Conteneurisation des Applications

Filière : ASEDS Semestre : 3 Année universitaire : 2023/2024

Prof. Driss ALLAKI

Corrigé du TP Kubernetes

Considérons une application web 3-tiers (composée de frontend, backend et base de données) développée avec le MERN stack (MongoDB, Express, React, Node.js).

Le code source de cette application est disponible sur ces repos : frontend et backend.

Prérequis : TP - Docker

Travail à réaliser :

Dans ce TP, il est permis d'utiliser n'importe quel outil permettant de créer un cluster Kubernetes en local (ex : Minikube, MicroK8s, kind, k3d, k3s, etc.)

1. Créer les Kubernetes manifests (fichiers YAML) nécessaires pour déployer cette application dans un cluster Kubernetes (sous un namespace nommé "exam").
    a. Deployment
    b. Service
    c. ConfigMap
    d. Secret
    e. Statefulset
    f. PV / PVC

    NB2 : Créer 2 réplicas pour chaque partie de l'application (frontend, backend et base de données)

Corrigé :

Pour ce mini-corrigé, nous allons utiliser **Minikube**.

On lance minikube (avec une configuration personnalisée si c'est possible)

```
> minikube start --memory 8192 --cpus 4 --vm-driver virtualbox
```

Créer le namespace "exam"

Option 1 : Via une commande kubectl

```
> kubectl create namespace exam
```

Option 2 : Via un fichier YAML (nommé `exam-namespace.yaml`)

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

Puis taper la commande

```
> kubectl apply –f exam-namespace.yaml
```

Avant de commencer la création des manifestes pour le déploiement, on doit faire des petites modifications sur le code source du fichier « server.js ». On va utiliser des variables d'environnement pour la connexion avec la base de données, cette modification va nous permettre d'utiliser les objets « Configmap » et « Secret » pour le déploiement de notre application

```
const username = process.env.MONGO_USERNAME;
const password = process.env.MONGO_PASSWORD;
const dbhost = process.env.DB_HOST;
const dbname = process.env.DB_NAME;
const authSource = process.env.AUTH_SOURCE;

const conectionString = 'mongodb://'+username+':'+password+'@'+dbhost+'/'+dbname+'?authSource='+authSource;
mongoose
    .connect(conectionString, {
            useNewUrlParser: true,
            useUnifiedTopology: true
        })
    .then(() => console.log("MongoDB successfully connected"))
    .catch(err => console.log(err));
```

L'image Docker « **dall49/backend:1.0.1** » est celle qui couvre ces changements.

Un dernier point à prendre en considération, c'est la mise en place en local d'un moyen permettant d'encoder des mots de passe en base 64. Pour notre cas, on va installer via le package manager de windows « chocolatey » l'utilitaire « base64 ». Puis l'utiliser pour afficher une chaîne de caractères encodée en base 64 qu'on va utiliser après dans notre fichier Secret.

```
> choco install base64

> echo yourpassword | base64
```

Le manifest YAML de la base de données (contenant : Secret, Service et StatefulSet)

database.yaml

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: mongodb-secret
data:
  password: aGFtZGFuZTIwMjI=
  username: aGFtZGFuZQ==
---
apiVersion: v1
kind: Service
metadata:
  name: mongodb-service
spec:
  selector:
    app: mongodb
  ports:
    - port: 27017
      protocol: TCP
      targetPort: 27017
---
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: mongodb-statefulset
spec:
  replicas: 2
  serviceName: mongodb
  selector:
    matchLabels:
      app: mongodb
  template:
    metadata:
      labels:
        app: mongodb
    spec:
      terminationGracePeriodSeconds: 10
      containers:
        - name: mongodb-service
          image: mongo:5.0
          ports:
            - containerPort: 27017
              name: db-port
          env:
            - name: MONGO_INITDB_ROOT_USERNAME
              valueFrom:
                secretKeyRef:
                  name: mongodb-secret
                  key: username
            - name: MONGO_INITDB_ROOT_PASSWORD
              valueFrom:
                secretKeyRef:
                  name: mongodb-secret
                  key: password
```

