LAB 4 : Manipulation des services

Conditions préalables

- Cluster Kubernetes fonctionnel

Exercice 1 : un Service de type ClusterIP.

Dans cet exercice, vous allez créer un Pod et l'exposer à l'intérieur du cluster en utilisant un Service de type ClusterIP.

1. Création d'un Pod

Créez un fichier www_pod.yaml définissant un Pod ayant les propriétés suivantes:

- nom: www
- label associé au Pod: app:www (ce label est à spécifier dans les metadatas du Pod)
- nom du container: nginx
- image du container: nginx:1.14-alpine

```
apiVersion: v1
kind: Pod
metadata:
  name: www
spec:
  containers:
  - name: nginx
    image: nginx:1.16-alpine
```

2. Lancement du Pod

La commande suivante permet de créer le Pod

```
$ kubectl create -f www_pod.yaml
```

3. Définition d'un service de type ClusterIP

Créez un fichier www_service_clusterIP.yaml définissant un service ayant les caractéristiques suivantes:

- nom: www
- type: ClusterIP
- un selector permettant le groupement des Pods ayant le label app:www.
- exposition du port 80 dans le cluster
- forward des requètes vers le port 80 des Pods sous-jacents

```
apiVersion: v1
kind: Service
metadata:
  name: www
spec:
  selector:
    app: www
  type: ClusterIP
  ports:
  - port: 80
    targetPort: 80
```

4. Lancement du Service

A l'aide de kubectl créez le Service défini dans www_service_clusterIP.yaml

```
kubectl create -f www_service_clusterIP.yaml
```

5. Accès au Service depuis le cluster

Lancez le Pod dont la spécification est la suivante:

```
apiVersion: v1
kind: Pod
metadata:
  name: debug
spec:
  containers:
  - name: debug
    image: alpine
    command:
    - "sleep"
    - "10000"
```

Nous allons utiliser ce Pod pour accèder au Service www depuis l'intérieur du cluster. Ce Pod contient un seul container, basé sur alpine et qui est lancé avec la commande sleep 10000 . Ce container sera donc en attente pendant 10000 secondes. Nous pourrons alors lancer un shell intéractif à l'intérieur de celui-ci et tester la communication avec le Service www.

- Lancez le Pod avec kubectl.
- Lancez un shell intéractif sh dans le container debug du Pod.
- Installer l'utilitaire curl

le container debug du Pod du mêm nom est basé sur l'image alpine qui ne contient pas l'utilitaire curl par défaut. Il faut donc l'installer avec la commande suivante:

```
/ # apk update && apk add curl
```

- Utilisez curl pour envoyer une requête HTTP Get sur le port 80 du service www. Vous devriez obtenir le contenu, sous forme textuel, de la page index.html servie par défaut par nginx.

Ceci montre que depuis le cluster, si l'on accède au Service www la requête est bien envoyée à l'un des Pods (nous en avons créé un seul ici) regroupé par le Service (via la clé selector).

```
kubectl create -f debug_pod.yaml
kubectl exec -ti debug – sh
```

```
/ # curl www
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
body {
width: 35em;
margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif;
}
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>
<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>
<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

6. Visualisation de la ressource

A l'aide de **`kubectl get`** , visualisez la spécification du service www.

```
kubectl get services www
```

7. Détails du service

- A l'aide de **`kubectl describe`**, listez les détails du service www
- Notez l'existence d'une entrée dans Endpoints, celle-ci correspond à l'IP du Pod qui est utilisé par le Service.

Note: si plusieurs Pods avaient le label app:www, il y aurait une entrée Endpoint pour chacun d'entre eux.

Exercice 2 : un Service de type NodePort.

Dans cet exercice, vous allez créer un Pod et l'exposer à l'extérieur du cluster en utilisant un Service de type NodePort.

1. Création d'un Pod

Créez un fichier www_pod.yaml définissant un Pod ayant les propriétés suivantes:

- nom: www
- label associé au Pod: app:www (ce label est à spécifier dans les metadatas du Pod)
- nom du container: nginx
- image du container: nginx:1.14-alpine

```
apiVersion: v1
kind: Pod
metadata:
  name: www
spec:
  containers:
  - name: nginx
    image: nginx:1.16-alpine
```

2. Lancement du Pod

La commande suivante permet de créer le Pod

```
$ kubectl create -f www_pod.yaml
```

3. Définition d'un service de type NodePort

Créez un fichier www_service_NodePort.yaml définissant un service ayant les caractéristiques suivantes:

- nom: www-np
- type: NodePort
- un selector permettant le groupement des Pods ayant le label app:www.
- forward des requètes vers le port 80 des Pods sous-jacents
- exposition du port 80 dans le cluster
- exposition du port 31000 sur le cluster

```
apiVersion: v1
kind: Service
metadata:
  name: www-np
spec:
  selector:
```

```
    app: www
  type: NodePort
  ports:
  - port: 80
    targetPort: 80
    nodePort: 31000
```

4. Lancement du Service

- A l'aide de **`kubectl`** créez le Service défini dans www_service_NodePort.yaml

```
kubectl create -f www_service_nodePort.yaml
```

5. Accès au Service depuis l'extérieur

- Lancez un navigateur sur le port 31000 de l'une des machines du cluster.

http://<IP>:31000