Rapport de projet « HumansBestFriend »

Groupe TD-49 – Virtualisation & Conteneurisation

Table des matières

Introduction

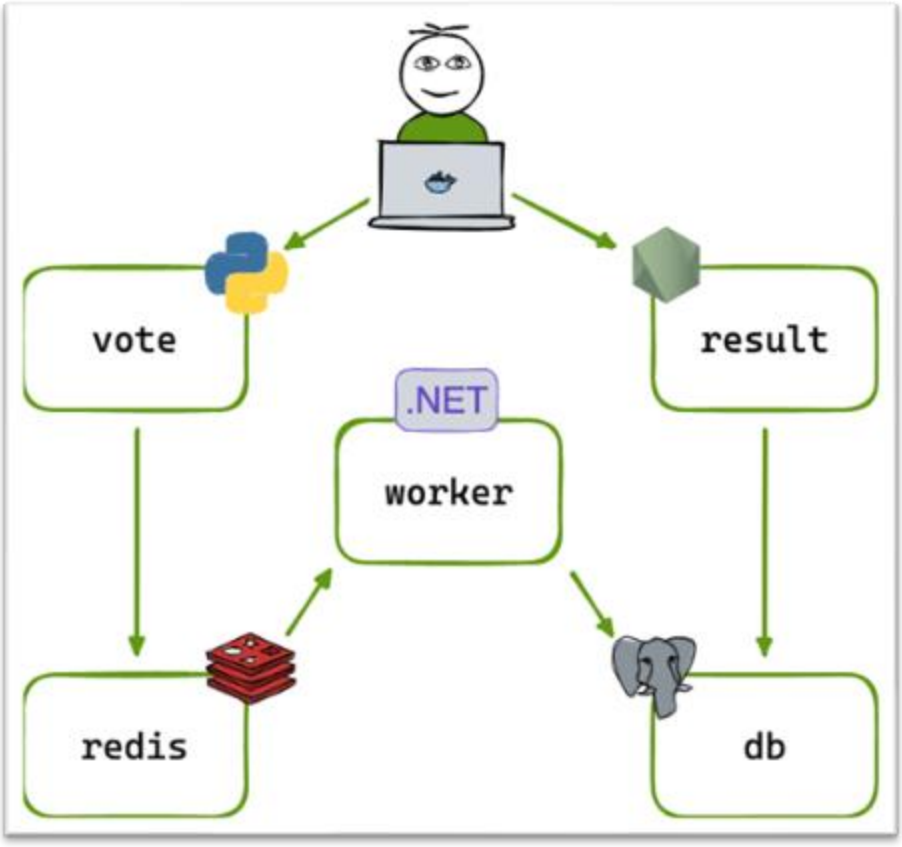
Equipe

- DA CRUZ Benjamin
- DE CASTRO Lucas
- DRAMÉ Arouna
- LATEB Samy

Sujet

Le projet consiste en la création d'une application distribuée simple s'exécutant sur plusieurs conteneurs Docker. L'ensemble du développement doit être réalisé à l'intérieur d'une machine virtuelle exécutant Docker et Docker Compose, conformément à la documentation Docker fournie au cours. Les technologies utilisées incluent Python, Node.js, .NET, Redis pour la messagerie, et Postgres pour le stockage.

Pour la réalisation de ce projet, nous avons créé un fork du projet initial (voir ici) vers notre dépôt Github (voir ici).

