

# TP Final Docker\_HumansBestFriend

## Intro

Ce projet a pour but de créer une application web permettant de voter pour son animal de compagnie préféré (chien ou chat), les votes seront ensuite stockés et nous pourrons consulter le résultat du vote. Pour cela, nous avons donc besoin de créer 5 containers :

- Un service "vote" pour l'interface de vote
- Une base Redis
- Une base postgres
- Un nommé "résultat" pour la gestion du résultat des votes
- Un worker pour faire le lien entre redis et postgres

Pour cela, nous avons récupéré le répertoire Github associé puis par étape :

- 1) Ecrire le fichier docker-compose.build.yml pour créer les images à utiliser
- 2) Tagger les images déjà créer et les push dans le container Registry pour utiliser nos propres images
- 3) Ecrire le fichier compose.yml pour créer et lancer nos 5 containers

## 1) Docker-compose.build.yml

Ce fichier nous permet de configurer la construction de notre application Docker et ainsi, créer les images.

Nous allons décomposer ce fichier :

```
services:
  worker:
    build:
      context: ./worker
```

Le *context* nous indique où se trouve le Dockerfile et mes fichiers nécessaires à la construction de l'image.

```
vote:
  build:
```

```
context: ./vote
healthcheck:
  test: ["CMD", "curl", "-f", "http://localhost"]
  interval: 15s
  timeout: 5s
  retries: 3
  start_period: 10s
```

Cette section “*healthcheck*” est utilisée pour définir un test de santé pour le service. Un test de santé permet à Docker de vérifier si le service est en cours d'exécution correctement.

On définit dans la section “*interval*” que ces tests de santé vont être fait toutes les 15 sec. “*Timeout*” est le temps d'attente maximal pour qu'un test de santé réussisse. “*Retries*” signifie nombre de tentatives avant de considérer le test de santé comme échoué.

```
seed-data:
  build:
    context: ./seed-data
  restart: "no"

result:
  build:
    context: ./result

db:
  image: postgres:15-alpine
```

Ici, on spécifie l'image Docker que l'on veut utiliser.

```
healthcheck:
  test: /healthchecks/postgres.sh
  interval: "5s"

redis:
  image: redis
  healthcheck:
    test: /healthchecks/redis.sh
    interval: "5s"
```

## 2) compose.yml

Voici une configuration Docker Compose qui définit plusieurs services pour l'application "HumansBestFriend".

```
services:
  vote:
    image: localhost:5000/vote:v0
    depends_on:
      - redis
    healthcheck:
      test: ["CMD", "curl", "-f", "http://localhost"]
      interval: 15s
      timeout: 5s
      retries: 3
      start_period: 10s
    volumes:
      - ./vote:/usr/local/app
```

Cette section montre de quelle manière sont configurés les volumes. Par exemple, le service "vote" monte le répertoire local ./vote dans le répertoire /usr/local/app du container.

```
ports:
  - "5002:80"
```

Spécifie la liaison de ports entre l'hôte et le container. Par exemple, le service vote expose le port 80 à l'hôte sur le port 5002.

```
networks:
  - humansbestfriend-network
```

Spécifie les réseaux auxquels chaque service est connecté. Ici, tous les services sont connectés au réseau "humansbestfriend-network".

```
result:
  image: localhost:5000/result:v0
  build: ./result
```

Pour le service "result", cela spécifie le chemin vers le dossier contenant le Dockerfile pour construire l'image localement. Il indique que l'image doit être construite à partir du contexte ./result.

```
entrypoint: ["nodemon", "--inspect=0.0.0.0", "server.js"]
```

Pour le service “*result*”, cela spécifie la commande d'entrée à exécuter lorsque le conteneur démarre.

```
depends_on:
  - db
volumes:
  - ./result:/usr/local/app
ports:
  - "5001:80"
  - "127.0.0.1:9229:9229"
networks:
  - humansbestfriend-network

worker:
  image: localhost:5000/worker:v0
  build:
    context: ./worker
  depends_on:
    - redis
    - db
  networks:
    - humansbestfriend-network
```

