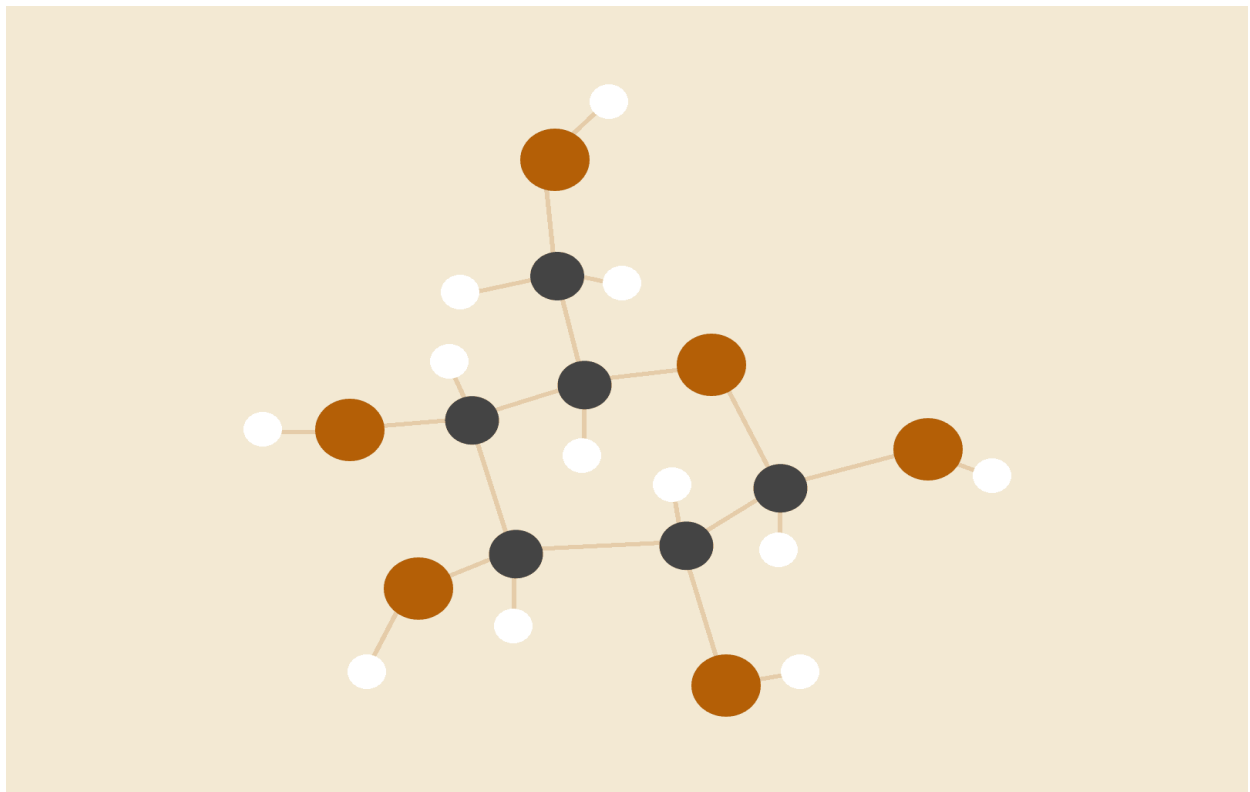


RAPPORT PROJET DE VIRTUALISATION



Audrey Jordan SOUOP

2023-2024

TC3 - 4A - Classe 49

INTRODUCTION

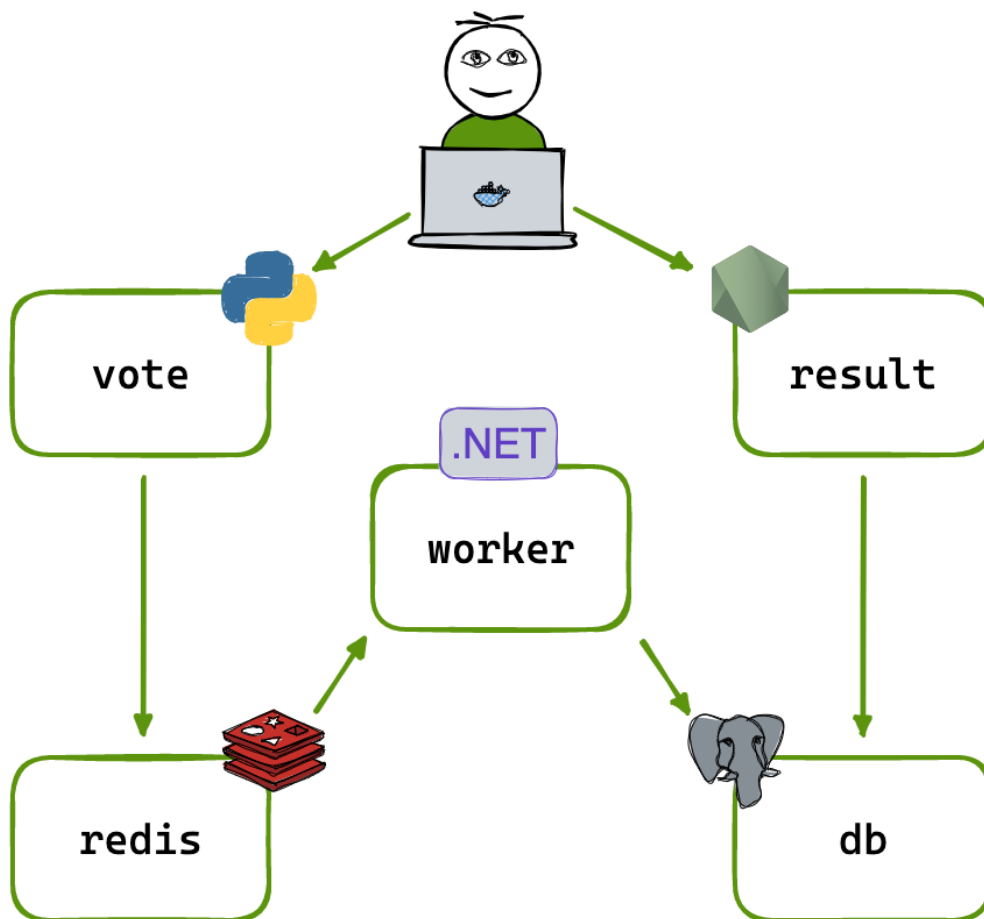
La virtualisation est une technologie informatique révolutionnaire qui permet de créer des environnements informatiques virtuels, distincts des systèmes physiques sous-jacents. Elle permet l'exécution simultanée de plusieurs systèmes d'exploitation et applications sur une seule machine physique. Cette approche offre une flexibilité inégalée, améliore l'utilisation des ressources matérielles et facilite la gestion et la sécurité des infrastructures informatiques.

HYPOTHÈSE

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 depuis un client.

Ceci n'est pas un exemple d'application distribuée parfaitement architecturée et conçue... C'est simplement un exemple simple des différents types de composants et de langages que l'on pourrait rencontrer (files d'attente, données persistantes, etc.) et de la manière de les gérer dans Docker à un niveau basique.

ARCHITECTURE



PROCÉDURE

Afin de mener à bien notre projet et de bien construire nos fichiers d'exécution, nous aurons besoin de respecter méticuleusement un certain nombre de règles avec Docker afin de ne créer aucun bug.

Pour débiter, nous allons commencer par builder chacune des images de base fourni:

