Lab3 – Les services

Brahim HAMDI

Introduction

Les services permettent aux applications déployées sur Kubernetes de communiquer ensemble, c'est l'une des briques les plus importantes de la partie réseau.

Nous allons voir les différents types de services et les notions associées :

- ClusterIP
- NodePort
- ExternalIP
- LoadBalancer

ClusterIP

Chaque cluster Kubernetes dispose d'un réseau interne pour les services. Le type de service par défaut est **ClusterIP** , ces IP sont joignables uniquement à l'intérieur du cluster.

Avoir une seule IP permet de load balancer le trafic automatiquement entre de multiples pods (replicas).

1. Dans cette partie nous allons créer un déploiement et un service ClusterIP qui va l'exposer en interne du cluster.

◦ Supprimer tous les objets du le namespace par défaut

```
brahim@Training:~/Lab3$ kubectl delete all --all
pod "first-deployment-685cd76f87-g8v5f" deleted
pod "first-deployment-685cd76f87-gv9kx" deleted
pod "first-deployment-685cd76f87-j8q7m" deleted
service "first-deployment" deleted
service "kubernetes" deleted
deployment.apps "first-deployment" deleted
```

◦ Copier le contenu suivant dans le fichier *clusterip.yaml* et appliquez le sur le cluster . Quelle est l'IP du service ?

```
apiVersion: v1
kind: Service
metadata:
  name: webapp1-clusterip-svc
  labels:
    app: webapp1-clusterip
spec:
  ports:
  - port: 80
  selector:
    app: webapp1-clusterip
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: webapp1-clusterip-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: webapp1-clusterip
  template:
    metadata:
      labels:
        app: webapp1-clusterip
    spec:
      containers:
      - name: webapp1-clusterip-pod
        image: katacoda/docker-http-server:latest
        ports:
        - containerPort: 80
```

```
brahim@Training:~/Lab3$ kubectl  apply -f clusterip.yaml
service/webapp1-clusterip-svc created
deployment.apps/webapp1-clusterip-deployment created
brahim@Training:~/Lab3$
brahim@Training:~/Lab3$ kubectl get all -o wide
NAME                                              READY   STATUS    RESTARTS   AGE   IP            NODE          NOMINATED NODE   READINESS
GATES
pod/webapp1-clusterip-deployment-648c8cb679-c9dms   1/1     Running   0          7s    10.244.1.28   k8s-worker1   <none>           <none>
pod/webapp1-clusterip-deployment-648c8cb679-xx4dl   1/1     Running   0          7s    10.244.2.30   k8s-worker2   <none>           <none>

NAME                           TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE     SELECTOR
service/kubernetes             ClusterIP   10.96.0.1       <none>        443/TCP   2m49s   <none>
service/webapp1-clusterip-svc  ClusterIP   10.97.209.216   <none>        80/TCP    7s      app=webapp1-clusterip

NAME                                         READY   UP-TO-DATE   AVAILABLE   AGE   CONTAINERS              IMAGES
     SELECTOR
deployment.apps/webapp1-clusterip-deployment   2/2     2            2           7s    webapp1-clusterip-pod   katacoda/docker-http-server:late
st   app=webapp1-clusterip

NAME                                                    DESIRED   CURRENT   READY   AGE   CONTAINERS              IMAGES
           SELECTOR
replicaset.apps/webapp1-clusterip-deployment-648c8cb679   2         2         2       7s    webapp1-clusterip-pod   katacoda/docker-http-serve
r:latest   app=webapp1-clusterip,pod-template-hash=648c8cb679
brahim@Training:~/Lab3$
```

◦ Remarquez que, par défaut, le port du service est la même que le port du conteneur (80). Il est possible de dissocier le port du conteneur du port du service grâce à la notion de *targetPort*.

