Rapport de Projet: CRV

Kerrian AZIZA-Luka BALY

M1-S2 Réseaux/2024-2025

Table des matières

0.1 Introduction. 3

1 Redis **3**
1.1 Fichiers de déploiements redis- 3
1.1.1 redis-service.yml 3
1.1.2 redis-deployment.yml 3
1.1.3 redis-hpa.yml 3
1.1.4 redis-pv.yml/redis-pvc.yml 3

2 Nodejs **4**
2.1 Dockerfile 4
2.2 Fichiers de déploiements NodeJS 4
2.2.1 backend-service.yml 4
2.2.2 backend-deployment.yml 4
2.2.3 backend-hpa.yml 4

3 React **4**
3.1 Fichiers de déploiements React 5
3.1.1 redis-node-frontend-service.yml 5
3.1.2 redis-node-frontend-deployment 5
3.1.3 redis-node-frontend-hpa.yml 5

4 Prometheus/Grafana **5**
4.1 Fichiers de déploiements Prometheus/Grafana 5
4.1.1 prometheus-service.yml 5
4.1.2 prometheus-deployment.yml 5
4.1.3 prometheus-config.yml 5

5 Deploy.sh **6**

6 Shutdown.sh **6**

7 Auteurs **6**

0.1 Introduction.

LienGitHUB :

Ce projet de CRV consistait à créer une infrastructure Kubernetes permettant la mise à l'échelle automatique pour tous ses composants ainsi que l'implémentation de Prometheus et Grafana. Il est constitué de 3 dossiers ainsi que de deux scripts d'automatisation et d'un README pour expliciter le lancement du projet.

1 Redis

Redis est uniquement récupéré au début du script deploy.sh avec un pull redis et un run pour l'avoir installé directement sur la machine hôte.

1.1 Fichiers de déploiements redis-

Pour ce qui est des fichiers de déploiements concernant Redis, ils sont dans le dossier deployments. Il y a 5 fichiers concernant Redis : redis-hpa.yml, redis-pv.yml, redis-pvc.yml, redis-service.yml et redis-deployment.yml.

1.1.1 redis-service.yml

Fichier relativement simple permettant d'obtenir le service redis, on indique que son port usuel est 6379 ainsi que le protocole qui est TCP. Il y a aussi une partie encore inutilisée mais présente pour le scrapping de données pour la partie 2 avec redis-exporter. Redis-exporter est correctement lancé mais il n'est pas utilisé dans cette partie du projet. Les deux services sont en LoadBalancer permettant de leur donner une adresse IP au besoin.

1.1.2 redis-deployment.yml

Ce fichier contient tous les déploiements concernant Redis. Il permet de déployer redis-master, redis-replica qui contient un réplica de redis-master. Il précise aussi les volumes concernés par ces déploiements ainsi que le déploiement de redis-exporter. Il sera utilisé pour la partie 2 du projet.

1.1.3 redis-hpa.yml

Ce fichier permet la mise à l'échelle de Redis par une mise à l'échelle horizontale concernant le CPU. Si un réplica utilise plus de 50% du CPU, il créera un réplica supplémentaire.

1.1.4 redis-pv.yml/redis-pvc.yml

Ces deux fichiers permettent d'obtenir un volume persistant distinct pour le redis-master et les redis-replicas. Les données sont stockées dans deux dossiers créés dans le path /data/redis-replica et /redis/redis-master. Pour accéder à

ces volumes, les pvc sont créées avec un pvc pour le redis-master et un pour le redis-replica. Ils ont une taille de 1Gi soit environ 1Go

2 Nodejs

L'image nodejs est créée avec un Dockerfile présent dans le dossier /redis-node. Il est lancé à la fois par le script d'automatisation mais l'image utilisée dans les fichiers est récupérée sur le Docker hub. Il s'agit de la même image.

2.1 Dockerfile

Simple fichier permettant de créer une image nodeJS en exposant le port 3000 avec npm, il utilise le package.json fourni dans le dossier de GitHub.

2.2 Fichiers de déploiements NodeJS

Pour ce qui est des fichiers de déploiements NodeJS, ils sont aussi dans le dossier deployments. Les 3 fichiers en rapport avec NodeJS sont nommés backend-deployment.yml, backend-hpa.yml et backend-service.yml

2.2.1 backend-service.yml

Ce fichier permet simplement de lancer le service NodeJS sur le port 8080 avec le type LoadBalancer.

2.2.2 backend-deployment.yml

Ce fichier permet le déployment de NodeJS en pullant l'image sur le Docker HUB : keker03/nodejs :latest. Pour lier redis et NodeJS, on utilise l'adresse IP de redis dans le cluster Kubernetes

2.2.3 backend-hpa.yml

Ce fichier similaire au redis-hpa.yml permet la mise à l'échelle automatique basée sur l'utilisation du CPU. Si elle dépasse 50%, un réplica sera créé.

3 React

L'image docker pour React est créée à l'aide du Dockerfile présent dans le dossier redis-react. Elle utilise nginx pour hoster un serveur web. L'image est créée au lancement du script mais n'est pas utilisée directement, l'image est stockée dans un Docker HUB et c'est cette image que les déploiements utilisent.

3.1 Fichiers de déploiements React

Les fichiers concernant React sont au nombre de 3 : redis-node-frontend-deployment.yml, redis-node-frontend-hpa.yml

3.1.1 redis-node-frontend-service.yml

Ce fichier permet simplement d'avoir le service node-redis-front-service. Il est aussi de type LoadBalancer

3.1.2 redis-node-frontend-deployment

Ce fichier est le fichier de déploiement, il crée deux réplicas du frontend. L'image est récupérée du DockerHUB : keker03/react :latest.

3.1.3 redis-node-frontend-hpa.yml

Ce fichier permet la mise à l'échelle automatique de React, elle fonctionne de manière similaire aux autre fichiers hpa.yml

4 Prometheus/Grafana

Prometheus et Grafana sont présents dans le cluster à l'aide de différents fichiers, ils sont en partie configurés mais ne sont pas opérationnels comme ils ne sont pas compris dans la partie 1 du projet.

4.1 Fichiers de déploiements Prometheus/Grafana

Les fichiers concernants Prometheus et Grafana sont au nombre de 3 : prometheus-config.yml, prometheus-deployment.yml et prometheus-service.yml

4.1.1 prometheus-service.yml

Ce fichier permet de lancer les deux services Prometheus et Grafana, chacun sur leur port respectif, tous deux LoadBalancer.

4.1.2 prometheus-deployment.yml

Le fichier permet le déploiement de Prometheus et Grafana, Prometheus a une esquisse de volume mais n'est pas totalement configuré.

4.1.3 prometheus-config.yml

Ce fichier n'est pas encore utilisé mais est présent, il sera utile pour le scrapping des données des différents services du cluster.

5 Deploy.sh

Ce script permet dans l'ordre de récupérer redis sur la machine et de le lancer, de lancer minikube en mode docker. Applique tous les fichiers présents dans le dossier /deployment puis lance les pods.

6 Shutdown.sh

Ce script permet l'arrêt du projet. Il permet de supprimer les pods en limitant la taille maximale à 0. Il supprime ensuite tous les pods, les déploiements, les services, les volumes claims et les volumes. Il éteint Minikube, supprime toutes les images docker, puis nettoie Minikube.

7 Auteurs

Kerrian AZIZA - Luka BALY