

Rapport projet Docker Cat VS Dog

Guillaume POTIER
François JONATHAN
Florian LAUNAY

Enseignant :
Jean-Pascal MEWENEMESSE

Introduction :

Nous avons réalisé une application de vote en ligne tournant sur Docker.

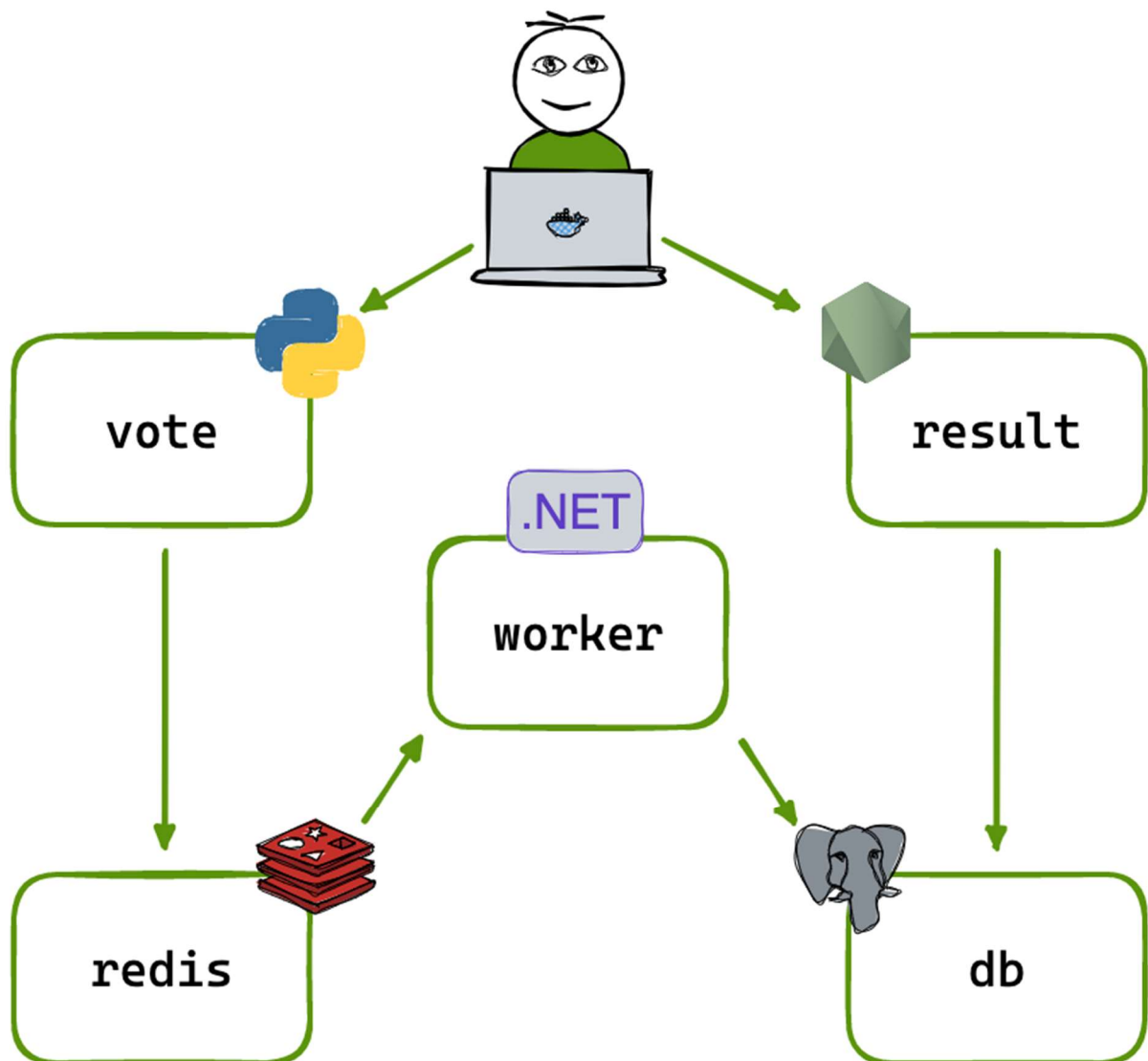
L'application HumansBestFriend n'accepte qu'un seul vote par navigateur client. Elle n'enregistre pas de votes supplémentaires si un vote a déjà été soumis par un client.