### 3) Construction de l'application

Pour build nos images, voici la commande que nous avons utilisée :

```
groov@dataeng-lab-vm:~/Human-best-friend/2023/m2/dataeng/humans-best-friend$ nano docker-compose.build.yml
groov@dataeng-lab-vm:~/Human-best-friend/2023/m2/dataeng/humans-best-friend$ docker compose -f docker-compose.build.yml build
[+] Building 34.2s (17/45)
=> [worker build 1/17] FROM mcr.microsoft.com/dotnet/sdk:7.0@sha256:4be8ff7cb847f9e7ea1ac194920b0a6d41956af89f934b61a2ce64dc8563471c
=> sha256:0f0fbcade3825e1d159f6bcb6d9b3910e65deadeaa651a2f6a4108c3d8234568b 82.84MB / 180.94MB
=> extracting sha256:b5a0d5c14ba9ece1eecd5137c468d9a123372b0af2ed2c8c4446137730c90e5b
=> [worker stage-1 1/3] FROM mcr.microsoft.com/dotnet/runtime:7.0@sha256:b41a241da8624e65544dd83cbcc642152f10a751082d1ea1a912e238b8259533
=> sha256:bdf2c62d9548601f6df118ad7ddf984e15a0aa258cc56b4b77945fa07a6978e7 155B / 155B
=> extracting sha256:b5a0d5c14ba9ece1eecd5137c468d9a123372b0af2ed2c8c4446137730c90e5b
=> [worker internal] load build context
=> transferring context: 7.07kB
=> [vote base 1/5] FROM docker.io/library/python:3.11-slim@sha256:8f64a67710f3d981cf3008d6f9f1dbe61accd7927f165f4e37ea3f8b883ccc3f
=> resolve docker.io/library/python:3.11-slim@sha256:8f64a67710f3d981cf3008d6f9f1dbe61accd7927f165f4e37ea3f8b883ccc3f
=> sha256:8f64a67710f3d981cf3008d6f9f1dbe61accd7927f165f4e37ea3f8b883ccc3f 1.65kB / 1.65kB
=> sha256:233e4c53aebbd019c06f6d927b93de5292cfc5091ae22bcbdd9630e600107c750 1.37kB / 1.37kB
=> sha256:dd150e5400f1a756b7752931d3064a57f529325b74e18d56bc0a4e08adb9f95f0 6.93kB / 6.93kB
=> sha256:8ce3f2b601ccac03ff1858022363c325355bafba224123a4563dade58bc8e70f 3.51MB / 3.51MB
=> extracting sha256:8ce3f2b601ccac03ff1858022363c325355bafba224123a4563dade58bc8e70f
=> sha256:b05f8bf97bacb24c0f8ddcd5ea6764c49aff2a325c50071f9e2042b6e3d4286c 241B / 241B
=> sha256:6ac7ac839d9db5b76a12d1c4e07d66079748ed2dcddc51a644b95d492902e34d 12.84MB / 12.84MB
=> sha256:b5a30bc7f1f876ea3d8555c4c65117ee16c4e62241a1a64fd3f14ecb794edd5 3.39MB / 3.39MB
=> extracting sha256:6ac7ac839d9db5b76a12d1c4e07d66079748ed2dcddc51a644b95d492902e34d
=> [vote internal] load build context
=> transferring context: 6.46kB
=> [seed-data 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
=> sha256:79d902e3c05bb26b70c5bdf4942e7e6383b927e29d91349eca306a433ae41050 1.37kB / 1.37kB
=> sha256:4fd8d6bf114c05509e38026f76e0f5d82df784a88b099349ba2341269aaf0aa2 6.92kB / 6.92kB
=> sha256:8ce3f2b601ccac03ff1858022363c325355bafba224123a4563dade58bc8e70f 3.51MB / 3.51MB
=> extracting sha256:8ce3f2b601ccac03ff1858022363c325355bafba224123a4563dade58bc8e70f
=> sha256:171d5fbee177d6d62ded7b4d7e09a1d4d9ec25b93c7e13d7cce55aafa185ede27 11.89MB / 11.89MB
=> sha256:4572660747e0f9f82652c13b5d63a04a4ef2483d337a7a69b5c7e1e0925ff6a3 245B / 245B
=> sha256:9ba7876fd8272e267f8a70670150c8f701d282039597d894ce21830039ddfb2f 3.13MB / 3.13MB
=> extracting sha256:171d5fbee177d6d62ded7b4d7e09a1d4d9ec25b93c7e13d7cce55aafa185ede27
=> [seed-data internal] load build context
=> transferring context: 1.09kB
```

