2024

Lab 5 - Communiquer avec les Pods via des règles de routage Ingress

COMPTE RENDU

YOUSSEF BEN GHORBEL

Table des matières

Introduction

Dans cet exercice, nous explorons les Ingress, une solution pour accéder aux Pods depuis l'extérieur d'un cluster Kubernetes, sans utiliser de Services de type NodePort ou LoadBalancer

Objectifs

- Écrire une configuration Ingress;
- Créer des règles de type fanoutou hôtes virtuels ;
- Gérer des hôtes virtuels.

Étapes à suivre

1. Préparation

- Supprimer le Namespace précédent :

  kubectl delete namespace mynamespaceexercice3

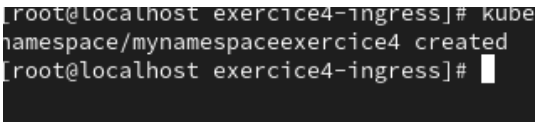
2. Créer un Namespace :

- Créer un fichier appelé mynamespaceexercice4.yaml

```
apiVersion:
v1 kind:
Namespace
metadata:
  name: mynamespaceexercice4
```

- Appliquez le fichier :

```
$ kubectl apply -f exercice4-ingress/mynamespaceexercice4.yaml
```



3. Créer les Deployments et Services

a. Application 1

- Créer app1deployment.yaml :

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name:
```

```
app1deployment
spec:
  replicas: 2
  selector:
    matchLabels
      : app:
      app1pod
  template:
    metadata:
      labels:
        app: app1pod
    spec:
      containers:
      - name:
        app1container
        image:
        nginx:latest
        ports:
        -   containerPo
        rt:          80
        lifecycle:
          postStart:
            exec:
              command:
                - /bin/sh
                - -c
                - >

                  mkdir
                  /usr/share/nginx/html/app1;
                  echo App 1 fanout from
                  $HOSTNAME >
                  /usr/share/nginx/html/app1/index.html;
                  echo App 1 vhosts from $HOSTNAME >
                  /usr/share/nginx/html/index.html
---


apiVersion: v1 kind: Service metadata:
name: app1service spec:
```

```
selector:
app: app1pod type: ClusterIP ports:
-    protocol: TCP targetPort: 80
port: 8080
```

b. Application 2

- Créer app2deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name:
app2deployment spec:
  replicas: 2
  selector:
    matchLabels:
      app:
      app2pod
  template:
    metadata:
      labels:
        app: app2pod
    spec:
      containers:
      - name:
        app2container
        image:
        nginx:latest
        ports:
        -   containerPor
        t: 80 lifecycle:
          postStart:
            exec:
              command:
                - /bin/sh
                - -c
                - >
                  mkdir /usr/share/nginx/html/app2;
                  echo App 2 fanout from $HOSTNAME >
                  /usr/share/nginx/html/app2/index.html;
                  echo App 2 vhosts from $HOSTNAME >
                  /usr/share/nginx/html/index.html
```

```
apiVersion: v1 kind:
Service metadata:
  name: app2service spec:
  selector:
    app: app2pod
  type: ClusterIP
  ports:
    - protocol: TCP
      targetPort: 80
      port: 8080
```

- **Appliquez la configuration :**

- Appliquer les fichiers :

```
kubectl apply -f exercice4-ingress/app1deployment.yaml -n
mynamespaceexercice4
kubectl apply -f exercice4-ingress/app2deployment.yaml -n
mynamespaceexercice4
```

```
[root@localhost exercice4-ingress]# kubec
deployment.apps/app1deployment created
service/app1service created
```

```
[root@localhost exercice4-ingress]# kube
deployment.apps/app2deployment created
service/app2service created
[root@localhost exercice4-ingress]#
```

