Projet Rattrapage Architecture As Service

Who Run The World app? CATs or DOGs ?

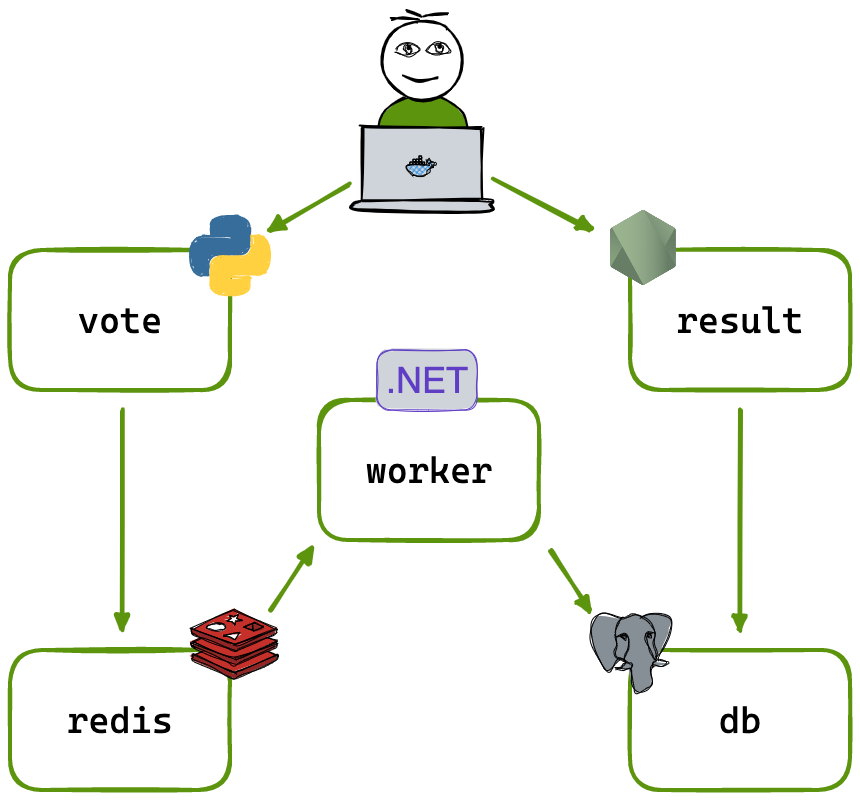
Dans le contexte actuel, nous devons mettre en place une application distribuée sur plusieurs conteneurs Docker.

La solution que nous allons mettre en place comporte 5 services différents :

* Une application web frontale en python qui permet de voter entre deux options
* Un Redis qui récolte de nouveaux votes
* Un Worker .net qui consomme les votes et les stocke dans une base de données.
* Une base de données postgres appuyée par un volume docker
* Une Application web Node.js qui affiche les résultats du vote en temps réel

Notre infrastructure se compose de 3 VM sous Ubuntu Server : 1 Vm nommé Lacinga-master et 2 VM nommé Lacinga-worker1 et Lacinga-worker2.

Ces 3 VM appartiennent à un cluster Kubernetes qui a été déployé préalablement avant le projet.



Build des images Docker avec les fichiers dockerfile fournis :

Pour chacun des services, nous avons des fichiers dockerfile qui nous ont été fournis avec l’énoncé, chacun de ces dockerfile dans chaque service permet de faire fonctionner le logiciel service par service.

J’ai mis en place un fichier docker-compose.yml qui me permets de build chacune de mes images en allant chercher les dockerfiles pour chacun de mes 5 services.