Ce n'est pas un exemple d'application distribuée correctement architecturée et parfaitement conçue... c'est juste un exemple simple des différents types de pièces et de langages que vous pourriez voir (files d'attente, données persistantes, etc.), et comment les gérer dans Docker à un niveau de base.

Cette application tourne en local sur une VM Linux sur l'IP 192.168.252.131 et les ports 5001 et 5002.

Nous n'avons pas réalisé l'étape de kubernetes afin de pouvoir voter de différentes machines car nous avons rencontré un problème lors de l'installation.

Voici l'architecture globale du projet :



Présentation de la procédure de mise en place :

Tout d'abord nous avons lancé un ESXI qui permet de créer et d'exécuter notre machine virtuelle (VM) sur un même serveur.



Ensuite, nous avons lancé notre VM afin d'y installer docker et docker compose.

Cela à l'aide du terminal Windows connecté en SSH avec la commande :

ssh potier@192.168.252.131

```
potier@k8s-master: ~  
Microsoft Windows [version 10.0.19045.3803]  
(c) Microsoft Corporation. Tous droits réservés.  
  
C:\Users\axbyb>ssh potier@192.168.252.131  
potier@192.168.252.131's password:  
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-78-generic x86_64)  
  
* Documentation:  https://help.ubuntu.com  
* Management:    https://landscape.canonical.com  
* Support:       https://ubuntu.com/advantage  
  
System information as of jeu. 04 janv. 2024 17:29:57 UTC  
  
System load:                0.3759765625  
Usage of /:                  74.1% of 17.20GB  
Memory usage:               15%  
Swap usage:                 0%  
Processes:                  209  
Users logged in:            0  
IPv4 address for br-13a2151c61da: 172.20.0.1  
IPv4 address for br-3beabed8634e: 172.23.0.1  
IPv4 address for br-9c373fc5e4b8: 172.19.0.1  
IPv4 address for br-d1bf68e1dd1e: 172.21.0.1  
IPv4 address for br-e1603a7a7458: 172.24.0.1  
IPv4 address for br-efc36d82348c: 172.18.0.1  
IPv4 address for docker0:    172.17.0.1  
IPv4 address for docker_gwbridge: 172.22.0.1  
IPv4 address for ens34:      192.168.252.131
```

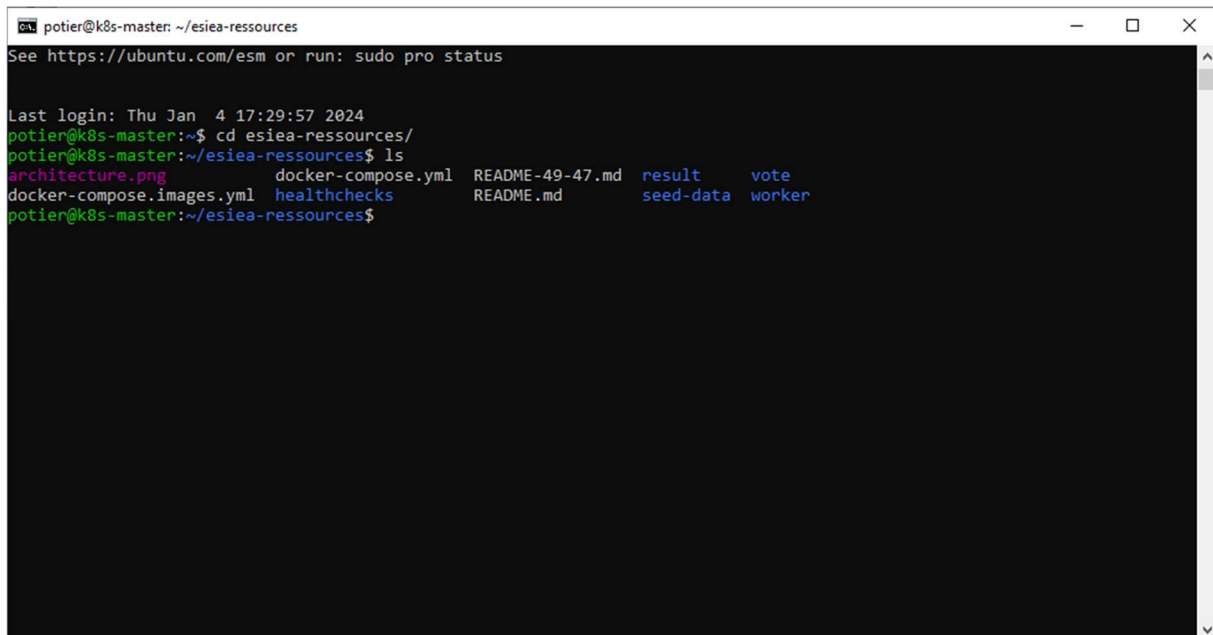
Grâce à la commande **ls**, voici le contenu du dossier projet.

