Mini-projet [Sujet]

Ouvert le : mardi 2 décembre 2025, 16:00
À rendre : dimanche 4 janvier 2026, 23:59

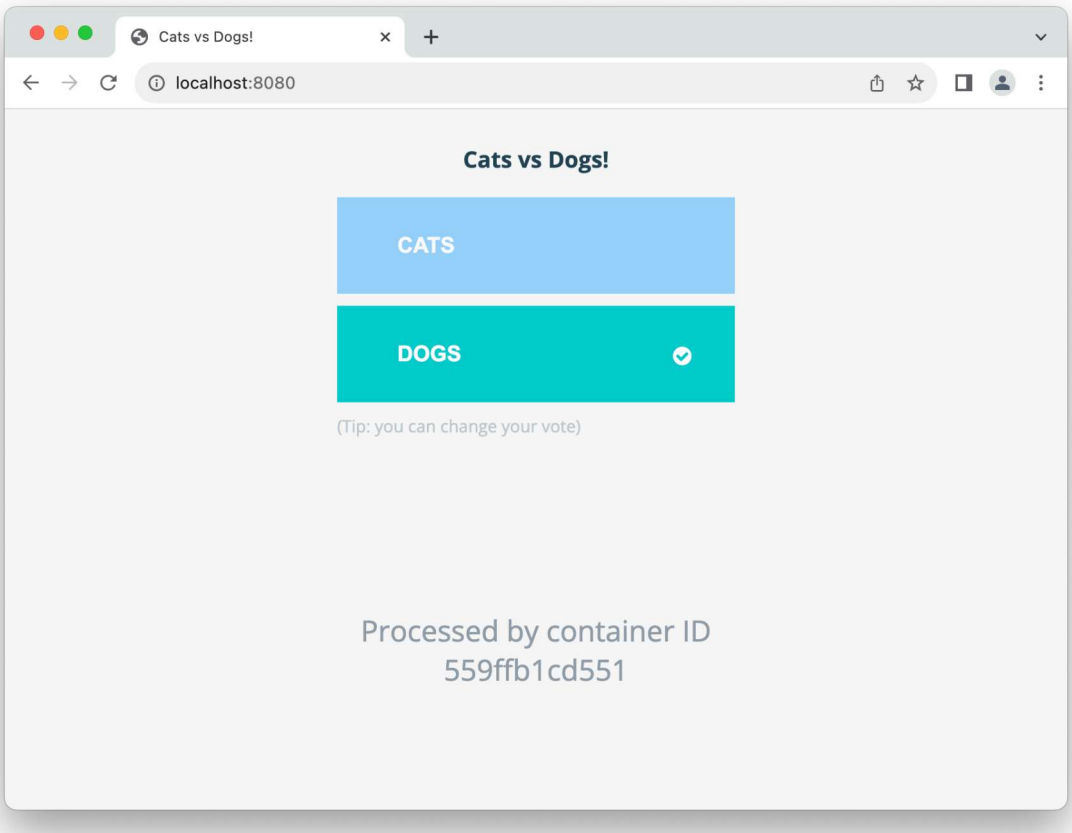
Ce mini-projet est à réaliser par groupes de 2 étudiants au maximum.

Le code source est accessible ici.

