SUP DE VINCI
CFA DU SUPÉRIEUR DE L'OUEST PARISIEN

Kubernetes

Mastère DevOps

Nicolas BISSIERES

Intervenant : Samir LAKHDARI

Année : 2021

Table des matières

Rappel des consignes

Producteur

1. Créer un daemonSet nommé producteur basé sur l'image alpine

2. Créer un volume

3. Monter le volume sur le conteneur

4. Ecrire dans le fichier index.html le nom du hostname et la date toutes les 60 secondes

Consommateur

1. Créer un deploiement nommé web (avec 3 replicas) basé sur l'image httpd

2. Monter le volume créé précédemment sur le chemin htdocs du serveur Apache

Service NodePort

1. Créer un service nommé web de type NodePort qui rassemble les pods du deploiement web

Utilisation

1. Depuis le poste de travail, exécuter toutes les minutes commande "curl node1:30000"

Producteur

Création d'un daemonSet producteur basé sur une image alpine

La première étape est de créer le fichier yaml **producteur.yml** dans mon espace de travail **K8work**.

Etant sous Windows 10 il me suffit de faire un clic droit et nouveau fichier.

Celui-ci sera un **daemonset** appelé **producteur** et basé sur une **image alpine 3.12** :

```yaml
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: producteur
spec:
  selector:
    matchLabels:
      app3: producteur
  template:
    metadata:
      labels:
        app3: producteur
    spec:
      containers:
      - name: producteur
        image: alpine:3.12
        command: ["sleep","3600"]
```

J'applique ensuite le fichier producteur.yml crée précédemment avec la commande :

kubectl apply -f producteur.yml

Il est à préciser que je travail sous l'environnement **minikube** installé avec le logiciel **chocolatey** comme montré lors de notre cours **Kubernetes**

Une fois le fichier appliqué, on visualise le résultat grâce à la commande

kubectl get ds,pods -o wide

```
PS C:\Users\bwene\K8Work> kubectl get ds,pods -o wide
NAME                          DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE   CONTAINERS   IMAGES         SELECTOR
daemonset.apps/producteur     1         1         1       1            1           <none>          47s   producteur   alpine:3.12    app3=producteur

NAME                   READY   STATUS    RESTARTS   AGE   IP           NODE       NOMINATED NODE   READINESS GATES
pod/producteur-sfbx5   1/1     Running   0          47s   172.17.0.3   minikube   <none>           <none>
PS C:\Users\bwene\K8Work>
```

Création du volume Web

Il faut maintenant crée **un volume** qui sera **utilisé par le producteur** que nous nommerons **volume-web**.

Notre volume sera comme ceci :

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
    name: volume-web
spec:
    storageClassName: manual
    capacity:
      storage: 200Mi
    accessModes:
    - ReadWriteOnce
    hostPath:
      path: /data/web
```

L'objectif étant de **conserver les données** même si le conteneur se voit supprimé le type du volume sera un **volume persistant** nommé **volume-web** avec une capacité de **200Mo** accessible en **ReadWriteOnce** depuis **le chemin /data/web .**

Comme d'habitude on applique notre fichier yml avec la commande **kubectl apply -f volume-web.yml**

Et on vérifie la bonne création du volume avec : **kubectl get pv**

```
PS C:\Users\bwene\K8Work> kubectl get pv
NAME         CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM   STORAGECLASS   REASON   AGE
volume-web   200Mi      RWO            Retain           Available           manual                  6m41s
PS C:\Users\bwene\K8Work>
```

Monter le volume dans le conteneur

On retourne sur notre producteur.yml afin de rajouter les lignes **volumeMounts** et **volumes** pour pouvoir monter le volume crée précédemment sur notre producteur.

```yaml
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: producteur
spec:
  selector:
    matchLabels:
      app3: producteur
  template:
    metadata:
      labels:
        app3: producteur
    spec:
      containers:
      - name: producteur
        image: alpine:3.12
        command: ["sleep","3600"]
        volumeMounts:
        - name: volume-web
          mountPath: /web
      volumes:
      - name: volume-web
        hostPath:
          path: /data/web
```

On n'oublie pas **d'appliquer la nouvelle configuration** de notre producteur.

```
PS C:\Users\bwene\K8Work> kubectl apply -f producteur.yml
daemonset.apps/producteur configured
PS C:\Users\bwene\K8Work>
```

Le producteur a bien redémarré :

```
PS C:\Users\bwene\K8Work> kubectl get ds,pods -o wide
NAME                          DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE   CONTAINERS   IMAGES         SELECTOR
daemonset.apps/producteur     1         1         1       1            1           <none>          16m   producteur   alpine:3.12    app3=producteur

NAME                    READY   STATUS    RESTARTS   AGE    IP           NODE       NOMINATED NODE   READINESS GATES
pod/producteur-prbgv    1/1     Running   0          117s   172.17.0.3   minikube   <none>           <none>
PS C:\Users\bwene\K8Work>
```

Nous allons maintenant nous connecter dessus pour vérifier que le dossier web apparait

kubectl exec -it producteur-prbgv -- sh

puis ls