Les fichiers présents ont été importés avec la commande :

git clone <https://github.com/pascalito007/esiea-ressources.git>

Et nous avons rajouté deux fichiers pour le bon fonctionnement du projet avec Docker :

1. **docker-compose.images.yml**
2. **docker-compose.yml**



```
potier@k8s-master: ~/esiea-ressources
See https://ubuntu.com/esm or run: sudo pro status

Last login: Thu Jan  4 17:29:57 2024
potier@k8s-master:~$ cd esiea-ressources/
potier@k8s-master:~/esiea-ressources$ ls
architecture.png  docker-compose.yml  README-49-47.md  result  vote
docker-compose.images.yml  healthchecks        README.md        seed-data  worker
potier@k8s-master:~/esiea-ressources$
```

Voici le fichier **docker-compose.images.yml**:

Ce fichier **docker-compose.images.yml** configure une application de vote avec plusieurs services (vote, result, worker, redis, db) et utilise des vérifications de santé personnalisées pour s'assurer du bon état de fonctionnement des services dépendants.

```

potier@k8s-master: ~/esiea-ressources
potier@k8s-master:~/esiea-ressources$ cat docker-compose.images.yml
services:
  vote:
    image: potier_jonathan_launay/examplevotingapp_vote
    depends_on:
      redis:
        condition: service_healthy
    ports:
      - "5000:80"
    networks:
      - front-tier
      - back-tier

  result:
    image: potier_jonathan_launay/examplevotingapp_result
    depends_on:
      db:
        condition: service_healthy
    ports:
      - "5001:80"
    networks:
      - front-tier
      - back-tier

  worker:
    image: potier_jonathan_launay/examplevotingapp_worker
    depends_on:
      redis:
        condition: service_healthy
      db:
        condition: service_healthy
    networks:
      - back-tier

  redis:
    image: redis:alpine
    volumes:
      - "/healthchecks:/healthchecks"
    healthcheck:
      test: /healthchecks/redis.sh
      interval: "5s"
    networks:
      - back-tier

  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

volumes:
  db-data:

networks:
  front-tier:

```

Le fichier **docker-compose.yml** étend le premier fichier en ajoutant des configurations de construction personnalisées, des healthchecks spécifiques, et introduit un nouveau service "seed" pour ajouter à la base de données les votes. Il est orienté vers un environnement de développement local avec des configurations adaptées à cette fin, telles que l'utilisation de nodemon et des ports spécifiés.

```

potier@k8s-master: ~/esiea-ressources
potier@k8s-master:~/esiea-ressources$ cat docker-compose.yml
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=0.0.0.0 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
      db:
        condition: service_healthy
    networks:
      - back-tier

  redis:
    image: redis:alpine
    volumes:
      - "/healthchecks:/healthchecks"
    healthcheck:
      test: /healthchecks/redis.sh
      interval: "5s"
    networks:
      - back-tier

  db:
    image: postgres:15-alpine
    environment:

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

  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:
potier@k8s-master:~/esiea-ressources$

```

Maintenant que tous nos fichiers de configuration sont prêts, lançons notre application à l'aide de docker.

Utilisons la commande :