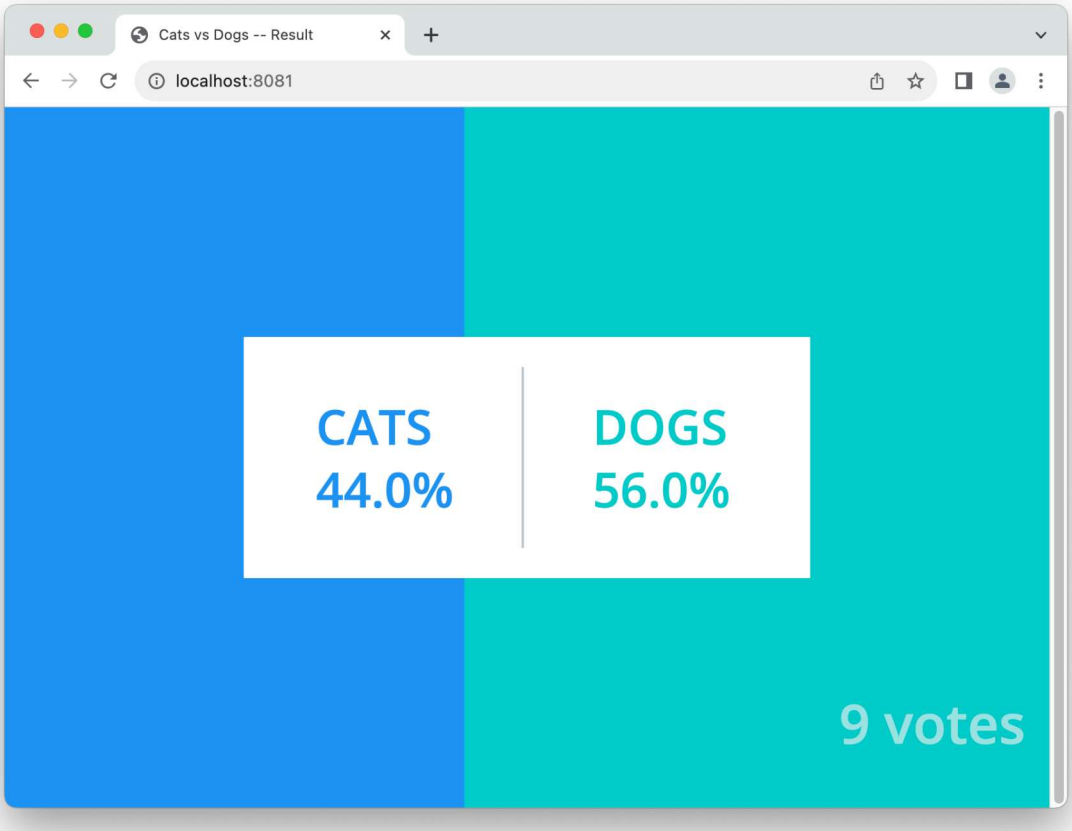
Pour ce mini-projet, on vous donne en pièce-jointe le code source d'une application distribuée permettant à une audience d'effectuer un vote parmi deux propositions.

Celle-ci comporte deux interfaces web :

- La première, permet de voter pour l'une des deux options. On ne peut voter qu'une seule fois par navigateur web, mais il est toujours possible de changer son vote.



- La seconde, permet de visualiser le résultat du vote. La page est mise à jour automatiquement dès qu'un nouveau vote est pris en compte.



Actuellement, cette application est lancée avec plusieurs scripts *bash* présents dans le projet. Votre rôle est de moderniser la manière dont ce projet est déployé au moyen des conteneurs Docker.

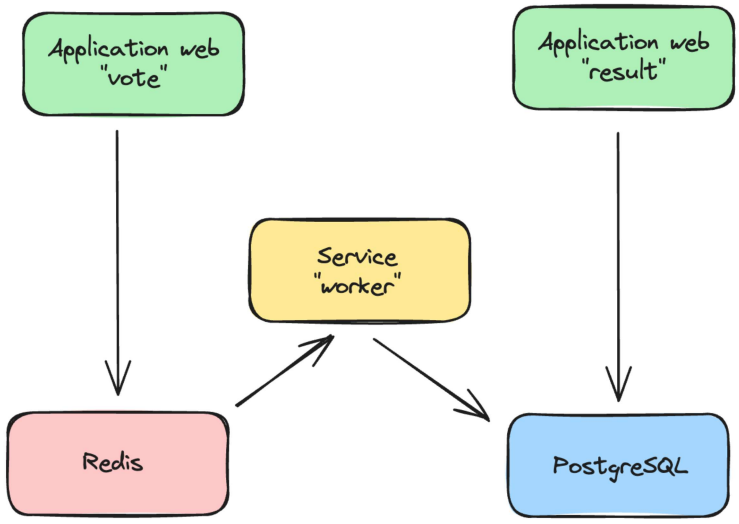
Le projet comporte les modules suivants :

- **vote** : une application web Python qui permet de voter pour l'une des deux options.
- **worker** : un service .NET qui consomme les votes depuis une instance Redis et les stocke dans une base de données PostgreSQL.
- **result** : une application web Node.js qui affiche les résultats du vote en temps réel.

De plus, l'application utilise :

- Une instance de PostgreSQL pour stocker les votes.
- Une instance de Redis pour les transmettre.