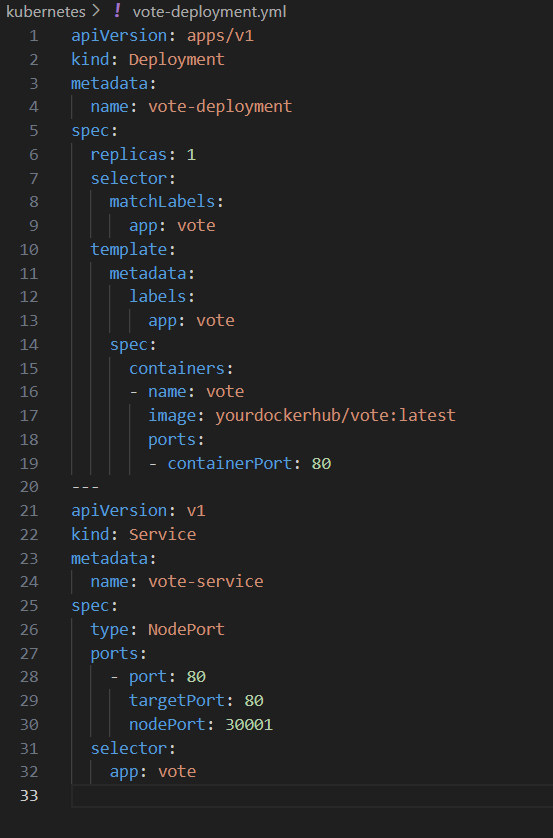
Donc avec la commande docker build -t « nom de l’image », cela va me créer mes images pour chacun de mes services, puis avec la commande docker run « nom de l’image » je vais pouvoir lancer mes différents conteneurs pour chacun de mes services.

Nous nous retrouvons donc avec 5 conteneurs contenant chacun des services et leur permettant de fonctionner indépendamment.

Pour Kubernetes, nous allons créer nos manifestes pour chaque service, le but étant de créer nos images pour chacun des services puis de les déployer sur notre cluster Kube.

**Une application web frontale en python qui permet de voter entre deux options :**

Un fichier vote-deployment.yaml est créé avec la commande vi



Cela va me permettre de créer mon image pour le service Vote et de déployer le conteneur

Avec la commande kubectl apply -f vote-deployment.yaml, cela crée et lance le déploiement de notre conteneur, et ce fichier sera modifiable sans besoin de le redéployer.

Comme demandé dans les tâches, je déploie un node port pour ce service vu que c’est un service qui a besoin d’être exposé au public. Ce service est exposé sur le port 30001.

**Un Redis qui récolte de nouveau votes :**

Un fichier redis-deployment.yaml est créé avec la commande vi



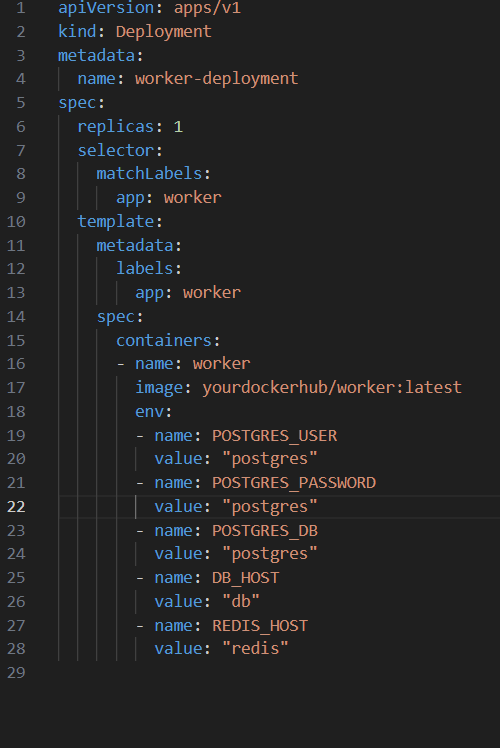
Cela va me permettre de créer mon image pour le service Redis et de déployer le conteneur

Avec la commande kubectl apply -f redis-deployment.yaml, cela crée et lance le déploiement de notre conteneur, et ce fichier sera modifiable sans besoin de le redéployer.

Comme demandé dans les tâches, je déploie un cluster IP pour ce service vu que c’est un service qui n’a pas besoin d’être exposé au public.