On commence par builder l'image de result avec la commande **docker build result_image**. On obtient le résultat suivant:

```
houop@AN-0591p:~$ docker build -t result_image .
[+] Building 0.9s (10/10) FINISHED                                docker:default
=> [internal] load build definition from Dockerfile              0.0s
=> => transferring dockerfile: 212B                               0.0s
=> [internal] load .dockerignore                                  0.0s
=> => transferring context: 2B                                     0.0s
=> [internal] load metadata for docker.io/library/httpd:latest   0.7s
=> [1/6] FROM docker.io/library/httpd@sha256:f0a93744d8006e6f7ee5086c9ddccdcfa33d1091f15269a00547b4c382459c1f  0.0s
=> CACHED [2/6] RUN apt update -y                                0.0s
=> CACHED [3/6] RUN apt install -y iputils-ping                  0.0s
=> CACHED [4/6] RUN apt install -y inetutils-traceroute          0.0s
=> CACHED [5/6] RUN apt install -y iproute2                      0.0s
=> CACHED [6/6] RUN apt install -y curl telnet dnsutils vim      0.0s
=> exporting to image                                           0.0s
=> => exporting layers                                           0.0s
=> => writing image sha256:faf07ca4c950a15fa8f42ff5708d92d6bbec3434bc8808f8f15f72de2d3c1903  0.0s
=> => naming to docker.io/library/result_image                   0.0s

What's Next?
View a summary of image vulnerabilities and recommendations → docker scout quickview
```

Après cela, on build l'image de vote avec la commande **docker build vote_image**. On obtient ceci:

```
houop@AN-0591p:~$ docker build -t vote_image .
[+] Building 1.1s (10/10) FINISHED                                docker:default
=> [internal] load build definition from Dockerfile              0.1s
=> => transferring dockerfile: 212B                               0.0s
=> [internal] load .dockerignore                                  0.0s
=> => transferring context: 2B                                     0.0s
=> [internal] load metadata for docker.io/library/httpd:latest   0.9s
=> [1/6] FROM docker.io/library/httpd@sha256:f0a93744d8006e6f7ee5086c9ddccdcfa33d1091f15269a00547b4c382459c1f  0.0s
=> CACHED [2/6] RUN apt update -y                                0.0s
=> CACHED [3/6] RUN apt install -y iputils-ping                  0.0s
=> CACHED [4/6] RUN apt install -y inetutils-traceroute          0.0s
=> CACHED [5/6] RUN apt install -y iproute2                      0.0s
=> CACHED [6/6] RUN apt install -y curl telnet dnsutils vim      0.0s
=> exporting to image                                           0.0s
=> => exporting layers                                           0.0s
=> => writing image sha256:faf07ca4c950a15fa8f42ff5708d92d6bbec3434bc8808f8f15f72de2d3c1903  0.0s
=> => naming to docker.io/library/vote_image                     0.0s

What's Next?
View a summary of image vulnerabilities and recommendations → docker scout quickview
```

Après cela, et pour que notre exécution se passe bien sans aucune interruption ni bug, nous avons apporté des modifications et lancé l'exécution de seed-data grâce à la commande **docker build seed-data_image**. le résultat est tel que vous pouvez voir ci dessous

```

souop@AN-0591p:~$ docker build -t seed-data_image .
[+] Building 0.7s (10/10) FINISHED                                docker:default
=> [internal] load .dockerignore                                0.0s
=> => transferring context: 2B                                   0.0s
=> [internal] load build definition from Dockerfile              0.0s
=> => transferring dockerfile: 212B                              0.0s
=> [internal] load metadata for docker.io/library/httpd:latest  0.6s
=> [1/6] FROM docker.io/library/httpd@sha256:f0a93744d8006e6f7ee5086c9ddccdcfa33d1091f15269a00547b4c382459c1f 0.0s
=> CACHED [2/6] RUN apt update -y                               0.0s
=> CACHED [3/6] RUN apt install -y iputils-ping                 0.0s
=> CACHED [4/6] RUN apt install -y inetutils-traceroute         0.0s
=> CACHED [5/6] RUN apt install -y iproute2                     0.0s
=> CACHED [6/6] RUN apt install -y curl telnet dnsutils vim     0.0s
=> exporting to image                                           0.0s
=> => exporting layers                                           0.0s
=> => writing image sha256:faf07ca4c950a15fa8f42ff5708d92d6bbe3434bc8808f8f15f72de2d3c1903 0.0s
=> => naming to docker.io/library/seed-data_image              0.0s

What's Next?
View a summary of image vulnerabilities and recommendations → docker scout quickview