Ensuite, nous avons pull Postgres et push dans Registry :

```
Status: Download newer image for registry:latest
2d2b809572a1ed425bbb38fdbf4e79743e253ccafcc0350bcacc27efb5f9cf97f
groov@dataeng-lab-vm:~/Human-best-friend/2023/m2/dataeng/humans-best-friend$ docker push localhost:5000/redis:v0
The push refers to repository [localhost:5000/redis]
3cb938f918a7: Pushed
5f70bf18a086: Pushed
97c159eabe6d: Pushed
987aefc1915a: Pushed
3dbd108cdf2f: Pushed
5b323dfbafb0: Pushed
a3688c42eece: Pushed
7292cf786aa8: Pushing [=====] 70.39MB/74.77MB

kom adk re me/docker-registry-frontend v2 60d4b91e087a 8 years ago 200MB
groov@dataeng-lab-vm:~/registry$ docker pull postgres:15-alpine
15-alpine: Pulling from library/postgres
661ff4d9561e: Pull complete
e4a3f96ea8e5: Pull complete
0c1e2e159ea1: Pull complete
26c071a8426e: Pull complete
e9a1ba05d22c: Pull complete
efc39a79d7dc: Pull complete
72124e665f9e: Pull complete
aa569f3e770e: Pull complete
86d5fe07cb37: Pull complete
Digest: sha256:e2a22801fcab638f9491039f8257e9f719ab02e8c78c6a6f2c0349505f92dc35
Status: Download newer image for postgres:15-alpine
docker.io/library/postgres:15-alpine
groov@dataeng-lab-vm:~/registry$ git clone https://github.com/Mohamed-Diouf/Human-best-friend.git
Cloning into 'Human-best-friend':
Command 'git' not found, did you mean:
  command 'quit' from snap quit (1.0+git)
  command 'luit' from deb x11-utils (7.7+5build2)
  command 'guix' from deb guix (1.3.0-4)
  command 'git' from deb git (1:2.34.1-1ubuntu1.10)
  command 'guilt' from deb guilt (0.36-2)
See 'snap info <snapname>' for additional versions.
groov@dataeng-lab-vm:~/registry$ git clone https://github.com/Mohamed-Diouf/Human-best-friend.git
Cloning into 'Human-best-friend'...
remote: Enumerating objects: 402, done.
remote: Counting objects: 100% (402/402), done.
remote: Compressing objects: 100% (265/265), done.
remote: Total 402 (delta 64), reused 402 (delta 64), pack-reused 0
Receiving objects: 100% (402/402), 11.49 MiB | 17.33 MiB/s, done.
Resolving deltas: 100% (64/64), done.
```

