LAB 5 : Manipulation des objets Deployments

Conditions préalables

- Cluster Kubernetes fonctionnel

Exercice 1 : Deployment.

Dans cet exercice, vous allez créer un Deployment et l'exposer à l'extérieur du cluster via un service de type NodePort.

1. Spécification d'un Deployment

Créez un fichier vote_deployment.yaml définissant un Deployment ayant les propriétés suivantes:

- nom: vote
- nombre de replicas: 3
- définition d'un selector sur le label app:vote
- spécification du Pod:
    - label app:vote
    - un container nommé vote basé sur l'image instavote/vote et exposant le port 80

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: vote
spec:
  replicas: 3
  selector:
    matchLabels:
      app: vote
  template:
    metadata:
      labels:
        app: vote
    spec:
      containers:
      - name: vote
        image: instavote/vote
        ports:
        - containerPort: 80
```

2. Création du Deployment

Utilisez kubectl pour créer le Deployment

```
kubectl create -f vote-deployment.yaml
```

3. Status du Deployment

A l'aide de kubectl, examinez le status du Deployment vote.

A partir de ces informations, que pouvez-vous dire par rapport au nombre de Pods gérés par ce Deployment ?

```
kubectl get deploy vote
kubectl get rs
```

4. Status des Pods associés

A l'aide de kubectl, lister les Pods associés à ce Deployment.

```
kubectl get pods
```

5. Exposition des Pods du Deployment

Créez un Service permettant d'exposer les Pods du Deployment à l'extérieur du cluster

Conseils:

- Vous pourrez commencer par créer une spécification pour le Service, en spécifiant que le selector doit permettre de regrouper les Pods ayant le label app:vote.
- Utilisez un service de type NodePort, vous pourrez par exemple le publier sur le port 31001 des nodes du cluster
- le container basé sur l'image instavote/vote tourne sur le port 80, ce port devra donc être référencé en tant que targetPort dans la spécification du Service.

```
apiVersion: v1
kind: Service
metadata:
  name: vote-np
spec:
  selector:
    app: vote
  type: NodePort
  ports:
  - port: 80
    targetPort: 80
    nodePort: 31001
```

Note: n'hésitez pas à vous reporter à l'exercice sur les Services de type NodePort que nous avons vu précédemment

Une fois le service créé, vous pourrez accéder à l'interface de vote sur http://IP:31001 ou IP est l'adresse IP d'une machine du cluster Kubernetes.

Attention: cette interface n'est pas branchée à un backend, il n'est pas encore possible de voter, si vous cliquez sur l'un des choix, vous obtiendrez une erreur.

Exercice 2 : rolling update.

Dans cet exercice, vous allez créer un Deployment et effectuer un rolling update.

1. Création d'un Deployment

Créez un Deployment avec les propriétés suivantes:

- nom: www
- 3 replicas d'un Pod avec un container basé sur nginx:1.12

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web
spec:
  replicas: 3
  selector:
    matchLabels:
      app: www
  template:
    metadata:
      labels:
        app: www
    spec:
      containers:
      - name: vote
        image: nginx:1.12
        ports:
        - containerPort: 80
```

Spécifiez l'option --record=true à la fin de la commande afin de conserver l'historique des commandes de mises à jour du Deployment.

```
kubectl create -f www_deployment.yml  --record=true
```

2. Liste des ressources

- Listez les ressources créées par la commande précédente (Deployment, ReplicaSet, Pod).

Note: utilisez une seule fois la commande kubectl pour lister l'ensemble des ressources.

```
kubectl get deploy,rs,pod
```

3. Mise à jour de l'image

- Mettez l'image nginx à jour avec la version 1.14-alpine

```
kubectl set image deploy/www www=nginx:1.14-alpine
```

4. Liste des ressources

Une nouvelle fois, listes les ressource.

- Que constatez-vous ?

```
kubectl get deploy,rs,pod
```

5. Historique des mises à jour

- Listez les mises à jour (= révisions) du Deployment.

Note: utilisez la commande **`kubectl rollout...`**

```
kubectl rollout history deploy/www
```