docker compose up

Cette commande permet de construire les images et démarrer les services définis dans le fichier. Elle assure la création des conteneurs Docker, la configuration du réseau, le montage des volumes, et le lancement des services.

```
potier@kb-master: ~/esiea-ressources
volumes:
  - "db-data:/var/lib/postgresql/data"
  - "/healthchecks/healthchecks"
healthcheck:
  test: /healthchecks/postgres.sh
  interval: "5s"
networks:
  - back-tier

seed:
  build: ./seed-data
  profiles: ["seed"]
  depends_on:
    vote:
      condition: service_healthy
  networks:
    - front-tier
  restart: "no"

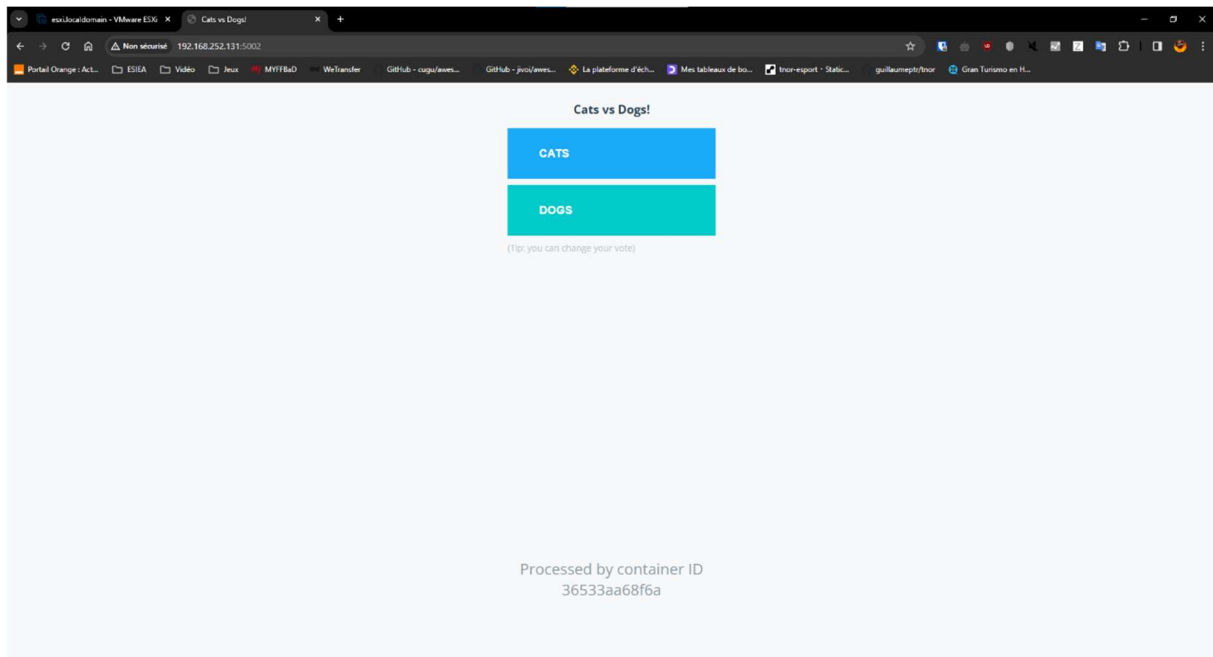
volumes:
  db-data:

networks:
  front-tier:
  back-tier:

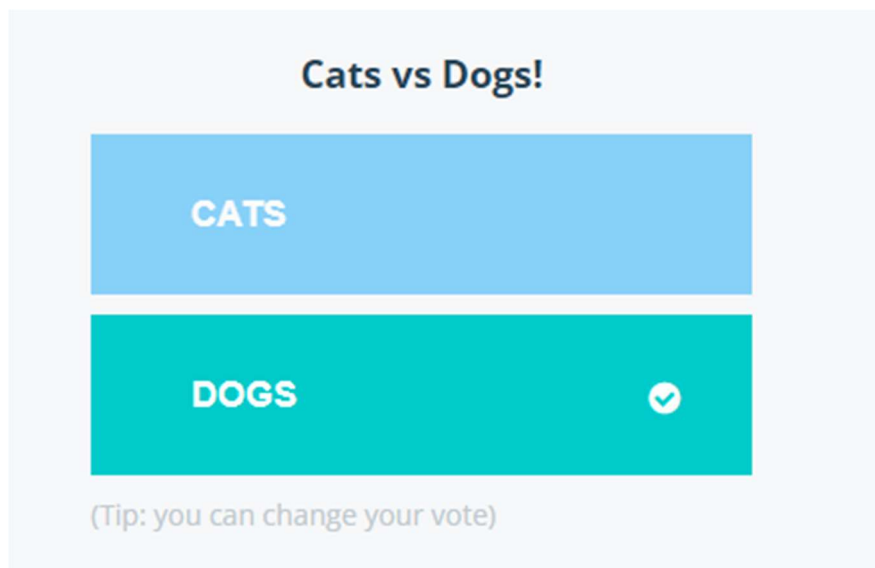
[1] Running S/Backend/esiea-ressources$ docker compose up
[+] Container esiea-ressources-redis-1 Created
[+] Container esiea-ressources-vote-1 Created
[+] Container esiea-ressources-db-1 Created
[+] Container esiea-ressources-worker-1 Created
[+] Container esiea-ressources-result-1 Created
Attaching to db-1, redis-1, result-1, vote-1, worker-1
redis-1 | 11:04 Jan 2024 17:32:35.702 # WARNING Memory overcommit must be enabled! Without it, a background save or replication may fail under low memory condition. Being disabled, it can also cause failures without low memory condition, see https://github.com/jemalloc/jemalloc/issues/1328. To fix this issue add 'vm.overcommit_memory = 1' to /etc/sysctl.conf and then reboot or run the command 'sysctl vm.overcommit_memory=1' for this to take effect.
redis-1 | 11:04 Jan 2024 17:32:35.702 # oOoOoOoOoOoOo Redis is starting oOoOoOoOoOoOo
redis-1 | 11:04 Jan 2024 17:32:35.702 # Redis version=7.2.3, bits=64, commit=00000000, modified=0, pid=1, just started
redis-1 | 11:04 Jan 2024 17:32:35.702 # Warning: no config file specified, using the default config. In order to specify a config file use redis-server /path/to/redis.conf
redis-1 | 11:04 Jan 2024 17:32:35.703 # monotonic clock: POSIX clock_gettime
redis-1 | 11:04 Jan 2024 17:32:35.706 # Running mode=standalone, port=6379.
redis-1 | 11:04 Jan 2024 17:32:35.708 # Server initialized
redis-1 | 11:04 Jan 2024 17:32:35.710 # Loading RDB produced by version 7.2.3
redis-1 | 11:04 Jan 2024 17:32:35.710 # RDB memory usage when created 1.04 Mb
redis-1 | 11:04 Jan 2024 17:32:35.710 # Done loading RDB, keys loaded: 0, keys expired: 0.
redis-1 | 11:04 Jan 2024 17:32:35.711 # DB loaded from disk: 0.002 seconds
redis-1 | 11:04 Jan 2024 17:32:35.711 # Ready to accept connections tcp
db-1 | PostgreSQL Database directory appears to contain a database; Skipping initialization
db-1 | 2024-01-04 17:32:35.846 UTC [1] LOG: starting PostgreSQL 15.5 on x86_64-pc-linux-musl, compiled by gcc (Alpine 13.2.1_git20231014) 13.2.1 20231014, 64-bit