**Un Worker .net qui consomme les votes et les stocke dans une base de données :**

Un fichier worker-deployment.yaml est créé avec la commande vi

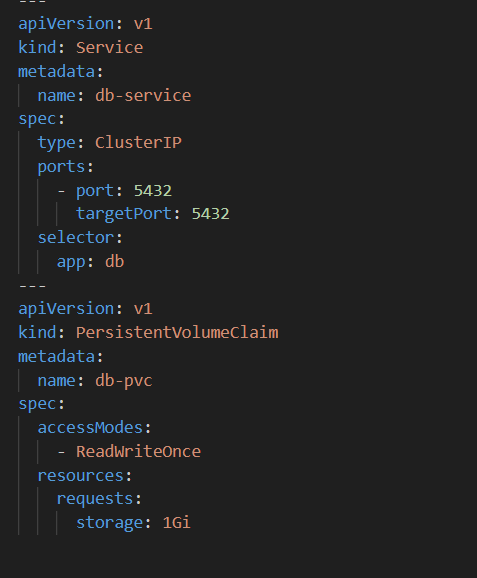
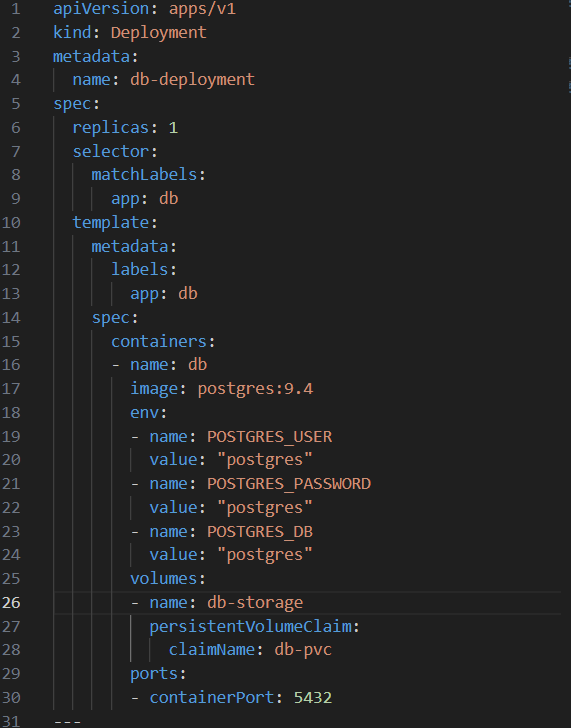


Cela va me permettre de créer mon image pour le service worker et de déployer le conteneur

Avec la commande kubectl apply -f worker-deployment.yaml, cela crée et lance le déploiement de notre conteneur, et ce fichier sera modifiable sans besoin de le redéployer.

**Une base de données postgres appuyée par un volume docker :**

Un fichier db-deployment.yaml est créé avec la commande vi



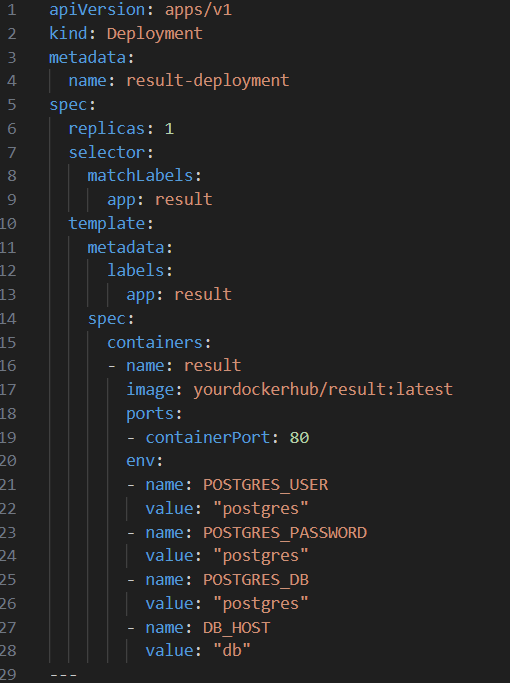
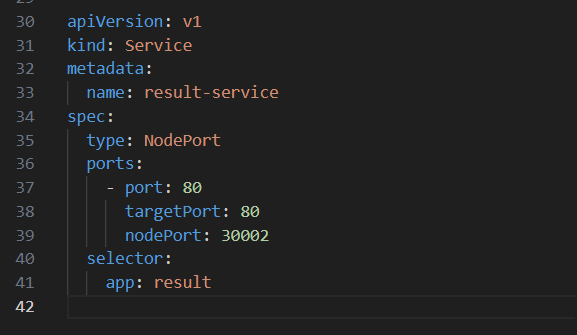
Cela va me permettre de créer mon image pour le service Base de données et de déployer le conteneur

Avec la commande kubectl apply -f db-deployment.yaml, cela crée et lance le déploiement de notre conteneur, et ce fichier sera modifiable sans besoin de le redéployer.

