Sorbonne Université

Campus Sciences et Ingénierie, Jussieu

COMPTE RENDU PROJET Autoscaling Iac

KITOKO David 28706212

KANDIL Omar : 21107819

CRV

Introduction

Nous allons simuler la mise à l'échelle de Redis et du backend Node.js en utilisant les scripts d'automatisation sans avoir à intervenir manuellement sur les configurations Kubernetes.

Prérequis

Avant de simuler le déploiement, assurez-vous d'avoir installé les outils suivants sur votre machine :

- Docker : Pour la gestion des images Docker.
- Kubernetes (kubectl) : Pour interagir avec le cluster Kubernetes.
- Minikube : Pour créer un cluster Kubernetes local.
- Prometheus et Grafana : Pour la surveillance de la performance.

Ces outils vous permettront de déployer l'infrastructure Kubernetes localement et de simuler les tests de mise à l'échelle.

Automatisation via les Scripts

L'un des principaux objectifs de ce projet est d'automatiser les étapes de déploiement, de mise à l'échelle et de surveillance en utilisant les scripts fournis. Ces scripts permettent de simuler un déploiement complet avec une montée en charge et de tester l'autoscaling de Redis ainsi que du serveur Node.js.

Démarrer le Cluster Kubernetes

Avant d'utiliser les scripts, il faut démarrer un cluster Kubernetes avec Minikube :

```
minikube start
eval $(minikube -p minikube docker-env)
kubectl get nodes
```

Cela démarre un cluster local et vérifie que le cluster Kubernetes est fonctionnel.

Déploiement Complet avec le Script `deploy_all.sh`

Pour déployer l'ensemble de l'application (backend, frontend, Redis, Prometheus, Grafana), vous pouvez utiliser le script `deploy_all.sh`. Ce script déploie tous les composants nécessaires en une seule commande, ce qui facilite le processus de mise en place.

```
chmod +x script/deploy_all.sh
./script/deploy_all.sh
```

Ce script applique tous les fichiers YAML nécessaires pour déployer le backend Node.js, le frontend React, Redis, et les outils de monitoring (Prometheus et Grafana).

Simulation de la Mise à l'Échelle avec `scale_test.sh`

Une fois l'infrastructure déployée, vous pouvez tester l'autoscaling en simulant une charge avec le script `scale_test.sh`. Ce script simule une augmentation de la charge sur le serveur backend Node.js et Redis, ce qui déclenche l'autoscaling si la charge dépasse un certain seuil (par exemple, une utilisation CPU élevée).

```
chmod +x script/scale_test.sh
./script/scale_test.sh
```

Le script `scale_test.sh` fonctionne en envoyant une charge artificielle au backend pour observer comment Kubernetes réagit en ajoutant des réplicas supplémentaires au backend et à Redis, selon les configurations d'autoscaling définies dans les fichiers YAML.

Vérification du Statut avec `status.sh`

Après avoir simulé la mise à l'échelle, vous pouvez utiliser le script `status.sh` pour vérifier l'état de vos pods, services et de l'auto-scaling. Ce script fournit une vue d'ensemble de la santé des composants du cluster.

```
chmod +x script/status.sh
./script/status.sh
```

Le script affichera l'état des différents pods et réplicas dans Kubernetes, vous permettant de confirmer que l'autoscaling a bien eu lieu et que les services fonctionnent correctement.

Suppression des Ressources avec `delete_all.sh`

Une fois que vous avez terminé les tests, vous pouvez supprimer toutes les ressources Kubernetes du cluster avec le script `delete_all.sh`. Ce script supprime les déploiements, services et autres ressources créées lors du processus de simulation.

```
chmod +x script/delete_all.sh
./script/delete_all.sh
```

Cela réinitialise l'environnement et vous permet de repartir de zéro si nécessaire.

Simulation de l'Autoscaling

Les scripts fournis facilitent la simulation de la mise à l'échelle des services sur Kubernetes, en particulier Redis et le backend Node.js.

Autoscaling de Redis

Redis est configuré pour l'autoscaling à l'aide du fichier `redis_autoscaling.yaml`. Ce fichier définit un Horizontal Pod Autoscaler (HPA) pour Redis, ce qui permet d'ajuster automatiquement le nombre de réplicas de Redis en fonction de l'utilisation de la mémoire.

Lors de l'exécution du script `scale_test.sh`, le nombre de réplicas de Redis augmentera si la charge dépasse le seuil spécifié dans le HPA.

Autoscaling du Backend Node.js

Le backend Node.js bénéficie également de l'autoscaling. Le fichier `autoscaling_js.yaml` configure l'autoscaling en fonction de la charge CPU. Lorsqu'une charge artificielle est générée via `scale_test.sh`, Kubernetes crée de nouveaux réplicas du backend si nécessaire pour maintenir des performances optimales.

Accès au Monitoring avec Prometheus et Grafana

Une fois que l'application est déployée et que l'autoscaling est simulé, vous pouvez utiliser Prometheus et Grafana pour surveiller les performances du système. Après avoir déployé Prometheus et Grafana avec `deploy_all.sh`, vous pouvez accéder à leurs interfaces web respectives pour consulter les métriques de performance.

Accéder à Prometheus

Exécutez la commande suivante pour accéder à l'interface web de Prometheus :
installation du service metrics-server :

```
kubectl apply -f https://github.com/kubernetes-sigs/metrics-server/re

kubectl patch deployment metrics-server -n kube-system --type='json'
    -p='[{"op": "add", "path": "/spec/template/spec/containers/0/args
```

```
kubectl port-forward svc/prometheus 9090:9090
```

Accédez ensuite à `http://localhost:9090` dans votre navigateur.

Accéder à Grafana

De même, pour accéder à Grafana, exécutez :

```
kubectl port-forward svc/grafana 3000:3000
```

Accédez à `http://localhost:3000` et utilisez les identifiants par défaut `admin/admin`.

Conclusion

En utilisant les scripts fournis, vous pouvez facilement simuler le déploiement et la mise à l'échelle d'une application Node.js avec une base de données Redis sur Kubernetes. Le script `deploy_all.sh` simplifie le déploiement de l'ensemble de l'infrastructure, tandis que `scale_test.sh` permet de simuler une charge pour tester l'autoscaling. Les scripts `status.sh` et `delete_all.sh` vous aident à vérifier l'état du système et à nettoyer l'environnement après les tests. Cette approche automatisée facilite le déploiement et la gestion de votre infrastructure Kubernetes tout en garantissant que vos services peuvent se scaler dynamiquement en fonction des besoins.