db-1 | 2024-01-04 17:32:35.847 UTC [1] LOG: listening on IPv4 address "0.0.0.0", port 5432
db-1 | 2024-01-04 17:32:35.848 UTC [1] LOG: listening on IPv6 address "::", port 5432
db-1 | 2024-01-04 17:32:35.854 UTC [1] LOG: listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
db-1 | 2024-01-04 17:32:35.860 UTC [23] LOG: database system was interrupted; last known up at 2024-01-02 18:14:49 UTC
db-1 | 2024-01-04 17:32:35.876 UTC [23] LOG: database system was not properly shut down; automatic recovery in progress
db-1 | 2024-01-04 17:32:35.881 UTC [23] LOG: redo starts at 0/152BF50
db-1 | 2024-01-04 17:32:35.882 UTC [23] LOG: Invalid record length at 0/152BF88: wanted 24, got 0
db-1 | 2024-01-04 17:32:35.882 UTC [23] LOG: redo done at 0/152BF50 system usage: CPU: user: 0.00 s, system: 0.00 s, elapsed: 0.00 s
db-1 | 2024-01-04 17:32:35.887 UTC [21] LOG: checkpoint starting: end-of-recovery immediate wait
db-1 | 2024-01-04 17:32:35.892 UTC [21] LOG: checkpoint complete: wrote 3 buffers (0.0%); 0 WAL file(s) added, 0 removed, 0 recycled; write=0.002 s, sync=0.001 s, total=0.006 s; sync files=2, longest=0.001 s, average=0.001 s; distance=0 kB, estimate=0 kB
db-1 | 2024-01-04 17:32:35.902 UTC [1] LOG: database system is ready to accept connections
[+] Container esiea-ressources-redis-1 Created
[+] Container esiea-ressources-vote-1 Created
[+] Container esiea-ressources-db-1 Created
[+] Container esiea-ressources-worker-1 Created
[+] Container esiea-ressources-result-1 Created
Attaching to db-1, redis-1, result-1, vote-1, worker-1
redis-1 | 11:04 Jan 2024 17:32:35.702 # WARNING Memory overcommit must be enabled! Without it, a background save or replication may fail under low memory condition. Being disabled, it can also cause failures without low memory condition, see https://github.com/jemalloc/jemalloc/issues/1328. To fix this issue add 'vm.overcommit_memory = 1' to /etc/sysctl.conf and then reboot or run the command 'sysctl vm.overcommit_memory=1' for this to take effect.
redis-1 | 11:04 Jan 2024 17:32:35.702 # oOoOoOoOoOoOo Redis is starting oOoOoOoOoOoOo
redis-1 | 11:04 Jan 2024 17:32:35.702 # Redis version=7.2.3, bits=64, commit=00000000, modified=0, pid=1, just started
redis-1 | 11:04 Jan 2024 17:32:35.702 # Warning: no config file specified, using the default config. In order to specify a config file use redis-server /path/to/redis.conf
redis-1 | 11:04 Jan 2024 17:32:35.703 # monotonic clock: POSIX clock_gettime
redis-1 | 11:04 Jan 2024 17:32:35.706 # Running mode=standalone, port=6379.
redis-1 | 11:04 Jan 2024 17:32:35.708 # Server initialized
redis-1 | 11:04 Jan 2024 17:32:35.710 # Loading RDB produced by version 7.2.3
redis-1 | 11:04 Jan 2024 17:32:35.710 # RDB memory usage when created 1.04 Mb
redis-1 | 11:04 Jan 2024 17:32:35.710 # Done loading RDB, keys loaded: 0, keys expired: 0.