Voici un schéma représentant les relations entre ces différents composants :



Les développeurs du projet ont indiqué qu'il était nécessaire d'avoir les environnements de développement suivants :

- Python version 3.11 + pip
- Node.js version 18 + npm
- SDK .NET Core version 7 (aussi disponibles sous Linux et macOS)

Ils ont aussi précisé que le projet fonctionnait "sans doute" avec des versions antérieures ou postérieures, mais qu'il fonctionnait avec "au moins celles-ci".

Pour lancer l'application, des scripts sont disponibles dans le dossier "run". Il est indiqué qu'il faut les exécuter dans l'ordre, de 1 à 5, et chacun dans un terminal séparé. Est également présent un script permettant de supprimer les dossiers et fichiers temporaires. Les développeurs ont commencé à utiliser Docker dans les scripts pour lancer les instances de PostgreSQL et Redis, mais ils ont communiqué ne pas maîtriser l'outil au-delà d'un "docker run".

Votre processus de conteneurisation de l'application se déroulera en 4 étapes :

1) Écrire un Dockerfile pour chaque module en reproduisant le comportement des scripts. Ils devront respecter toutes les bonnes pratiques que vous connaissez.

2) Supprimer les scripts (ou faire en sorte qu'ils utilisent Docker).

3) Écrire un fichier Docker Compose regroupant les conteneurs et bases de données du projet. On y déclarera les volumes nécessaires pour ne pas perdre les données au redémarrage du projet, ainsi que tous les réseaux que vous jugerez pertinents. Par ailleurs, on veillera à gérer les dépendances et les cas où les conteneurs ne démarrent pas.

4) Déployer votre application sur un cluster Docker Swarm. Pour des raisons de commodité, ce cluster n'est pas à rendre mais vous devrez rédiger un document qui décrit votre processus de mise en place de ce cluster (ainsi que le déploiement de votre application). Votre cluster devra comporter 1 nœud manager et 2 nœuds worker.

Vous êtes autorisés à changer tout ce que vous trouverez pertinent dans le code source, à condition que les changements soient répertoriés et ne modifient pas les fonctionnalités principales. Pour cette raison, vous êtes invités à versionner le code source sous Git. Il faudra notamment effectuer des changements d'hôte et de port pour Docker Compose. On veillera à garder des commentaires dans l'historique intelligibles.

Livrable

Un fichier .zip comportant (aucun autre format ne pourra être reçu) :

- Le code source avec toutes vos modifications.
- Un document texte dans le format de votre choix (.md, .docx, ou .pdf) reprenant toutes les informations que vous jugerez indispensables pour le correcteurs. Cela comprend entre le processus pour lancer le projet et les étapes nécessaires pour bâtir votre cluster Docker Swarm.

Toutes les ressources sont permises, mais on ne trompera en aucun cas le correcteur avec du code qui n'est pas le vôtre.

Barème (40 points)

- Le module *vote* est conteneurisé et respecte au mieux les bonnes pratiques : **3 points**
- Le module *worker* est conteneurisé et respecte au mieux les bonnes pratiques : **3 points**
- Le module *result* est conteneurisé et respecte au mieux les bonnes pratiques : **3 points**
- Tous les conteneurs sont regroupés dans un fichier Docker Compose complet : **7 points**
- Le fichier Docker Compose définit les dépendances entre les composants : **2 points**
- Le fichier Docker Compose fait en sorte de sonder le statut des conteneurs : **2 points**
- Les données de l'application ne sont pas perdues au redémarrage de celle-ci : **4 points**
- Les composants du projet sont isolés dans des réseaux appropriés : **4 points**
- Un processus de déploiement sur un cluster Docker Swarm est fourni dans le rendu : **6 points**
- Le fichier Docker Compose a été dupliqué puis adapté pour Docker Swarm : **2 points**
- Les deux applications web restent accessibles même si un des nœuds Docker Swarm quitte le cluster : **2 points**
- Les instructions fournies avec le code source modifié sont claires et concises : **2 points**

Ajouter un travail

Statut de remise

Statut des travaux remis	Aucun devoir n'a encore été remis
Statut de l'évaluation	Non évalué
Temps restant	6 jours 10 heures restants
Dernière modification	-
Commentaires	Commentaires (0)