Exercice

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*

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

4. Lancement du Service

A l'aide de *kubectl* créez le Service défini dans *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*).

6. Visualisation de la ressource

A l'aide de `kubectl get`, visualisez la spécification du service *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 plusiseurs Pods avaient le label *app: www*, il y aurait une entrée Endpoint pour chacun d'entre eux.

Correction

1. Création du Pod

La spécification du Pod est la suivante:

```
apiVersion: v1
kind: Pod
metadata:
  name: www
  labels:
    app: www
spec:
  containers:
  - name: nginx
    image: nginx:1.14-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

La spécification du Service demandé est la suivante:

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

4. Lancement du Service

La commande suivante permet de lancer le Service:

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

5. Accès au Service depuis le cluster

- Nous définissons la spécification suivante dans le fichier *debug_pod.yaml*:

```
$ cat <<EOF > debug_pod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: debug
spec:
  containers
  - name: debug
    image: alpine
    command:
    - "sleep"
    - "10000"
EOF
```

- La commande suivante permet de lancer le Pod

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

- La commande suivante permet de lancer un shell *sh* intéractif dans le container *debug* du Pod

```
$ kubectl exec -ti debug -- sh
```

- Le service *www* est directement accessible à l'intérieur du cluster (c'est à dire par les Pods tournant sur le cluster) par son nom:

```
/ # 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

La commande suivante permet d'obtenir une vue d'ensemble du service *www*

```
$ kubectl get services www
NAME      TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
www       ClusterIP   10.102.43.122   <none>        80/TCP    1h
```

On ajout l'option *-o yaml* pour avoir la spécification du service au format *yaml*.

```
$ kubectl get svc/www -o yaml
apiVersion: v1
kind: Service
metadata:
  creationTimestamp: 2018-04-01T13:37:42Z
  labels:
    app: www
  name: www
  namespace: default
  resourceVersion: "1214717"
  selfLink: /api/v1/namespaces/default/services/www
  uid: da271ad7-35b1-11e8-80f1-080027f0e385
spec:
  clusterIP: 10.102.43.122
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: www
  sessionAffinity: None
  type: ClusterIP
status:
  loadBalancer: {}
```

Note: comme nous l'avions vu pour les Pods, les commandes suivantes sont équivalentes (et utilisent des raccourcis pratiques):

- kubectl get services www
- kubectl get service www
- kubectl get svc www
- kubectl get svc/www

6. Détails du service

La commande suivante permet d'avoir les détails du service www

```
$ kubectl describe svc/www
Name:              www
Namespace:         default
Labels:            app=www
Annotations:       <none>
Selector:          app=www
Type:              ClusterIP
IP:                10.102.43.122
```

```
Port:              <unset>  80/TCP
TargetPort:        80/TCP
Endpoints:         172.17.0.10:80
Session Affinity:  None
Events:            <none>
```