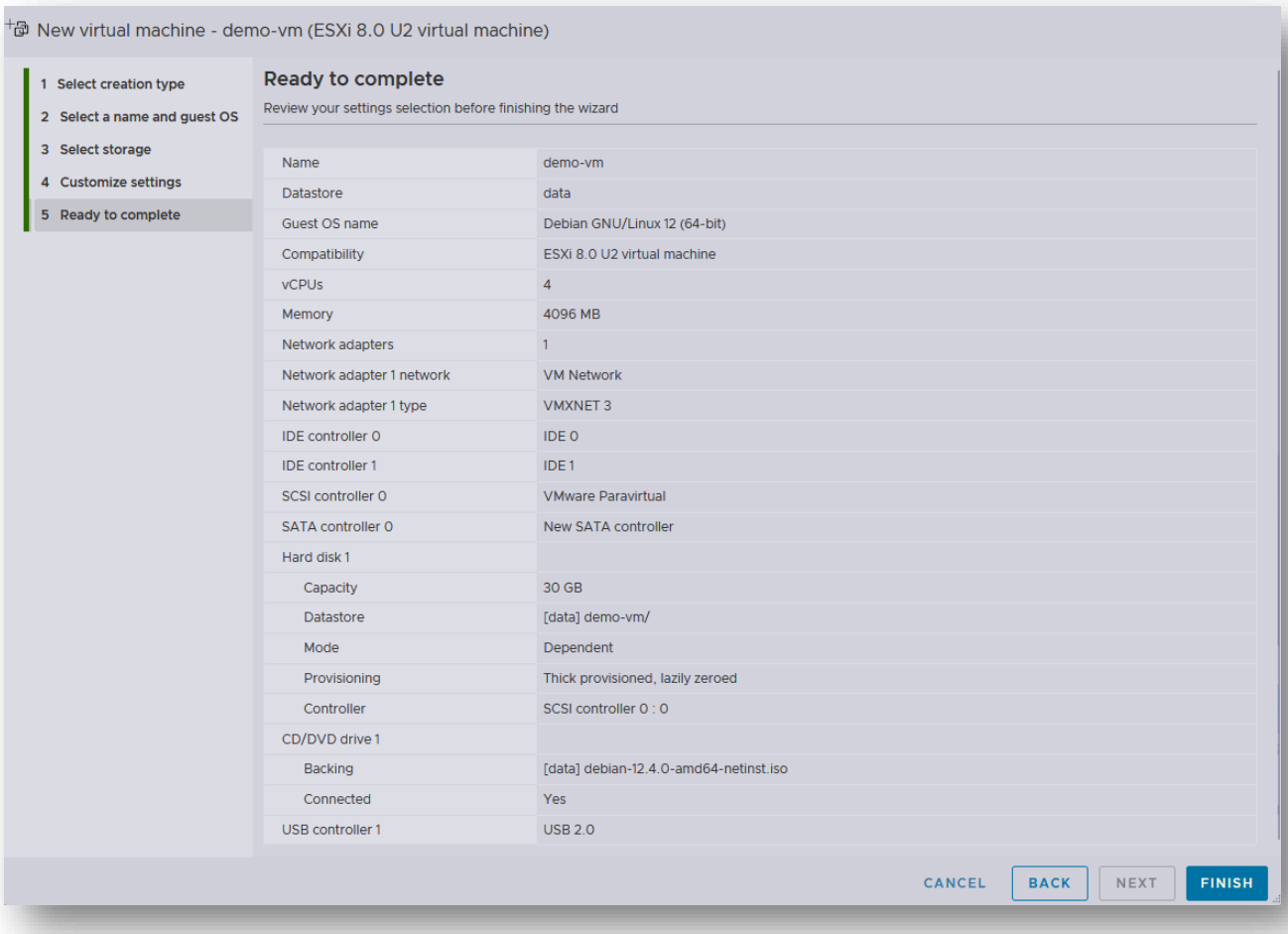


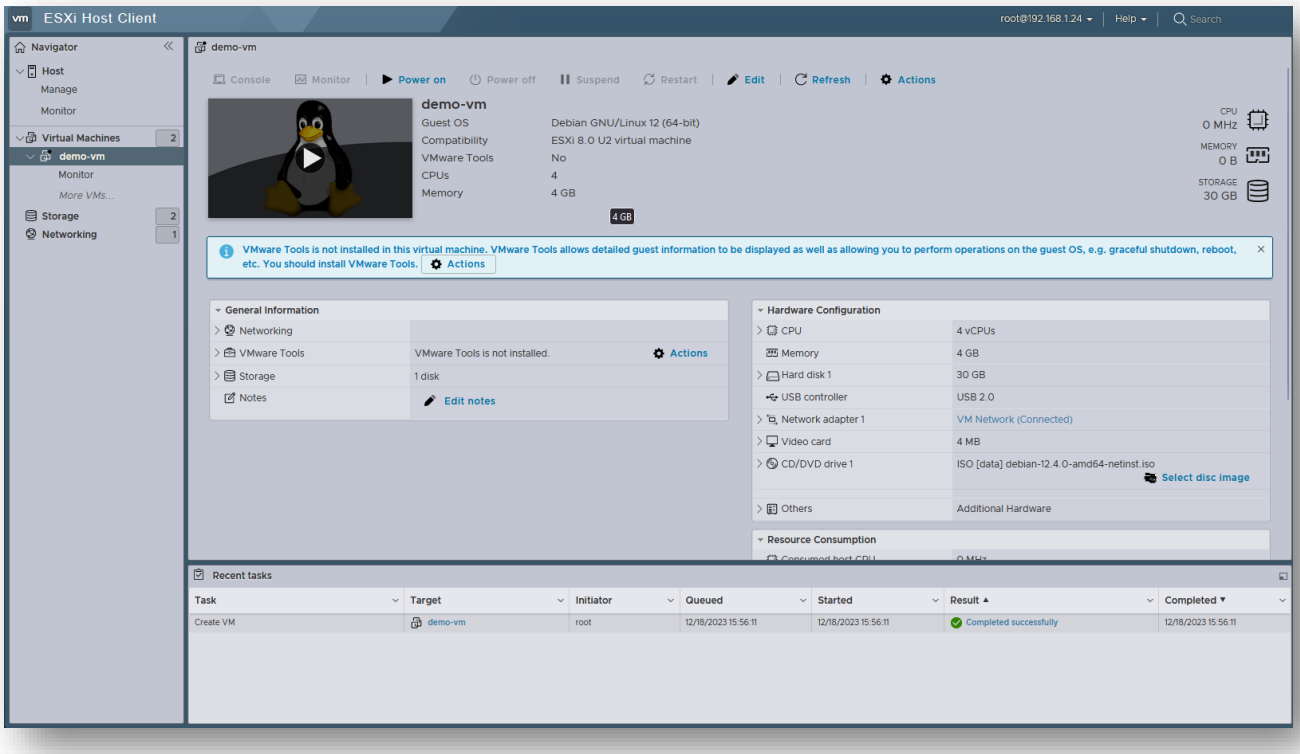
Réalisation

L'intégralité du code source de notre projet se trouve dans notre dépôt publique Github et c'est à partir de celui-ci que toute l'installation se fait dans notre procédure d'installation (cf. SUBMISSION.md), comme spécifié dans le sujet.