```

Et pour finir nous avons builder aussi l'image du worker avec la commande **docker build worker_image**. Nous avons obtenu ceci:

```

=> [internal] load metadata for docker.io/library/httpd:latest  2.7s
=> [auth] library/httpd:pull token for registry-1.docker.io    0.0s
=> [1/6] FROM docker.io/library/httpd@sha256:f0a93744d8006e6f7ee5086c9ddccdcfa33d1091f15269a00547b4c382459c1f 3.8s)
=> => resolve docker.io/library/httpd@sha256:f0a93744d8006e6f7ee5086c9ddccdcfa33d1091f15269a00547b4c382459c1f 0.0s
=> => sha256:6fd77d7e5eb732dacab601d4556c04a6c312928fb8989fe3b0a47d82db772441 8.17kB / 8.17kB 0.0s
=> => sha256:eba4da411ea035e6ea857a279fb974af65e44403318f84083f57a7f017261f60 145B / 145B 0.2s
=> => sha256:4f4fb700ef54461cfa02571ae0db9a0dc1e0cdb5577484a6d75e68dc38e8acc1 32B / 32B 0.2s)
=> => sha256:ed4d6291a6c22af972af117a61bed58c01e6dc8a0cd79bffb91ed7e58631dea2 4.20MB / 4.20MB 0.5s
=> => sha256:f0a93744d8006e6f7ee5086c9ddccdcfa33d1091f15269a00547b4c382459c1f 7.48kB / 7.48kB 0.0s
=> => sha256:448ce9c60d2ac0e8f779ce871b06df6452c30dae74f725d5694502da9c902036 1.79kB / 1.79kB 0.0s
=> => extracting sha256:eba4da411ea035e6ea857a279fb974af65e44403318f84083f57a7f017261f60 0.0s
=> => sha256:b42c390e1de91d17ab7cbb2842a1da5a4e2e9e3fb3f8986148eab7e61a87f72d 31.19MB / 31.19MB 1.5s
