Projet CRV - Infrastructure as Code (IAC)

Déploiement d'une application cloud-native avec Kubernetes

Projet réalisé dans le cadre du cours CRV

Table des matières

1. Présentation du projet

Ce projet met en œuvre une architecture **cloud-native** complète composée des services suivants :

— **Redis** : base de données clé/valeur avec réplication (master + replicas)
— **Node.js** : backend stateless accédant à Redis
— **React** : frontend léger
— **Prometheus** : collecte de métriques
— **Grafana** : visualisation des métriques
— **Kubernetes** : orchestration complète et autoscaling

2. Technologies utilisées

— **Kubernetes** : orchestration de conteneurs et gestion de l'autoscaling
— **Redis** : stockage en mémoire distribué avec réplication maître/réplique
— **Node.js** : application backend stateless pour traiter les requêtes
— **React** : interface utilisateur dynamique
— **Prometheus et Grafana** : solution de monitoring et dashboard

3. Déploiement de l'infrastructure

3.1 Prérequis

Avant de démarrer, il faut s'assurer que les outils suivants sont installés :
— `minikube`
— `kubectl`
— `docker`

3.2 Étapes

```
# Démarrer Minikube
minikube start

# Activer le metrics-server
minikube addons enable metrics-server

# Lancer le déploiement complet
./scripts/deploy-all.sh
```

4. Accès aux services

Les services déployés sont exposés via Minikube :
— `minikube service react-service` : Accès à l'application web
— `minikube service node-service` : Accès à l'API Node.js

— `minikube service prometheus` : Interface Prometheus
— `minikube service grafana` : Interface Grafana (login : admin/admin)

5. Autoscaling

Le **Horizontal Pod Autoscaler (HPA)** est activé sur :
— `redis-replica-hpa.yml`
— `node-hpa.yml`
Pour vérifier :

```
kubectl get hpa
kubectl top pod
```

6. Monitoring

— Prometheus scrape automatiquement `/metrics` depuis le backend Node.js
— Grafana est connecté à Prometheus comme source de données

7. Images Docker

Les Dockerfiles sont inclus dans :
— `NodeJs/Dockerfile`
— `React/Dockerfile`

8. Conclusion

Ce projet démontre comment déployer une application complète en environnement cloud-native en s'appuyant sur des outils modernes de conteneurisation, de monitoring et d'orchestration. L'autoscaling garantit l'élasticité de l'infrastructure et permet d'adapter dynamiquement les ressources aux besoins.

Contact

Pour toute question, vous pouvez me contacter via GitHub ou par email. **Auteur : BENCHAA Amine**