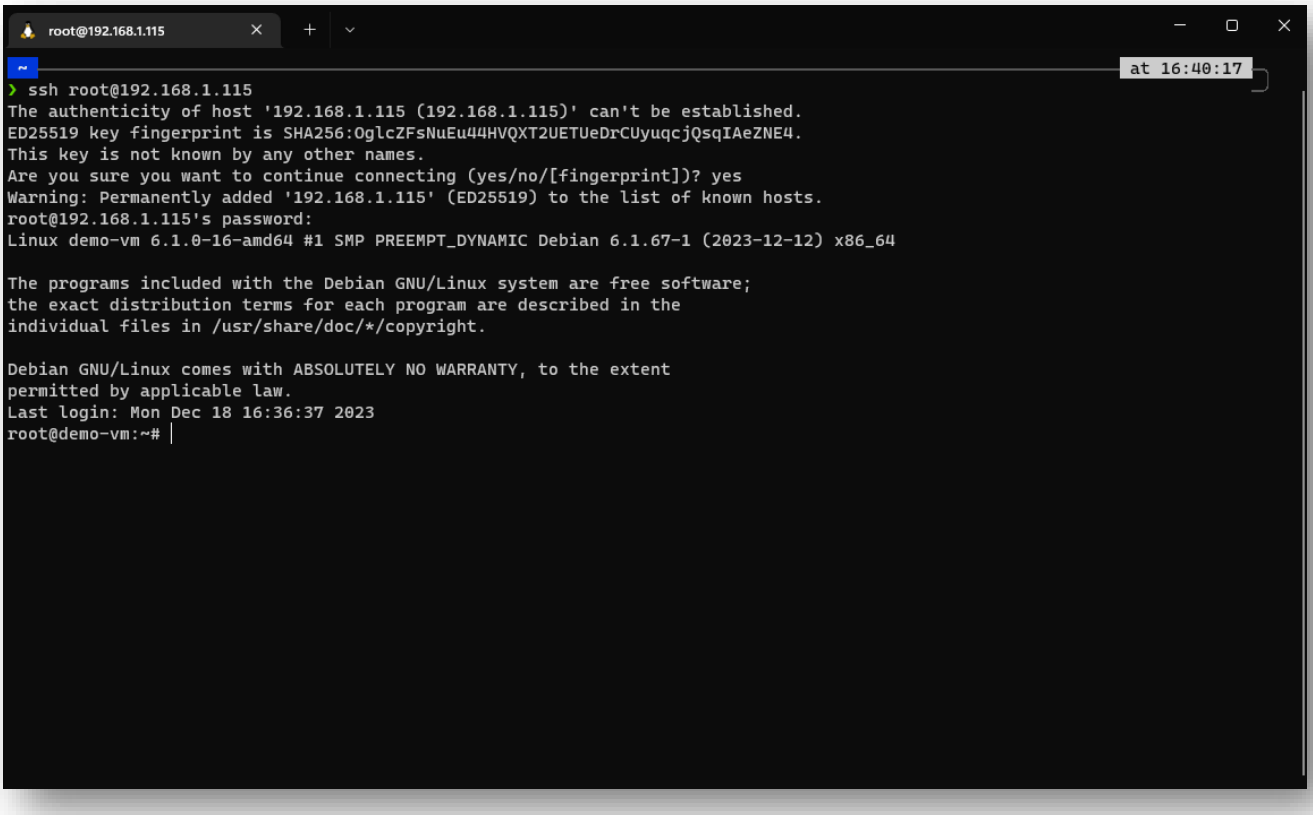
Afin de démontrer le bon fonctionnement de notre projet, nous avons réalisé une démo d'installation et de fonctionnement de notre projet sous ses deux formes : la forme « manuelle » qui n'utilise que des commandes docker et sans utiliser docker compose, et une forme dite « automatisée » qui utilise docker compose.

Tout cela a été réalisé dans une machine virtuelle nommée « demo-vm » installée sur notre instance ESXI.