=> => sha256:eafe388a0bb88eca31dff0e1325a8d0ed7162934dcd62dbabb1b5c97a5823eb1 293B / 293B 0.4s
=> => extracting sha256:4f4fb700ef54461cfa02571ae0db9a0dc1e0cdb5577484a6d75e68dc38e8acc1 0.0s
=> => extracting sha256:ed4d6291a6c22af972af117a61bed58c01e6dc8a0cd79bffb91ed7e58631dea2 0.6s
=> => extracting sha256:b42c390e1de91d17ab7cbb2842a1da5a4e2e9e3fb3f8986148eab7e61a87f72d 1.8s
=> => extracting sha256:eafe388a0bb88eca31dff0e1325a8d0ed7162934dcd62dbabb1b5c97a5823eb1 0.0s
=> [2/6] RUN apt update -y                                     5.3s
=> [3/6] RUN apt install -y iputils-ping                       2.4s
=> [4/6] RUN apt install -y inetutils-traceroute              4.4s
=> [5/6] RUN apt install -y iproute2                          3.6s
=> [6/6] RUN apt install -y curl telnet dnsutils vim          8.7s
=> exporting to image                                           0.4s
=> => exporting layers                                           0.3s
=> => writing image sha256:faf07ca4c950a15fa8f42ff5708d92d6bbe3434bc8808f8f15f72de2d3c1903 0.0s
=> => naming to docker.io/library/worker_image                 0.0s

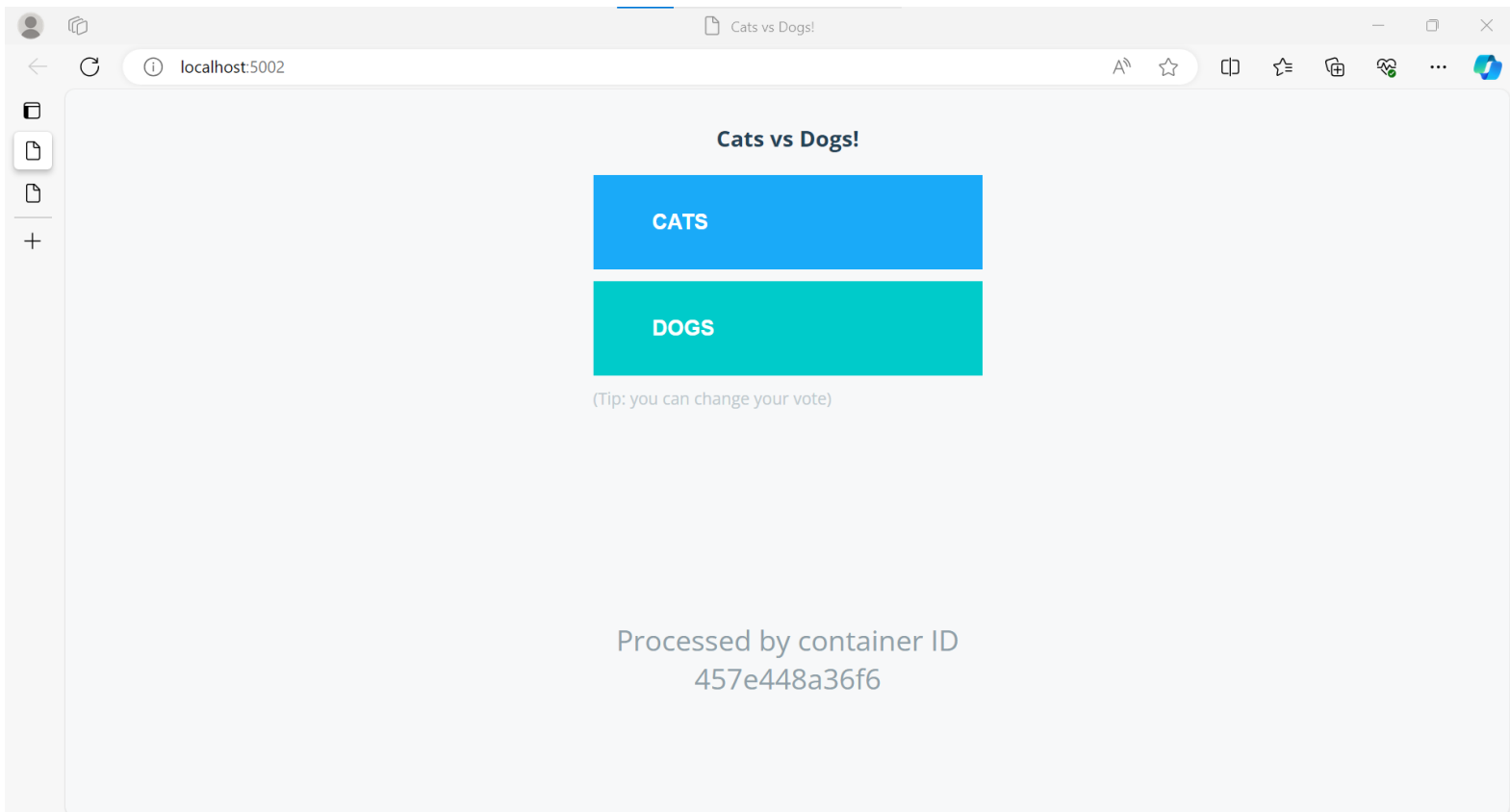
What's Next?
View a summary of image vulnerabilities and recommendations → docker scout quickview
souop@AN-0591p:~$

