nous avons navigué vers l'URL de l'interface de résultats, où nous avons été accueillis par un écran affichant les résultats actuels des votes. Le système a correctement enregistré et affiché un vote pour "Dogs", ce qui indique que le pipeline de traitement des votes est opérationnel.

Vérification des Résultats

L'interface affiche clairement "Dogs" avec 100% des votes, ce qui démontre que la fonctionnalité de vote fonctionne comme attendu et que la base de données enregistre et calcule les résultats en temps réel. Cela valide également la communication réussie entre le service de vote, la base de données, le service worker et le service de résultats.

Interface de Vote de l'Application "HumansBestFriend"



Interaction Utilisateur

L'étape suivante de notre validation a impliqué l'interaction avec l'interface de vote. En accédant à l'URL spécifiée, nous avons été présentés avec une interface utilisateur simple et intuitive, permettant de voter entre deux options : "Cats" et "Dogs".

Fonctionnalité de l'Interface

L'interface permet non seulement de voter mais aussi de modifier son vote, ce qui montre une fonctionnalité dynamique de l'application. Chaque vote est traité en temps réel et le résultat est mis à jour dans l'interface des résultats que nous avons vérifiée précédemment.

Traitement des Votes

En bas de la page, un identifiant de conteneur est affiché, ce qui indique quel conteneur a traité le vote. Cela démontre que l'équilibrage de charge fonctionne correctement et que notre architecture distribuée est non seulement opérationnelle mais également transparente pour l'utilisateur final.