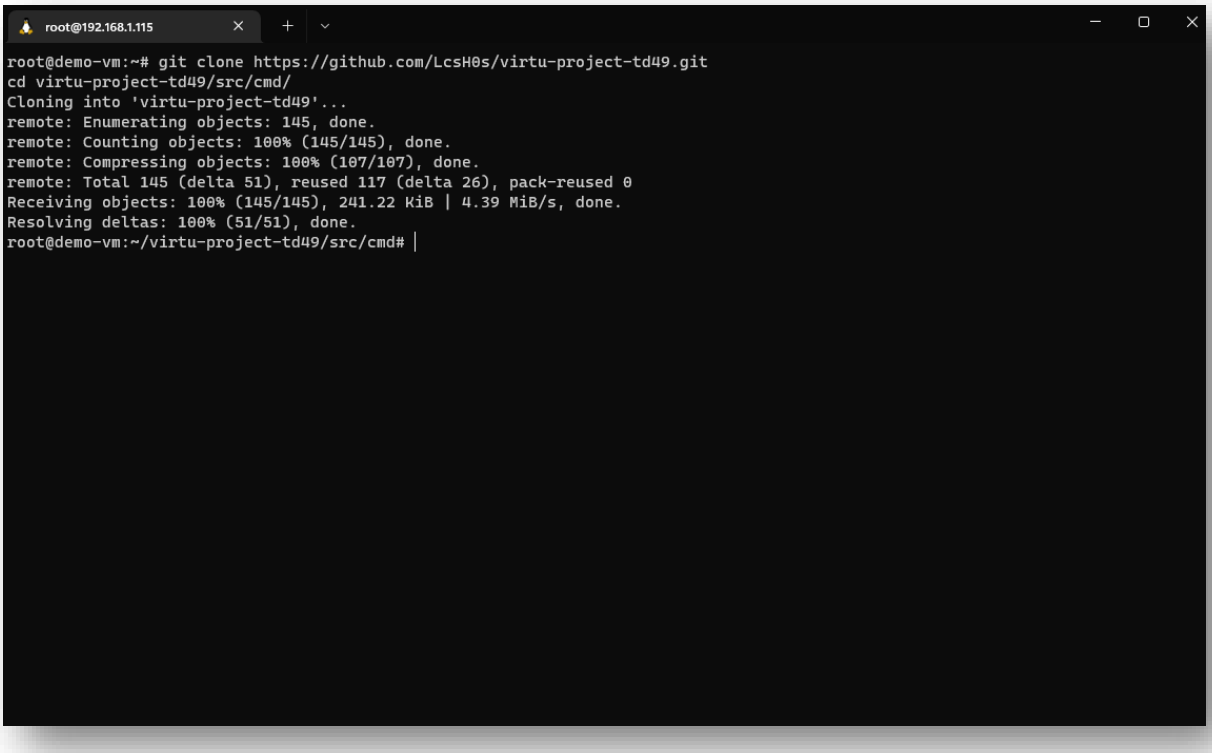


Nous avons ensuite installé un serveur SSH sur notre VM afin de pouvoir nous y connecter depuis notre hôte, et ainsi rendre les étapes suivantes de lancement de l'application beaucoup plus aisées.



Passons maintenant à l'installation de notre application de la première manière, l'installation « manuelle » (uniquement avec des commandes docker et sans docker compose).

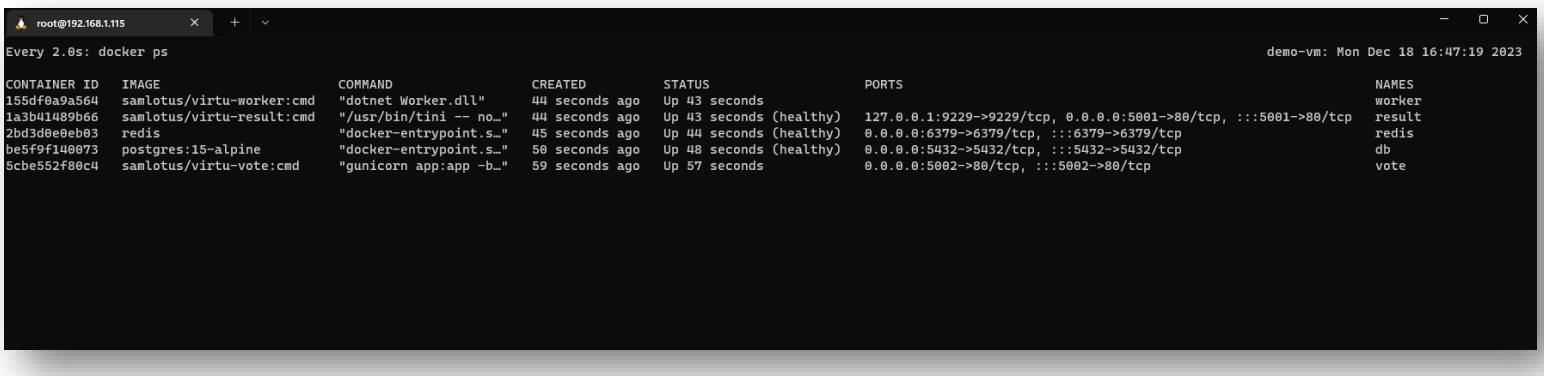
On clone d'abord le dépôt et on entre dedans.