```

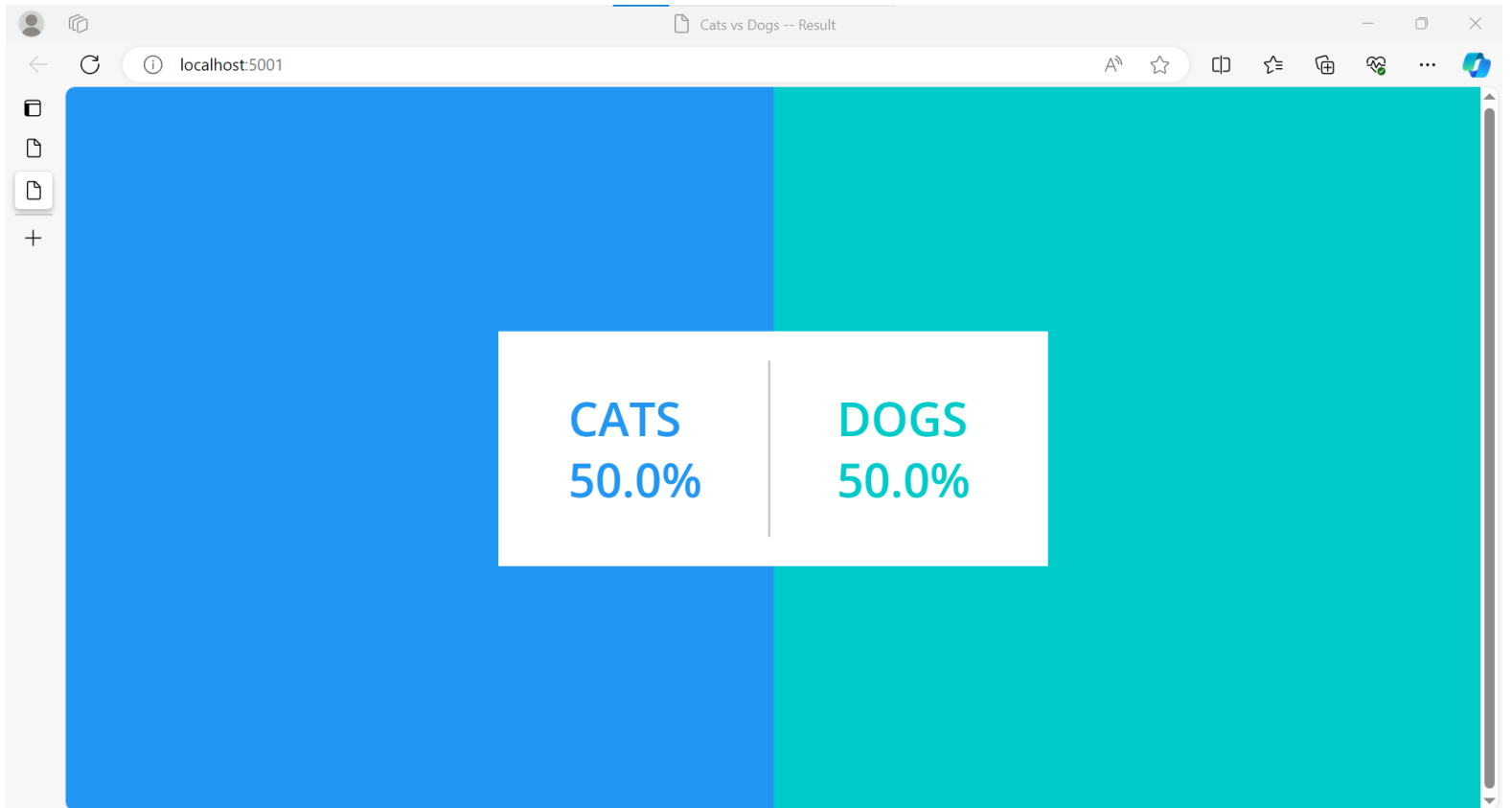
RÉSULTATS

Après toutes ces différentes opérations, nous avons créé le fichier YAML `compose.yml` avec notre commande **`docker-compose up`** et le résultat était concédé dans `compose.yml`

En mettant en marche nos conteneurs, nous avons eu ces différentes sorties. au port 5002 nous avons eu ceci:



En seconde sortie sur le port 5001, nous avons eu ceci comme sortie:



CONCLUSION

L'objectif ici n'était évidemment pas de faire une trop longue présentation sur le Projet de virtualisation, ce qui serait certes très intéressant, mais plutôt de poser les bases techniques et le vocabulaire employé dans ce système .

RÉFÉRENCES

1. <https://kubernetes.io/docs/setup/production-environment/tools/>
2. <https://docs.docker.com/engine/install/ubuntu/>
3. <https://docs.docker.com/compose/gettingstarted/>
4. <https://kubernetes.io/docs/reference/kubectl/cheatsheet/>
5. <https://github.com/JeffSouop/HUMAN-BEST-FRIENDS-APP-SOUOP-4A-49.git>