redis-1 | 11:04 Jan 2024 17:32:35.711 # DB loaded from disk: 0.002 seconds
redis-1 | 11:04 Jan 2024 17:32:35.711 # Ready to accept connections tcp
db-1 | PostgreSQL Database directory appears to contain a database; Skipping initialization
db-1 | 2024-01-04 17:32:35.846 UTC [1] LOG: starting PostgreSQL 15.5 on x86_64-pc-linux-musl, compiled by gcc (Alpine 13.2.1_git20231014) 13.2.1 20231014, 64-bit
db-1 | 2024-01-04 17:32:35.847 UTC [1] LOG: listening on IPv4 address "0.0.0.0", port 5432
db-1 | 2024-01-04 17:32:35.848 UTC [1] LOG: listening on IPv6 address "::", port 5432
db-1 | 2024-01-04 17:32:35.854 UTC [1] LOG: listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
db-1 | 2024-01-04 17:32:35.860 UTC [23] LOG: database system was interrupted; last known up at 2024-01-02 18:14:49 UTC
db-1 | 2024-01-04 17:32:35.876 UTC [23] LOG: database system was not properly shut down; automatic recovery in progress
db-1 | 2024-01-04 17:32:35.881 UTC [23] LOG: redo starts at 0/152BF50
db-1 | 2024-01-04 17:32:35.882 UTC [23] LOG: Invalid record length at 0/152BF88: wanted 24, got 0
db-1 | 2024-01-04 17:32:35.882 UTC [23] LOG: redo done at 0/152BF50 system usage: CPU: user: 0.00 s, system: 0.00 s, elapsed: 0.00 s
db-1 | 2024-01-04 17:32:35.887 UTC [21] LOG: checkpoint starting: end-of-recovery immediate wait
db-1 | 2024-01-04 17:32:35.892 UTC [21] LOG: checkpoint complete: wrote 3 buffers (0.0%); 0 WAL file(s) added, 0 removed, 0 recycled; write=0.002 s, sync=0.001 s, total=0.006 s; sync files=2, longest=0.001 s, average=0.001 s; distance=0 kB, estimate=0 kB
db-1 | 2024-01-04 17:32:35.902 UTC [1] LOG: database system is ready to accept connections
result-1 | [nodeemon] 3.0.2
result-1 | [nodeemon] to restart at any time, enter 'rs'
result-1 | [nodeemon] watching path(s): 'js,mjs,cjs,json'
result-1 | [nodeemon] watching extensions: js,mjs,cjs,json
result-1 | [nodeemon] starting 'node --inspect=0.0.0 server.js'
result-1 | Debugger listening on ws://0.0.0.0:9229/f933357d-83c3-4b05-8e5d-375932ed2962
result-1 | For help, see: https://nodejs.org/en/docs/inspector
result-1 | * Serving Flask app 'app'
result-1 | * Debug mode: on
result-1 | WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
result-1 | * Running on all addresses (0.0.0.0)
result-1 | * Running on http://127.0.0.1:80
result-1 | * Running on http://172.19.0.6:80
result-1 | Press CTRL+C to quit
result-1 | * Restarting with watchdog (inotify)
worker-2 | Connected to db
worker-2 | Found redis at 172.19.0.3
worker-3 | Connecting to redis
worker-3 | * Debugger is active!
worker-3 | * Debugger PIN: 115-003-073
result-1 | Thu, 04 Jan 2024 17:32:44 GMT body-parser deprecated undefined extended: provide extended option at server.js:67:17
result-1 | App running on port 80
result-1 | Connected to db
result-1 | 127.0.0.1 - - [04/Jan/2024 17:32:57] "GET / HTTP/1.1" 200 -
```