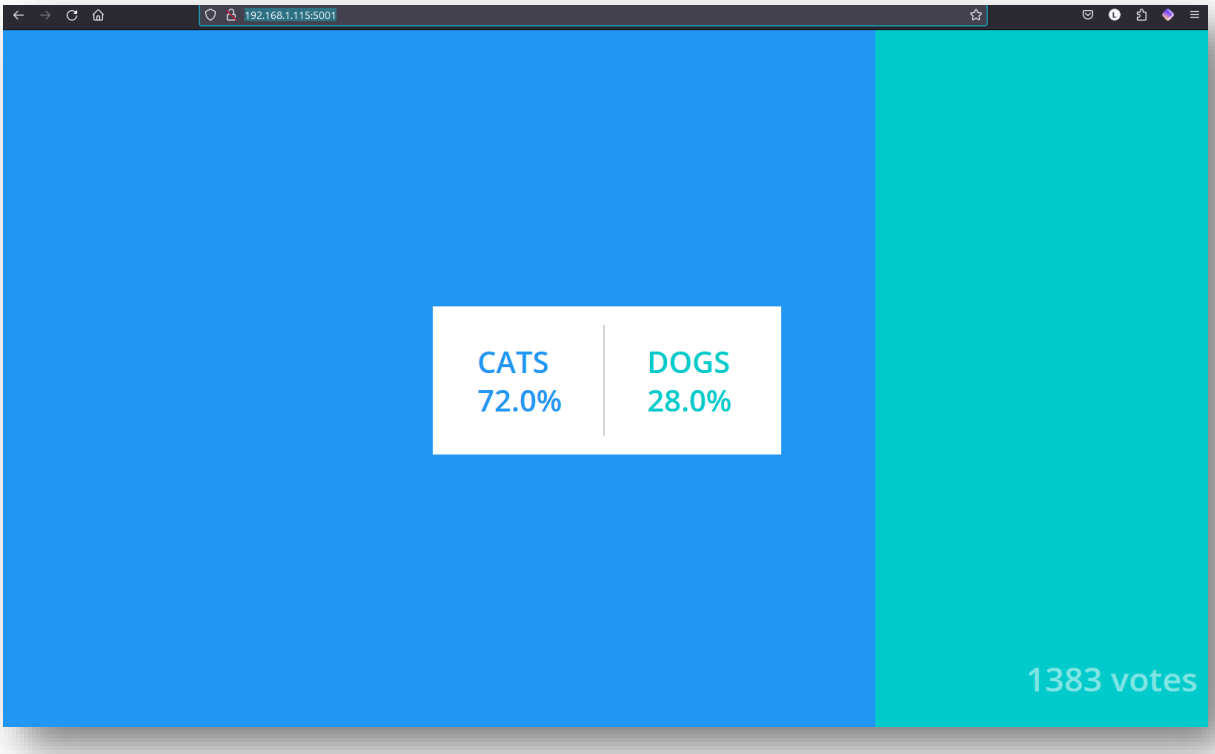
On lance ensuite successivement toutes les commandes docker spécifiées dans la procédure d'installation afin de créer les conteneurs, réseaux et volumes nécessaires.

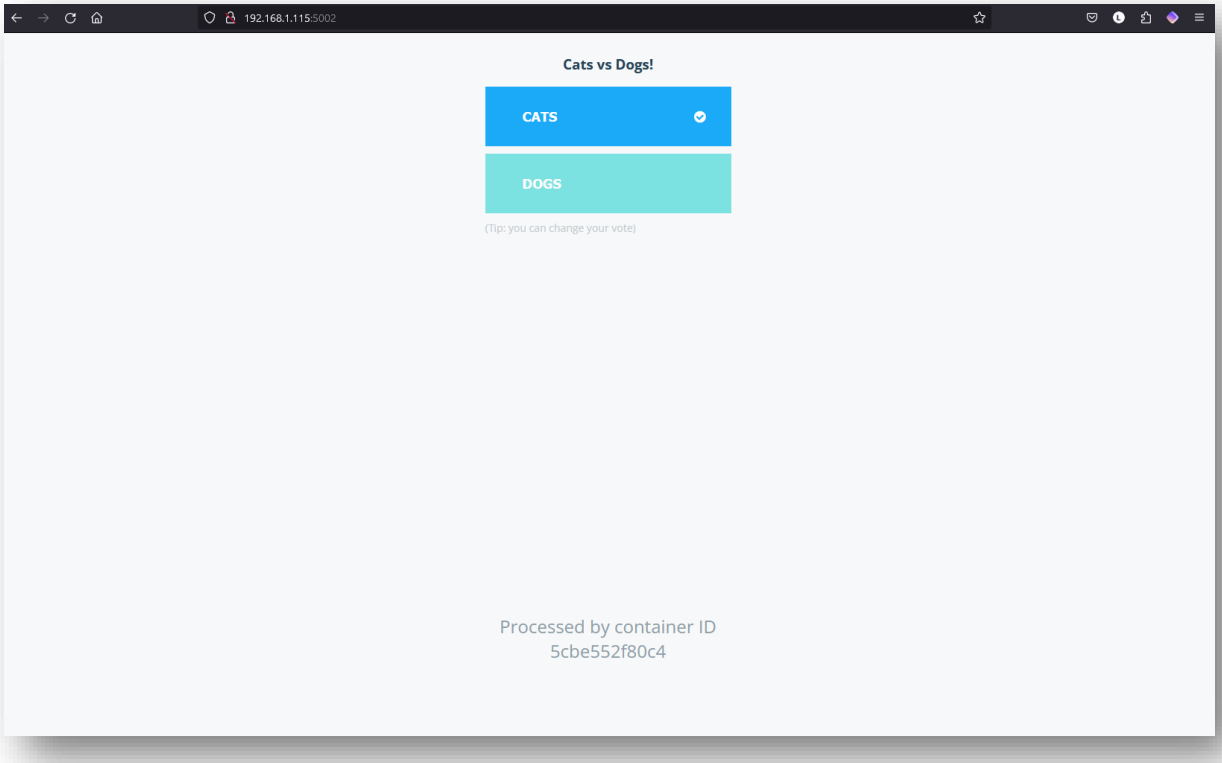
Une version automatisée de ces commandes existe également dans le Makefile de ce même dossier courant. Il a pour simple rôle d'accélérer le déploiement mais il exécute sensiblement les mêmes commandes.