```
/ # ls
bin   dev   etc   home  lib   media mnt   opt   proc  root  run   sbin  srv   sys   tmp   usr   var   web
/ #
```

Le dossier apparait bien, pour aller dedans on fait cd web puis touch index.html afin de pouvoir continuer la suite du tp

Ecrire dans le fichier index.html le nom du hostname et la date toutes les 60 secondes

Il faut maintenant **écrire** dans le fichier **index.html** le nom du **hostname** et la **date** toutes les **60 secondes**.
Pour cela **plusieurs méthodes** sont possibles comme la création du **tache cron** ou l'utilisation d'une **boucle while**. C'est **cette dernière solution** que je vais montrer n'ayant pas réussi a utiliser cron comme je le voulais.

Nous allons retourner sur notre fichier producteur.yml et modifier la ligne **command** et ajouter en dessous la ligne **args** ce qui va nous donner ceci :

```yaml
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: producteur
spec:
  selector:
    matchLabels:
      app3: producteur
  template:
    metadata:
      labels:
        app3: producteur
    spec:
      containers:
      - name: producteur
        image: alpine:3.12
        command: ["sh","-c"]
        args:
          - while true;
            do
              echo  "le Hostname est $(hostname) et la date est $(date)" > /web/index.html;
              sleep 60;
            done
        volumeMounts:
        - name: volume-web
          mountPath: /web
      volumes:
      - name: volume-web
        hostPath:
          path: /data/web
```

On **réapplique la configuration**, on se reconnecte dans le volume web (**attention vu qu'on réapplique une nouvelle configuration le nom du producteur change**) et on fait un **cat index.html**

```
PS C:\Users\bwene\K8Work> kubectl exec -it producteur-q21c9 -- sh
/ # ls
bin   dev   etc   home  lib   media mnt   opt   proc  root  run   sbin  srv   sys   tmp   usr   var   web
/ # cd web/
/web # cat index.html
le Hostname est producteur-q21c9 et la date est Sun Apr  4 10:41:54 UTC 2021
/web #
```

La boucle est bien opérationnelle.

Consommateur

Créer un déploiement nommé web (avec 3 réplicas) basé sur l'image httpd

Il faut maintenant crée le déploiement nommé web basé sur une image httpd (ici une 2.4 alpine) et ses 2 réplicas.

Le fichier deploiement_web aura le contenu suivant :

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: web
  name: web
spec:
  replicas: 2
  selector:
    matchLabels:
      app: web
  template:
    metadata:
      labels:
        app: web
    spec:
      containers:
      - image: httpd:2.4-alpine
        name: httpd
```

On peut voir notre déploiement grâce à la commande : **kubectl get deploy**

```
PS C:\Users\bwene\K8Work> kubectl get deploy
NAME   READY   UP-TO-DATE   AVAILABLE   AGE
web    2/2     2            2           3m52s
PS C:\Users\bwene\K8Work>
```

Ainsi que les pods et les replicas avec : **kubectl get pods**

```
PS C:\Users\bwene\K8Work> kubectl get pods
NAME                   READY   STATUS    RESTARTS   AGE
producteur-bxh9c       1/1     Running   0          43m
web-56bdb9f6fb-cjvfn   1/1     Running   0          3m58s
web-56bdb9f6fb-hzfsf   1/1     Running   0          3m58s
PS C:\Users\bwene\K8Work>
```

Monter le volume créé précédemment sur le chemin htdocs du serveur Apache

Il faut maintenant **modifier notre fichier deploiement_web.yml** comme ceci afin d'ajouter le volume crée précédemment.

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: web
  name: web
spec:
  replicas: 2
  selector:
    matchLabels:
      app: web
  template:
    metadata:
      labels:
        app: web
    spec:
      containers:
      - image: httpd:2.4-alpine
        name: httpd
        volumeMounts:
        - name: volume-web
          mountPath: /usr/local/apache2/htdocs
      volumes:
      - name: volume-web
        hostPath:
          path: /data/web
```

Comme on peut le voir celui-ci sera dans /usr/local/apache2/htdocs.

On n'oublie pas d'appliquer la nouvelle configuration :

kubectl apply -f deploiement_web.yml

De regarder le nom de nos pods :

```
PS C:\Users\bwene\K8Work> kubectl get pods
NAME                   READY   STATUS    RESTARTS   AGE
producteur-bxh9c       1/1     Running   0          54m
web-78959d84d7-fcbnr   1/1     Running   0          4m8s
web-78959d84d7-nc4gm   1/1     Running   0          4m6s
PS C:\Users\bwene\K8Work>
```

On se connecte sur un de nos replicas (ici web-78959d84d7-fcbnr), on se déplace dans l'arborescence **htdocs** et on affiche le **contenu du index.html du volume web**