Voilà notre application est maintenant démarrée et à l'écoute sur les ports 5001 et 5002. Nous l'avons défini dans notre fichier **docker-compose.images.yml** .

Rendons-nous sur l'URL suivant : **192.168.252.131:5002** afin d'accéder à la page de vote.



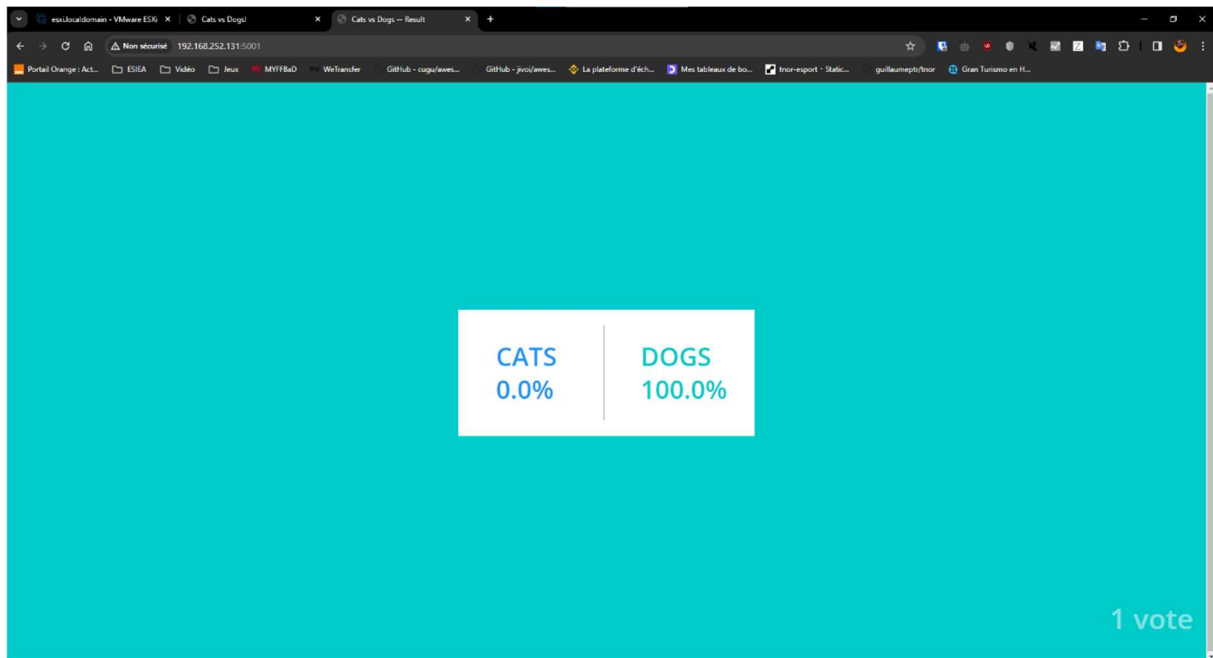
Nous votons pour le chien.



Ce vote va être transmis sur la base de données qui écoute sur le port 5001.

Allons donc voir sur l'URL : **192.168. 252.131:5001**

Nous obtenons bien le pourcentage de voix en direct et nous voyons bien notre voix pour le chien.



Pour le rendu de projet, nous l'avons push sur notre github personnel et déposé sur moodle.