Après cette procédure, nous pouvons vérifier que tous nos conteneurs sont bien lancés grâce à la commande « docker ps ».



Enfin, nous pouvons nous rendre sur l'URL de notre application, ici http://192.168.1.115:5002 pour voter et http://192.168.1.115:5001 pour les résultats du vote. **Cette IP dépend de l'IP de votre hôte docker.**

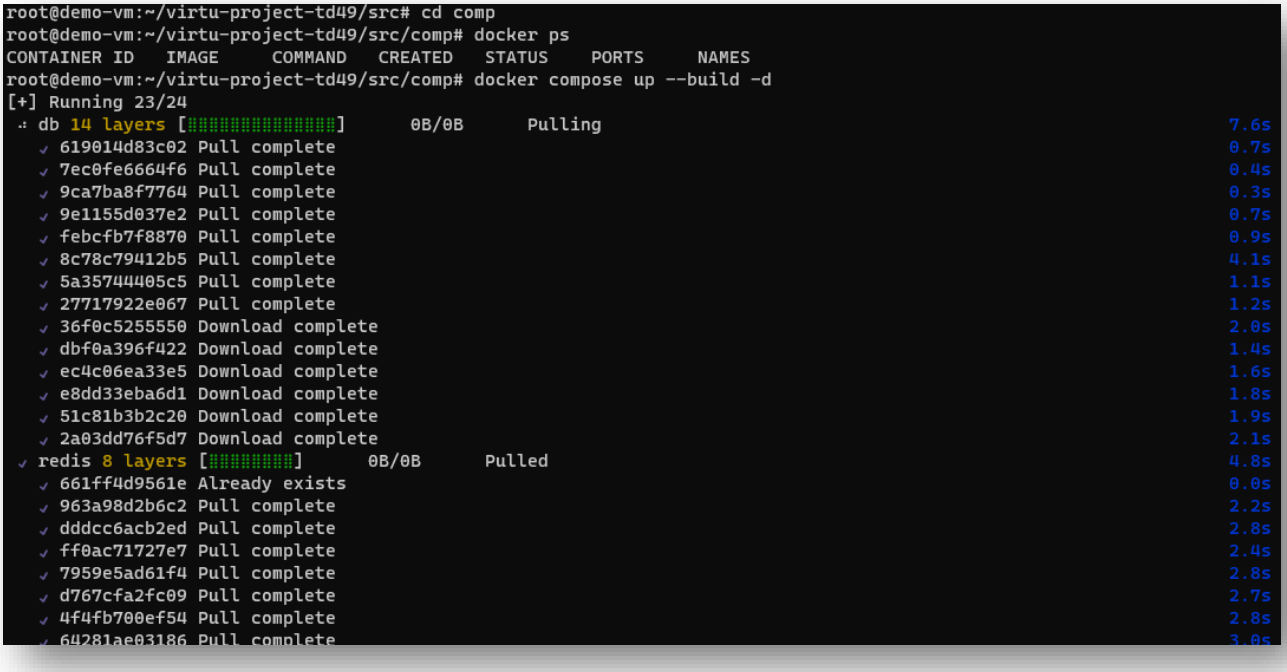




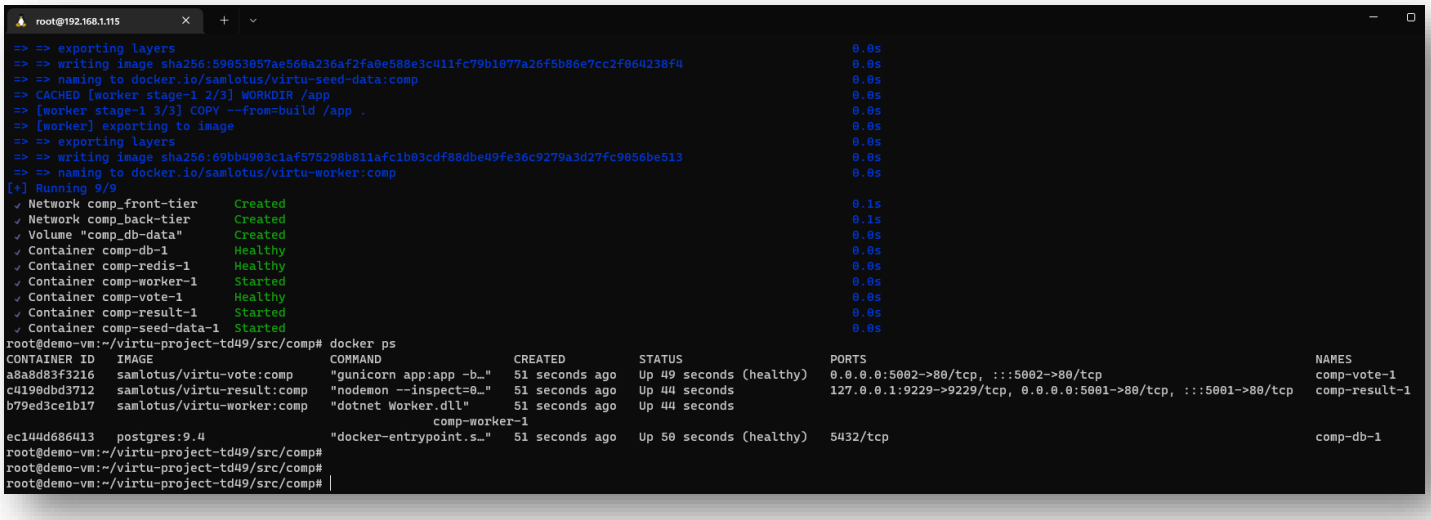
On constate bien que les deux pages fonctionnent correctement et que notre application est fonctionnelle.