```
PS C:\Users\bwene\K8Work> kubectl exec -it web-6944847668-5clbb -- sh
/usr/local/apache2 # ls
bin      build    cgi-bin  conf     error    htdocs   icons    include  logs     modules
/usr/local/apache2 # cd htdocs/
/usr/local/apache2/htdocs # cat index.html
le Hostname est producteur-q2lc9 et la date est Sun Apr  4 10:42:54 UTC 2021
/usr/local/apache2/htdocs #
```

Service NodePort

Créer un service nommé web de type NodePort qui rassemble les pods du deploiement web

Nous allons créer un fichier qui se nommera **service_nodeport.yml** avec le contenu suivant :

```yaml
apiVersion: v1
kind: Service
metadata:
  name: web-service
spec:
  selector:
    app: web
  type: NodePort
  ports:
    - port: 80
      targetPort: 80
      nodePort: 30000
```

On applique le fichier : **kubectl apply -f service_nodeport.yml**

On vérifie la bonne création avec : **kubectl get deploy,rs,pods,svc -o wide**

```
PS C:\Users\bwene\K8Work> kubectl get deploy,rs,pods,svc -o wide
NAME                     READY   UP-TO-DATE   AVAILABLE   AGE    CONTAINERS   IMAGES            SELECTOR
deployment.apps/web      2/2     2            2           102m   httpd        httpd:2.4-alpine  app=web

NAME                                DESIRED   CURRENT   READY   AGE    CONTAINERS   IMAGES            SELECTOR
replicaset.apps/web-56bdb9f6fb      0         0         0       102m   httpd        httpd:2.4-alpine  app=web,pod-template-hash=56bdb9f6fb
replicaset.apps/web-6944847668      2         2         2       26m    httpd        httpd:2.4-alpine  app=web,pod-template-hash=6944847668
replicaset.apps/web-78959d84d7      0         0         0       90m    httpd        httpd:2.4-alpine  app=web,pod-template-hash=78959d84d7

NAME                       READY   STATUS    RESTARTS   AGE     IP           NODE       NOMINATED NODE   READINESS GATES
pod/producteur-q21c9       1/1     Running   0          3m17s   172.17.0.3   minikube   <none>           <none>
pod/web-6944847668-5clbb   1/1     Running   0          26m     172.17.0.5   minikube   <none>           <none>
pod/web-6944847668-qssrk   1/1     Running   0          26m     172.17.0.4   minikube   <none>           <none>

NAME                  TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE     SELECTOR
service/kubernetes    ClusterIP   10.96.0.1        <none>        443/TCP        3h27m   <none>
service/web-service   NodePort    10.104.139.115   <none>        80:30000/TCP   44m     app=web
PS C:\Users\bwene\K8Work>
```

On peut vérifier la bonne liaison de nos 2 replica a notre service **nodeport dans le champ Endpoints** grâce a à la commande : **kubectl describe service web**

```
PS C:\Users\bwene\K8Work> kubectl describe service web
Name:                     web-service
Namespace:                default
Labels:                   <none>
Annotations:              <none>
Selector:                 app=web
Type:                     NodePort
IP Families:              <none>
IP:                       10.104.139.115
IPs:                      10.104.139.115
Port:                     <unset>  80/TCP
TargetPort:               80/TCP
NodePort:                 <unset>  30000/TCP
Endpoints:                172.17.0.6:80,172.17.0.7:80
Session Affinity:         None
External Traffic Policy:  Cluster
Events:                   <none>
```

Utilisation

Depuis le poste de travail, exécuter toutes les minutes la commande "curl node1:30000"

Pour que le NodePort soit accessible depuis mon poste de travail il faut exécuter la commande **kubectl port-forward service/web-service 30000:80** dans un premier terminal

```
PS C:\Users\bwene\K8Work> kubectl port-forward service/web-service 30000:80
Forwarding from 127.0.0.1:30000 -> 80
Forwarding from [::1]:30000 -> 80
Handling connection for 30000
```

On crée ensuite un script PowerShell appelé **boucle_curl.ps1** avec le contenu suivant afin de pouvoir faire une requête curl toutes les minutes :

```
while (curl 127.0.0.1:30000)

    {
    curl 127.0.0.1:30000
    sleep 60
     }
```

On exécute le script et on constate son bon fonctionnement !

```
StatusCode        : 200
StatusDescription : OK
Content           : le Hostname est producteur-q21c9 et la date est Sun Apr  4 10:44:54 UTC 2021

RawContent        : HTTP/1.1 200 OK
                    Accept-Ranges: bytes
                    Content-Length: 77
                    Content-Type: text/html
                    Date: Sun, 04 Apr 2021 10:44:55 GMT
                    ETag: W/"4d-5bf234582f460"
                    Last-Modified: Sun, 04 Apr 2021 10:44:54 GMT
                    Serve...
Forms             : {}
Headers           : {[Accept-Ranges, bytes], [Content-Length, 77], [Content-Type, text/html], [Date, Sun, 04 Apr 2021 10:44:55 GMT]...}
Images            : {}
InputFields       : {}
Links             : {}
ParsedHtml        : mshtml.HTMLDocumentClass
RawContentLength  : 77
```