Lab5 – Les volumes

Brahim HAMDI

Introduction

Le volume persistant utilise un fichier ou un répertoire pour émuler un stockage connecté au réseau.

Dans ce Lab nous allons utiliser un volume de type ***hostPath*** qui se trouve dans un répertoire */website* sur un nœud du cluster et sa taille est 100 Mo.

Création de PV et PVC

1. Dans la première partie, vous allez créer le PV (*PersistentVolumes*) qui représente un élément de stockage dans le cluster et que l'administrateur va le provisionner manuellement (pas de *storageClass*). Vous allez créer aussi le PVC (*PersistentVolumeClaim*) qui joue le rôle d'une requête permettant de consommer les ressources de stockage.

◦ Créez le fichier *pv-volume.yml* et y ajoutez le contenu ci-dessous. Appliquez le fichier et vérifiez la création du PV.

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: task-pv-volume
  labels:
    type: local
spec:
  capacity:
    storage: 100Mi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/website"
    type: DirectoryOrCreate
```

```
brahim@Training:~/Lab4$ kubectl  apply -f pv-volume.yaml
persistentvolume/task-pv-volume created
brahim@Training:~/Lab4$
brahim@Training:~/Lab4$ kubectl get pv
NAME             CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM   STORAGECLASS   REASON   AGE
task-pv-volume   100Mi      RWO            Retain           Available                                   5s
brahim@Training:~/Lab4$
```

◦ Créez le fichier *pvc-claims.yml* et y ajoutez le contenu ci-dessous. Appliquez le fichier et vérifiez la création du PVC.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: task-pv-claim
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 30Mi
```

```
brahim@Training:~/Lab4$ kubectl apply -f pvc-claims.yaml
persistentvolumeclaim/task-pv-claim created
brahim@Training:~/Lab4$
brahim@Training:~/Lab4$ kubectl get pv,pvc
NAME                              CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM                   STORAGECLASS   REASON   AGE
persistentvolume/task-pv-volume   100Mi      RWO            Retain           Bound    default/task-pv-claim                           85s

NAME                                    STATUS   VOLUME           CAPACITY   ACCESS MODES   STORAGECLASS   AGE
persistentvolumeclaim/task-pv-claim     Bound    task-pv-volume   100Mi      RWO                           10s
brahim@Training:~/Lab4$
```

Création du déploiement

2. Maintenant que le *PersistentVolumeClaim* est lié à votre *PersistentVolume*, l'étape suivante consiste à créer un *déploiement* qui utilise votre *PersistentVolume* comme volume. Là ou on va monter */website* qui se trouve sur le volume persistant vers le répertoire d'hébergement par défaut */www* du pod. On va exposer le déploiement par un service de type *NodePort*.

◦ Créez le fichier *deploy-with-vol.yaml* et y ajoutez le contenu suivant :

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: first-deployment
  labels:
    app: app1
spec:
```

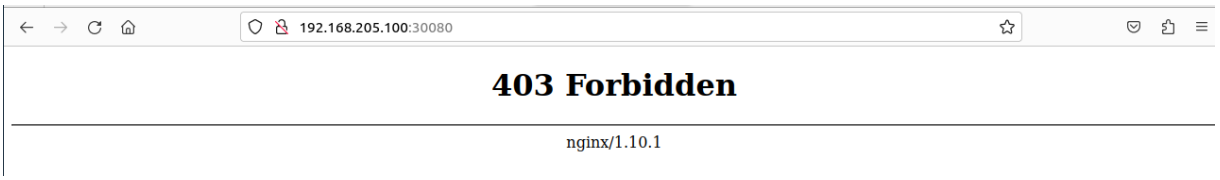
```
  selector:
    matchLabels:
      app: app1
  template:
    metadata:
      labels:
        app: app1
    spec:
      containers:
      - name: container1
        image: particule/helloworld:2.0.0
        ports:
        - containerPort: 80
        volumeMounts:
#         - mountPath: "/usr/share/nginx/html"
        - mountPath: "/www"
          name: task-pv-storage
      volumes:
      - name: task-pv-storage
        persistentVolumeClaim:
          claimName: task-pv-claim
---
apiVersion: v1
kind: Service
metadata:
  labels:
    app: app1
  name: first-deployment
spec:
  ports:
  - nodePort: 30080
    port: 80
    targetPort: 80
  selector:
    app: app1
  type: NodePort
```

◦ Appliquez le fichier yml et vérifiez si le déploiement est bien crée