4. Créer l'Ingress

- Créer myingressfanout.yaml :

```
apiVersion:
networking.k8s.io/v1 kind:
Ingress
metadata:
  name:
myingressfanout
spec:
  rules:
    - http:
        paths:
        - path: /app1
          pathType:
          Prefix
          backend:
            service:
              name: app1service
              port:
                number: 8080
        - path: /app2
          pathType:
          Prefix
          backend:
            service:
              name: app2service
              port:
               number: 8080
```

- Appliquez la configuration :

```
kubectl apply -f exercice4-ingress/myingressfanout.yaml -n
mynamespaceexercice4
```

```
[root@localhost exercice4-ingress]# kubectl apply -
ingress.networking.k8s.io/myingressfanout created
[root@localhost exercice4-ingress]#
```

5. Tester l'Ingress

- Vérifier la configuration :

```
kubectl describe ingress -n mynamespaceexercice4 myingressfanout
```

```
Name:             myingressfanout
Labels:           <none>
Namespace:        mynamespaceexercice4
Address:          172.18.0.2,172.18.0.3,172.18.0.4
Ingress Class:    traefik
Default backend:  <default>
Rules:
  Host        Path  Backends
  ----        ----  --------
  *
              /app1   app1service:8080 (10.42.0.28:80,10.42.2.21:80)
              /app2   app2service:8080 (10.42.1.27:80,10.42.2.22:80)
Annotations:  <none>
Events:       <none>
```

- Tester les applications :

```
curl localhost:80/app1/
```

```
[root@localhost exercice4-ingress]# curl localhost:80/app1/
App 1 fanout from app1deployment-85f6cdd948-bwslb
[root@localhost exercice4-ingress]#
```

```
curl localhost:80/app2/
```

```
[root@localhost exercice4-ingress]# curl localhost:80/app2/
App 2 fanout from app2deployment-869dc569d-2fkpk
[root@localhost exercice4-ingress]#
```

6. Configurer les Hôtes Virtuels

- Modifier /etc/hosts :

  127.0.0.1 app1.mydomain.test

  127.0.0.1 app2.mydomain.test

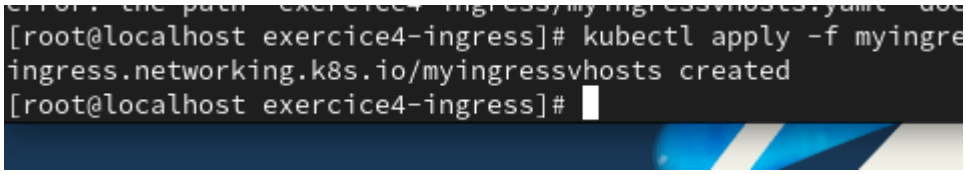
- Créer myingressvhosts.yaml :

```
apiVersion:
networking.k8s.io/v1 kind:
Ingress
metadata:
  name:
myingressvhosts
```

```
spec:
  rules:
    - host:
      "app1.mydomain.test"
      http:
        paths:
          - path: /
            pathType:
            Prefix
            backend:
              service:
                name:
                app1service
                port:
                  number: 8080
    - host:
      "app2.mydomain.test"
      http:
        paths:
          - path: /
            pathType:
            Prefix
            backend:
              service:
                name:
                app2service
                port:
                  number: 8080
```

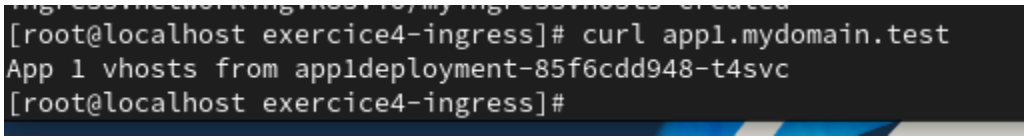
- Appliquez la configuration :

  `kubectl apply -f exercice4-ingress/myingressvhosts.yaml -n mynamespaceexercice4`



- Tester les hôtes virtuels :

  $ curl app1.mydomain.test



  $ curl app2.mydomain.test