Maintenant reproduisons les mêmes étapes avec la version « automatisée » de notre application, c'est-à-dire utilisant docker compose. Cette fois-ci, nous rentrons dans le dossier src/comp qui contient les sources de la version compose.

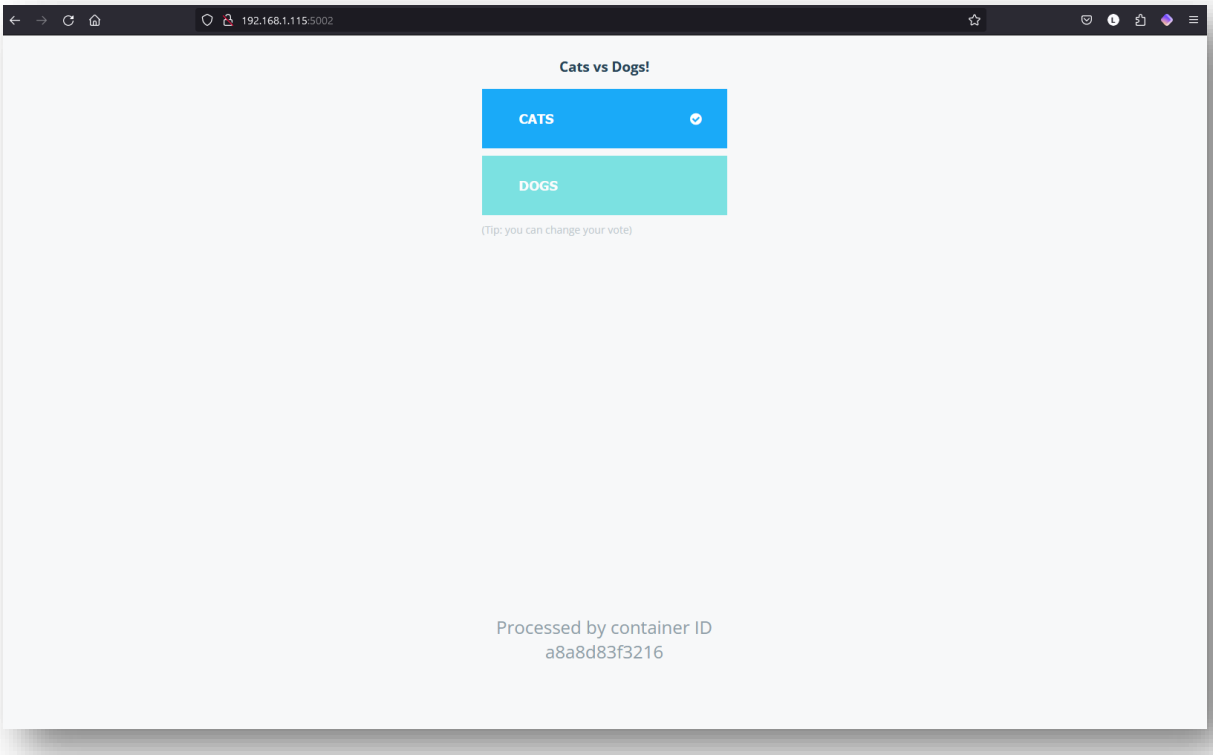
Il nous suffit simplement d'exécuter la commande « docker compose up –build -d » spécifiée dans la procédure de lancement et tout le reste se déroule de manière automatisée.