Ensuite, nous avons exécuté la commande pour créer nos containers :

```
groov@dataeng-lab-vm:~/Human-best-friend/2023/m2/dataeng/humans-best-friend$ nano docker-compose.build.yml
groov@dataeng-lab-vm:~/Human-best-friend/2023/m2/dataeng/humans-best-friend$ docker compose -f docker-compose.build.yml build
[+] Building 34.2s (17/45)
=> [worker build 1/7] FROM mcr.microsoft.com/dotnet/sdk:7.0@sha256:4be8ff7cb847f9e7ea1ac194920b0a6d41956af89f934b61a2ce64dc8563471c
=> => sha256:0f0fbcade3825e1d159fbcc6d9b3910e65deadeaa651a2f6a4108c3d8234568b 82.84MB / 180.94MB
=> => extracting sha256:b5a0d5c14ba9ece1eecd5137c468d9a123372b0af2ed2c8c4446137730c90e5b
=> [worker stage-1 1/3] FROM mcr.microsoft.com/dotnet/runtime:7.0@sha256:b41a241da8624e65544dd83cbcc642152f10a751082d1ea1a912e238b8259533
=> => sha256:bdf2c62d9548601f6df118ad7ddf984e15a0aa258cc56b4b77945fa07a6978e7 155B / 155B
=> => extracting sha256:b5a0d5c14ba9ece1eecd5137c468d9a123372b0af2ed2c8c4446137730c90e5b
=> [worker internal] load build context
=> => transferring context: 7.07kB
=> [vote base 1/5] FROM docker.io/library/python:3.11-slim@sha256:8f64a67710f3d981cf3008d6f9f1dbe61accd7927f165f4e37ea3f8b883ccc3f
=> => resolve docker.io/library/python:3.11-slim@sha256:8f64a67710f3d981cf3008d6f9f1dbe61accd7927f165f4e37ea3f8b883ccc3f
=> => sha256:8f64a67710f3d981cf3008d6f9f1dbe61accd7927f165f4e37ea3f8b883ccc3f 1.65kB / 1.65kB
=> => sha256:233e4c53aebbd019c06f6d927b93de5292cfc5091ae22bcbd9630e600107c750 1.37kB / 1.37kB
=> => sha256:dd150e5400f1a756b7752931d3064a57f529325b74e18d56bc0a4e08adb9f95f0 6.93kB / 6.93kB
=> => sha256:8ce3f2b601ccac03ff1858022363c325355bafba224123a4563dade58bc8e70f 3.51MB / 3.51MB
=> => extracting sha256:8ce3f2b601ccac03ff1858022363c325355bafba224123a4563dade58bc8e70f
=> => sha256:b05f8bf97bacb24c0f8ddcd5ea6764c49aff2a325c50071f9e2042b6e3d4286c 241B / 241B
=> => sha256:6ac7ac839d9ddb5b76a12d1c4e07d66079748ed2dc51a644b95d492902e34d 12.84MB / 12.84MB
=> => sha256:b5a30bc7f1f876ea3d8555c4c65117eee16c4e62241a1a64fd3f14ecb794edd5 3.39MB / 3.39MB
=> => extracting sha256:6ac7ac839d9ddb5b76a12d1c4e07d66079748ed2dc51a644b95d492902e34d
=> [vote internal] load build context
=> => transferring context: 6.46kB
=> [seed-data 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
=> => sha256:79d902e3c05bb26b70c5bdf4942e7e6383b927e29d91349eca306a433ae41050 1.37kB / 1.37kB
=> => sha256:4fd8d6bf114c05509e38026f76a0f5d82df784a88b099349ba2341269aaf0aa2 6.92kB / 6.92kB
=> => sha256:8ce3f2b601ccac03ff1858022363c325355bafba224123a4563dade58bc8e70f 3.51MB / 3.51MB
=> => extracting sha256:8ce3f2b601ccac03ff1858022363c325355bafba224123a4563dade58bc8e70f
=> => sha256:171d5f5be177dd6d2ded7b4d7e09a1d4d9ec25b93c7e13d7cce55aafa185ede27 11.89MB / 11.89MB
=> => sha256:4572660747e0f9f82652c13b5d63a04a4ef2483d337a7a69b5c7e1e0925ff6a3 245B / 245B
=> => sha256:9ba7876fd8272e267f8a70670150c8f701d282039597d894ce21830039ddf2f 3.13MB / 3.13MB
=> => extracting sha256:171d5f5be177dd6d2ded7b4d7e09a1d4d9ec25b93c7e13d7cce55aafa185ede27
=> [seed-data internal] load build context
=> => transferring context: 1.09kB
```