```
brahim@Training:~/Lab4$ kubectl apply -f deploy-with-vol.yaml
deployment.apps/first-deployment created
service/first-deployment created
brahim@Training:~/Lab4$
brahim@Training:~/Lab4$ kubectl get all -owide
NAME                                    READY   STATUS        RESTARTS   AGE   IP            NODE          NOMINATED NODE   READINESS GATES
pod/first-deployment-5554d585bd-7fbn7   1/1     Terminating   0          46s   10.244.2.58   k8s-worker2   <none>           <none>
pod/first-deployment-5554d585bd-d4hbb   1/1     Running       0          5s    10.244.1.52   k8s-worker1   <none>           <none>

NAME                       TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)        AGE   SELECTOR
service/first-deployment   NodePort    10.110.17.53   <none>        80:30080/TCP   5s    app=app1
service/kubernetes         ClusterIP   10.96.0.1      <none>        443/TCP        40m   <none>

NAME                               READY   UP-TO-DATE   AVAILABLE   AGE   CONTAINERS   IMAGES                        SELECTOR
deployment.apps/first-deployment   1/1     1            1           5s    container1   particule/helloworld:2.0.0   app=app1

NAME                                          DESIRED   CURRENT   READY   AGE   CONTAINERS   IMAGES                        SELECTOR
replicaset.apps/first-deployment-5554d585bd   1         1         1       5s    container1   particule/helloworld:2.0.0   app=app1,pod-templat
e-hash=5554d585bd
brahim@Training:~/Lab4$
```

```
brahim@Training:~/Lab4$ kubectl describe deployment.apps/first-deployment
Name:                   first-deployment
Namespace:              default
CreationTimestamp:      Wed, 31 May 2023 05:30:06 +0200
Labels:                 app=app1
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               app=app1
Replicas:               1 desired | 1 updated | 1 total | 1 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=app1
  Containers:
   container1:
    Image:        particule/helloworld:2.0.0
    Port:         80/TCP
    Host Port:    0/TCP
    Environment:  <none>
    Mounts:
      /www from task-pv-storage (rw)
  Volumes:
   task-pv-storage:
    Type:       PersistentVolumeClaim (a reference to a PersistentVolumeClaim in the same namespace)
    ClaimName:  task-pv-claim
    ReadOnly:   false
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  <none>
NewReplicaSet:   first-deployment-5554d585bd (1/1 replicas created)
Events:
  Type    Reason             Age   From                   Message
```

◦ Affichez l’interface de votre application. Que remarquez-vous ? Pourquoi ?




```
brahim@Training:~/Lab4$ kubectl get pod -o wide
NAME                                READY   STATUS    RESTARTS   AGE     IP            NODE          NOMINATED NODE   READINESS GATES
first-deployment-5554d585bd-d4hbb   1/1     Running   0          3m31s   10.244.1.52   k8s-worker1   <none>           <none>
brahim@Training:~/Lab4$
brahim@Training:~/Lab4$ kubectl logs first-deployment-5554d585bd-d4hbb
10.244.0.0 - - [27/May/2023:08:44:17 +0000] "GET / HTTP/1.1" 403 169 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/113
.0"
brahim@Training:~/Lab4$
```

◦ Créez le fichier *index.html* sous */website* et y ajouter un contenu (votre nom par exemple), puis rafraîchir l'interface web du navigateur.

```
brahim@Training:~/Lab4$ kubectl get pod -o wide
NAME                                READY   STATUS    RESTARTS   AGE     IP            NODE          NOMINATED NODE   READINESS GATES
first-deployment-5554d585bd-d4hbb   1/1     Running   0          6m16s   10.244.1.52   k8s-worker1   <none>           <none>
brahim@Training:~/Lab4$
brahim@Training:~/Lab4$ ssh vagrant@192.168.205.101
vagrant@192.168.205.101's password:
Last login: Sat May 27 08:48:04 2023 from 192.168.205.1
vagrant@k8s-worker1:~$
vagrant@k8s-worker1:~$ sudo su
root@k8s-worker1:/home/vagrant# echo '<b> Brahim HAMDI </b>' > /website/index.html
root@k8s-worker1:/home/vagrant#
root@k8s-worker1:/home/vagrant#
exit
vagrant@k8s-worker1:~$
logout
Connection to 192.168.205.101 closed.
brahim@Training:~/Lab4$
```