Modifier le fichier en remplaçant le *port* du service et en ajoutant le *targetPort*, puis ré-appliquer.

```
spec:
  ports:
  - port: 8080
    targetPort: 80
  selector:
```

```
brahim@Training:~/Lab3$ kubectl apply -f clusterip.yaml
service/webapp1-clusterip-svc configured
deployment.apps/webapp1-clusterip-deployment unchanged
brahim@Training:~/Lab3$
brahim@Training:~/Lab3$ kubectl get svc
NAME                     TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
kubernetes               ClusterIP   10.96.0.1       <none>        443/TCP    11m
webapp1-clusterip-svc    ClusterIP   10.97.209.216   <none>        8080/TCP   9m2s
brahim@Training:~/Lab3$
```

Les IP s de service ne sont pas joignables directement car nous sommes situés à l'extérieur du cluster. Pour y accéder, nous devons publier le service.

NodePort

Les services de type **NodePort** permettent d'exposer un service à l'extérieur du cluster en mappant un port sur tous les noeuds d'un cluster.

2. Supprimer tous les objets du default namespace.

◦ Créer le fichier *nodeport.yaml* et y copier le contenu suivant, puis appliquer-le.

```
apiVersion: v1
kind: Service
metadata:
  name: webapp1-nodeport-svc
  labels:
    app: webapp1-nodeport
```

```
spec:
  type: NodePort
  ports:
  - port: 80
    nodePort: 30080
  selector:
    app: webapp1-nodeport
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: webapp1-nodeport-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: webapp1-nodeport
  template:
    metadata:
      labels:
        app: webapp1-nodeport
    spec:
      containers:
```

- name: webapp1-nodeport-pod

image: katacoda/docker-http-server:latest

ports:

- containerPort: 80

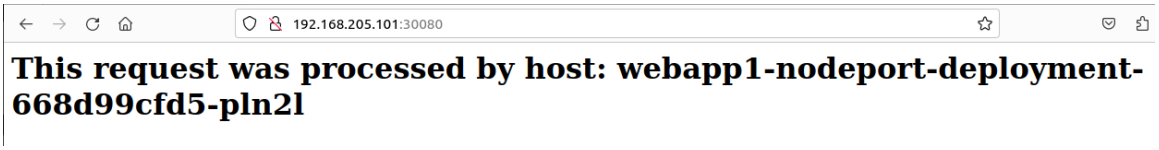
```
brahim@Training:~/Lab3$ kubectl  apply -f nodeport.yaml
service/webapp1-nodeport-svc created
deployment.apps/webapp1-nodeport-deployment created
brahim@Training:~/Lab3$
brahim@Training:~/Lab3$ kubectl get all -owide
NAME                                              READY   STATUS    RESTARTS   AGE   IP            NODE          NOMINATED NODE   READINESS G
ATES
pod/webapp1-nodeport-deployment-668d99cfd5-f5k9c   1/1     Running   0          3s    10.244.2.34   k8s-worker2   <none>           <none>
pod/webapp1-nodeport-deployment-668d99cfd5-pln2l   1/1     Running   0          3s    10.244.1.32   k8s-worker1   <none>           <none>

NAME                          TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE   SELECTOR
service/kubernetes            ClusterIP   10.96.0.1       <none>        443/TCP        4s    <none>
service/webapp1-nodeport-svc  NodePort    10.106.158.45   <none>        80:30080/TCP   4s    app=webapp1-nodeport

NAME                                         READY   UP-TO-DATE   AVAILABLE   AGE   CONTAINERS             IMAGES
   SELECTOR
deployment.apps/webapp1-nodeport-deployment   2/2     2            2           3s    webapp1-nodeport-pod   katacoda/docker-http-server:latest
   app=webapp1-nodeport

NAME                                                    DESIRED   CURRENT   READY   AGE   CONTAINERS             IMAGES
         SELECTOR
replicaset.apps/webapp1-nodeport-deployment-668d99cfd5   2         2         2       3s    webapp1-nodeport-pod   katacoda/docker-http-server:
latest   app=webapp1-nodeport,pod-template-hash=668d99cfd5
brahim@Training:~/Lab3$
```

◦ Récupérez l'IP de l'un des nœuds du cluster et accédez au service sur le port 30080.



Ce type de service permet d'exposer un ensemble de pods sur tous les noeuds d'un cluster. Si vous souhaitez exposer le service uniquement sur une IP , il existe un autre type de service comme **ExternalIP** et **LoadBalancer**.