Comme demandé dans les tâches, je déploie un cluster IP pour ce service vu que c’est un service qui n’a pas besoin d’être exposé au public.

**Une Application web Node.js qui affiche les résultats du vote en temps réel :**

Un fichier result-deployment.yaml est créé avec la commande vi



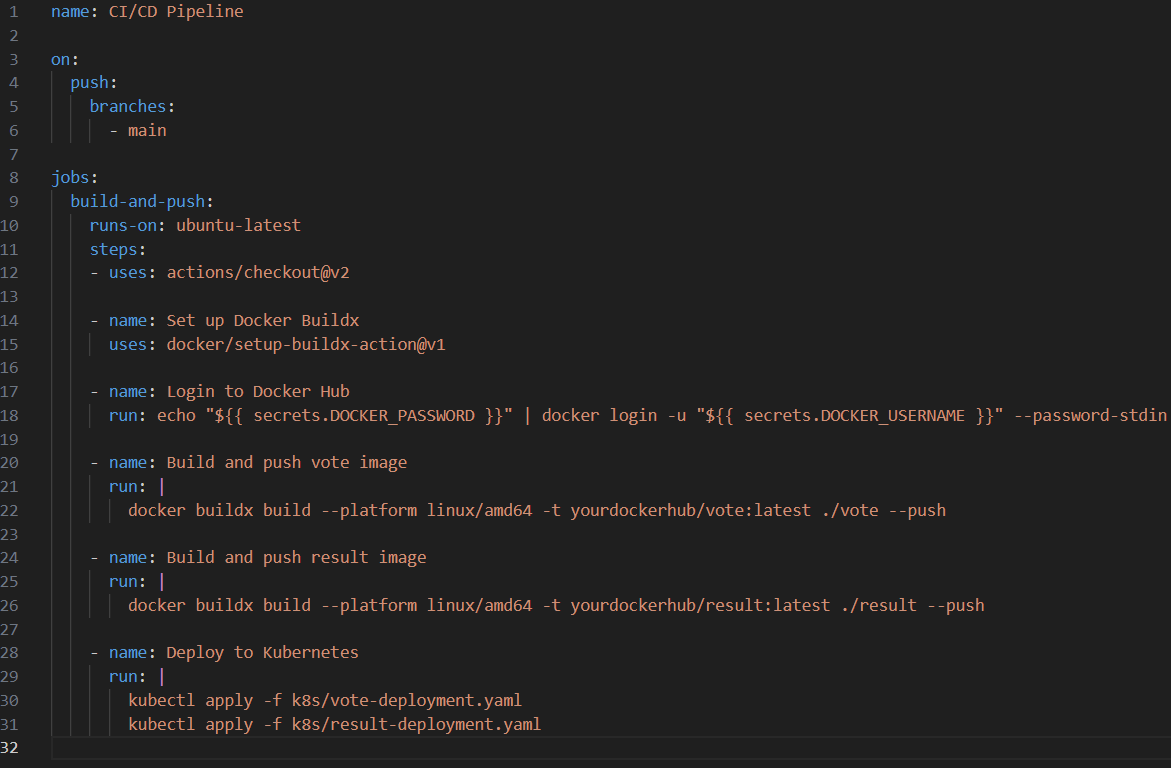
Cela va me permettre de créer mon image pour le service Redis et de déployer le conteneur

Avec la commande kubectl apply -f result-deployment.yaml, cela crée et lance le déploiement de notre conteneur, et ce fichier sera modifiable sans besoin de le redéployer.

Comme demandé dans les tâches, je déploie un node port pour ce service vu que c’est un service qui a besoin d’être exposé au public. Ce service est exposé sur le port 30002.

Gitlab Yaml :

Ce workflow est conçu pour être déclenché à chaque push sur la branche “main”. Le workflow comporte trois jobs, chacun responsable de la construction d'une partie spécifique de l’application.

****