```
            volumeMounts:
              - name: mongo-storage
                mountPath: /data/db
      volumeClaimTemplates:
        - metadata:
            name: mongo-storage
          spec:
            accessModes: ["ReadWriteOnce"]
            resources:
              requests:
                storage: 1Gi
```

Le manifest YAML du backend (contenant : ConfigMap, Service et Deployment)

backend.yaml

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: mongodb-conf
data:
  host: mongodb-service
  dbname: exam
  authSource: admin
---
apiVersion: v1
kind: Service
metadata:
  name: backend
spec:
  selector:
    app: backend
  ports:
    - port: 5000
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: backend-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: backend
  template:
    metadata:
      labels:
        app: backend
    spec:
      containers:
        - name: backend
          image: dall49/backend:1.0.1
          ports:
            - containerPort: 5000
              name: backend-port
          env:
```

```
                - name: DB_HOST
                  valueFrom:
                    configMapKeyRef:
                      name: mongodb-conf
                      key: host
                - name: DB_NAME
                  valueFrom:
                    configMapKeyRef:
                      name: mongodb-conf
                      key: dbname
                - name: AUTH_SOURCE
                  valueFrom:
                    configMapKeyRef:
                      name: mongodb-conf
                      key: authSource
                - name: MONGO_USERNAME
                  valueFrom:
                    secretKeyRef:
                      name: mongodb-secret
                      key: username
                - name: MONGO_PASSWORD
                  valueFrom:
                    secretKeyRef:
                      name: mongodb-secret
                      key: password
```

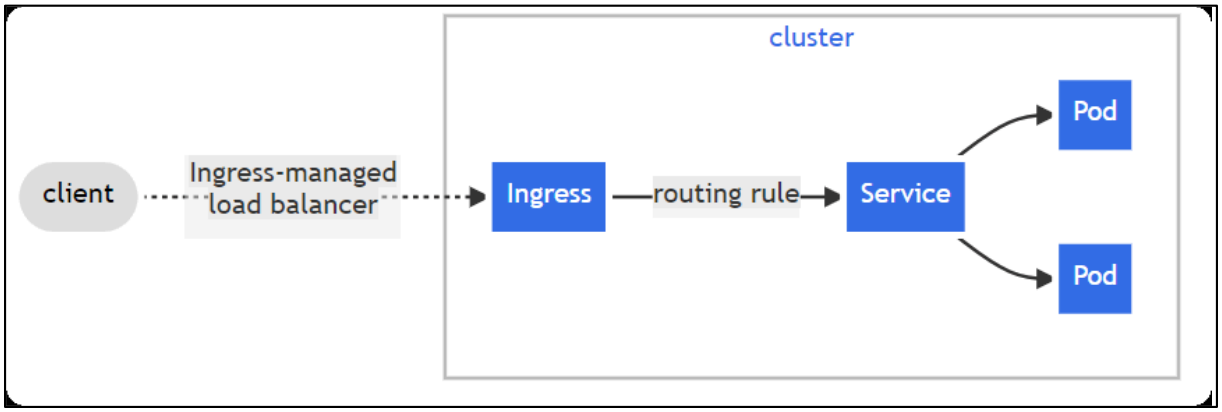
Le manifest YAML du frontend (contenant : Service et un Deployment)

frontend.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: frontend-service
spec:
  ports:
    - port: 80
  selector:
    app: frontend
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: frontend
  template:
    metadata:
      labels:
        app: frontend
    spec:
      containers:
        - name: frontend
```

```
            image: dall49/frontend:1.0.0
            ports:
              - containerPort: 80
                name: http-port
```

2. Créer un Ingress pour accéder à l'application depuis un navigateur moyennant un nom de domaine.



Il est possible de l'ajouter dans le manifest YAML du frontend.

Ou bien lui créer un fichier à part.

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: app-ingress
  annotations:
    kubernetes.io/ingress.class: "nginx"
    nginx.ingress.kubernetes.io/add-base-url: "true"
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
    - host: exam-aseds-ine2.com
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: frontend-service
                port:
                  number: 80
```

