POEI DevOps ajc-formation

Rapport de MiniProjet - Kubernetes

Luc ANTIMI

Professeur référent : Frazer SADO

21 octobre 2021

Table des matières

1 Introduction

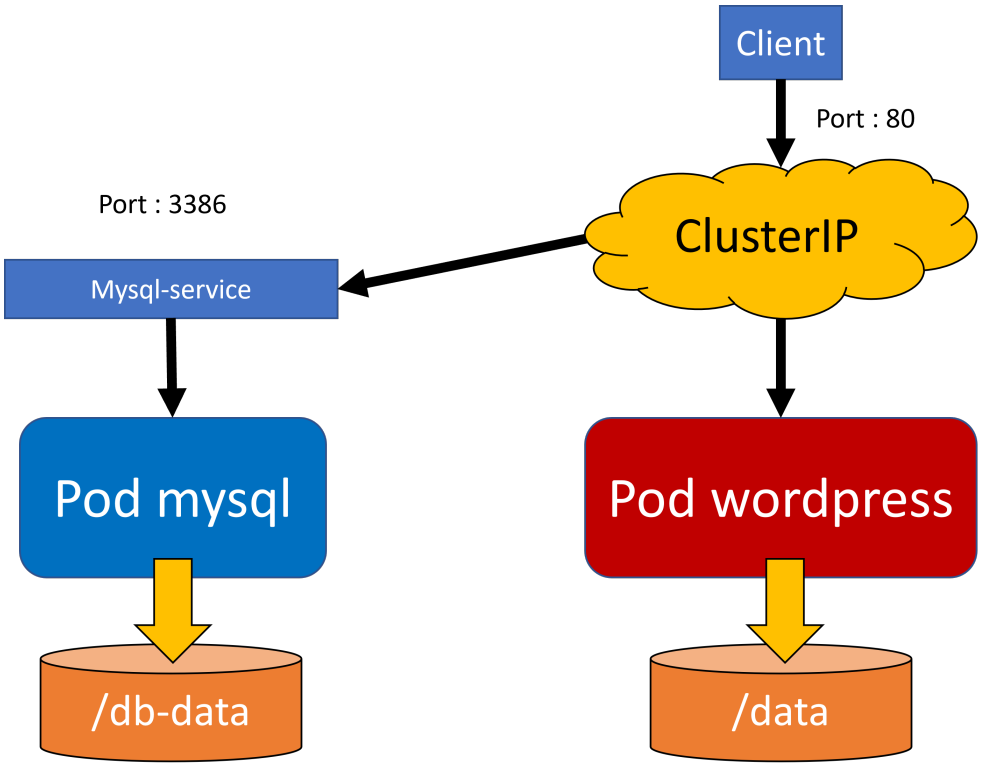
Dans le cadre de notre POEI (**P**réparation **O**pérationnelle à l'**E**mploi **I**ndividuelle) DevOps, nous avons été amené à découvrir l'outil Kubernetes, un système open-source permettant d'automatiser le déploiement, la mise à l'échelle et la gestion des applications conteneurisées. A la fin de ce module nous avons donc du réaliser un mini-projet dont voici l'énoncé.

- Déployez Wordpress à l'aide des étapes suivantes :
  - Créez un deployment mysql avec un seul réplica
  - Créez un service de type clusterIP pour exposer votre pod mysql
  - Créez un deployment wordpress avec les bonnes variables d'environnement pour se connecter à la base de données mysql
  - Votre deployment devra stocker les données de wordpress et de mysql sur un volume mounté dans le /data de votre noeud
  - Créez un service de type nodeport pour exposer le frontend Wordpress
- Nous vous conseillons d'utiliser les manifests pour réaliser cet exercice
- Rédigez un rapport retraçant l'ensemble des tâches que vous avez réalisés et poussez vos manifests sur github

Pour ce MiniProjet nous avons utilisé Katacoda.

2 Architecture

Lors de ce mini Projet nous avons utilisé l'architecture suivante :



3 Code

Pour deploy notre wordpress, nous avons chois d'utiliser des manifests en .yml nous avons donc les fichiers suivants :

Nous avons choisi de regrouper les manifests des PVC avec ceux des deployments pour plus de simplicité. Le fichier kustomization.yml sert à gérer les secrets et faciliter le lancement.

3.1 mysql-deploy.yml

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mysql-pv-claim
  labels:
```

```
    app: wordpress
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
hostPath:
    path: "/db-data"
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: wordpress-mysql
  labels:
    app: wordpress
spec:
  selector:
    matchLabels:
      app: wordpress
      tier: mysql
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: wordpress
        tier: mysql
    spec:
      containers:
      - image: mysql:5.6
        name: mysql
        env:
```

```
        - name: MYSQL_ROOT_PASSWORD
          valueFrom:
            secretKeyRef:
              name: mysql-pass
              key: password
        ports:
        - containerPort: 3306
          name: mysql
        volumeMounts:
        - name: mysql-persistent-storage
          mountPath: /var/lib/mysql
      volumes:
      - name: mysql-persistent-storage
        persistentVolumeClaim:
          claimName: mysql-pv-claim
```

3.2 mysql-service.yml

```
apiVersion: v1
kind: Service
metadata:
  name: wordpress-mysql
  labels:
    app: wordpress
spec:
  type: ClusterIP
  ports:
    - port: 3306
  selector:
    app: wordpress
    tier: mysql
```

3.3 wp-deploy.yml

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: wp-pv-claim
  labels:
    app: wordpress
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi


---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: wordpress
  labels:
    app: wordpress
spec:
  selector:
    matchLabels:
      app: wordpress
      tier: frontend
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: wordpress
```

```
        tier: frontend
    spec:
      containers:
      - image: wordpress:4.8-apache
        name: wordpress
        env:
        - name: WORDPRESS_DB_HOST
          value: wordpress-mysql
        - name: WORDPRESS_DB_PASSWORD
          valueFrom:
            secretKeyRef:
              name: mysql-pass
              key: password
        ports:
        - containerPort: 80
          name: wordpress
        volumeMounts:
        - name: wordpress-persistent-storage
          mountPath: /var/www/html
      volumes:
      - name: wordpress-persistent-storage
        persistentVolumeClaim:
          claimName: wp-pv-claim
```

3.4 front-service.yml

```
apiVersion: v1
kind: Service
metadata:
  name: wordpress-svc
spec:
  type: NodePort
  ports:
   - protocol: TCP
```

```
    targetPort: 80
    port: 8081
    nodePort: 30008
  selector:
    app: wordpress
```

3.5 kustomization.yml

```
secretGenerator:
- name: mysql-pass
  literals:
  - password=password
resources:
  - mysql-deploy.yml
  - mysql-service.yml
  - wp-deploy.yml
  - front-service.yml
```

4 Résultats

à paritr de là nous devons rentrer les commandes suivantes :

Cette commande va appeler kustomization.yml et initialiser notre déploiement.

```
kubectl apply -k ./
```

On verifie avec cette commande que notre service fonctionne

```
kubectl get services wordpress
```

si cela fonctionne on peut se rendre sur de notre service et consulter notre application