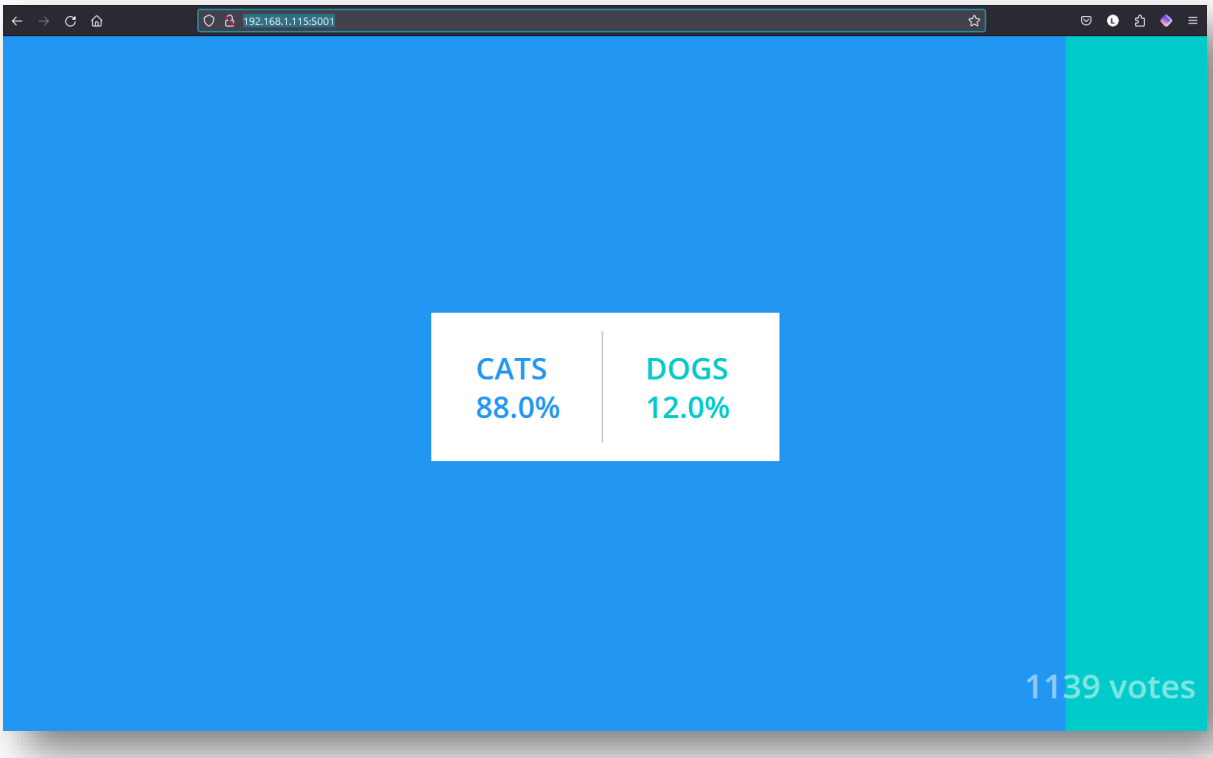


Notre « docker ps » nous affiche bien que tous les conteneurs sont lancés.



Nous pouvons alors vérifier que notre application fonctionne correctement en nous rendant sur les mêmes URLs que précédemment et nous constatons que tout fonctionne correctement.





Docker Hub

Toutes les images customisées que nous avons utilisées dans ce projet peuvent être retrouvées sur Docker Hub (voir ici la liste des images dans notre dépôt Github).

Conclusion

En conclusion, ce projet a été une plongée enrichissante dans le monde complexe et interconnecté des applications distribuées. En intégrant diverses technologies telles que Python, Node.js, .NET, Redis et Postgres, nous avons conçu une architecture complète orchestrée par Docker Compose. L'utilisation de conteneurs a démontré son efficacité dans la gestion des dépendances, la cohérence du déploiement et la facilité de mise à l'échelle.

Le déploiement sur une infrastructure VMware EXSI a mis en lumière l'importance de l'orchestration dans des environnements réels, tout en soulignant la nécessité d'une collaboration d'équipe bien organisée. La documentation claire, les fichiers Dockerfile et docker-compose.yml ont joué un rôle crucial dans la cohérence du processus de développement.

Le recours à Docker Compose a particulièrement marqué les esprits, soulignant son rôle central dans la simplification de la gestion et du déploiement de l'application. Cette expérience renforce notre compréhension des avantages pratiques de l'orchestration de conteneurs dans le développement d'applications distribuées. En somme, ce projet a été une exploration approfondie, combinant théorie et pratique, et nous a permis d'acquérir des compétences précieuses dans la conception et le déploiement d'applications distribuées à grande échelle.