Maintenant pour accéder à notre application depuis le nom de domaine qu'on a défini, on doit modifier le fichier « **C:\Windows\System32\drivers\etc\hosts** » sur notre machine, ce fichier est utilisé pour stocker les noms d'hôtes et les adresses IP correspondantes, on ajoute la ligne suivante :

```
# Copyright (c) 1993-2009 Microsoft Corp.
#
# This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
#
# This file contains the mappings of IP addresses to host names. Each
# entry should be kept on an individual line. The IP address should
# be placed in the first column followed by the corresponding host name.
# The IP address and the host name should be separated by at least one
# space.
#
# Additionally, comments (such as these) may be inserted on individual
# lines or following the machine name denoted by a '#' symbol.
#
# For example:
#
#      102.54.94.97     rhino.acme.com          # source server
#       38.25.63.10     x.acme.com              # x client host

# localhost name resolution is handled within DNS itself.
#   127.0.0.1       localhost
#   ::1             localhost

192.168.99.102 exam-aseds-ine2.com
```

L'adresse 192.168.99.102 est l'adresse IP de la VM de minikube, on peut obtenir cette adresse par la commande « `minikube ip` ».

Il faudra aussi activer le contrôleur Ingress au niveau de minikube avec la commande suivante : « `minikube addons enable ingress` ».

3. S’assurer que l’application a été bien conteneurisée et qu’elle fonctionne correctement. (Prendre des prises d’écran du navigateur)

```
> kubectl apply -f database.yaml -n=exam

> kubectl apply -f backend.yaml -n=exam

> kubectl apply -f frontend.yaml -n=exam
```

(Puis exécuter la commande : **kubectl get all –o wide)**

Conteneurisation des Applications

Filière : ASEDS Semestre : 3 Année universitaire : 2023/2024

Prof. Driss ALLAKI

```
PS C:\Users\INPT\Desktop\S4 SUD Courses\Containers - TP\merncontainers\kubernetes> kubectl get all -o wide -n=exam
NAME                                    READY   STATUS    RESTARTS   AGE     IP            NODE       NOMINATED NODE   READINESS
 GATES
pod/backend-deployment-6457c67dcd-qttbb    1/1     Running   0          35m     172.17.0.39   minikube   <none>           <none>
pod/backend-deployment-6457c67dcd-xgv7c    1/1     Running   0          35m     172.17.0.40   minikube   <none>           <none>
pod/frontend-deployment-678df798dc-bnjr4   1/1     Running   0          3m35s   172.17.0.43   minikube   <none>           <none>
pod/frontend-deployment-678df798dc-jd445   1/1     Running   0          3m35s   172.17.0.42   minikube   <none>           <none>
pod/mongodb-statefulset-0                  1/1     Running   0          48m     172.17.0.37   minikube   <none>           <none>
pod/mongodb-statefulset-1                  1/1     Running   0          44m     172.17.0.38   minikube   <none>           <none>

NAME                      TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)     AGE     SELECTOR
service/backend           ClusterIP   10.100.151.167   <none>        5000/TCP    42m     app=backend
service/frontend-service  ClusterIP   10.104.230.203   <none>        80/TCP      3m35s   app=frontend
service/mongodb-service   ClusterIP   10.101.128.84    <none>        27017/TCP   48m     app=mongodb

NAME                                  READY   UP-TO-DATE   AVAILABLE   AGE     CONTAINERS   IMAGES                   SELECTOR
deployment.apps/backend-deployment    2/2     2            2           35m     backend      dall49/backend:1.0.1     app=backend
deployment.apps/frontend-deployment   2/2     2            2           3m35s   frontend     dall49/frontend:1.0.0    app=frontend

NAME                                             DESIRED   CURRENT   READY   AGE     CONTAINERS   IMAGES                   SELECTOR
replicaset.apps/backend-deployment-6457c67dcd    2         2         2       35m     backend      dall49/backend:1.0.1     app=backe
nd,pod-template-hash=6457c67dcd
replicaset.apps/frontend-deployment-678df798dc   2         2         2       3m35s   frontend     dall49/frontend:1.0.0    app=front
end,pod-template-hash=678df798dc

NAME                                 READY   AGE   CONTAINERS        IMAGES
statefulset.apps/mongodb-statefulset   2/2     48m   mongodb-service   mongo:5.0
```