Lancement de l'application wordpress

Exercice

Dans cet exercice vous allez créer un Pod contenant 2 containers permettant de lancer une application wordpress.

1. Création de la spécification

Créez un fichier yaml *wordpress_pod.yaml* définissant un Pod ayant les propriétés suivantes:

- nom du Pod: *wp*
- un premier container:
  - nommé *wordpress*
  - basé sur l'image *wordpress:4.9-apache*
  - définissant la variable d'environnement *WORDPRESS_DB_PASSWORD* avec pour valeur *mysqlpwd* (cf note)
  - définissant la variable d'environnement *WORDPRESS_DB_HOST* avec pour valeur *127.0.0.1* (cf note)
- un second container:
  - nommé *mysql*
  - basé sur l'image *mysql:5.7*
  - définissant la variable d'environnement *MYSQL_ROOT_PASSWORD* avec pour valeur *mysqlpwd* (cf note)

Note: chaque container peut définir une clé *env*, celui contenant une liste de paires name / value

2. Lancement du Pod

Lancez le Pod à l'aide de *kubectl*

3. Vérification du status du Pod

Utilisez *kubectl* pour vérifier l'état du Pod.

Au bout de quelques secondes, il devrait être dans l'état *Running* (le temps que les images des containers soient téléchargées depuis le DockerHub).

4. Accès à l'application

Forwardez le port *80* du container *wordpress* sur le port *8080* de la machine hôte.

Lancez un navigateur sur http://localhost:8080

En ouvrant un navigateur sur l'URL indiquée, vous obtiendrez l'interface web de setup de *Wordpress*.

5. Suppression du Pod

A l'aide de *kubectl* supprimez le Pod *wp*.

Correction

1. Création de la spécification

La spécification, définie dans le fichier *wordpress_pod.yaml*, est la suivante:

```
apiVersion: v1
kind: Pod
metadata:
  name: wp
spec:
  containers:
  - image: wordpress:4.9-apache
    name: wordpress
    env:
    - name: WORDPRESS_DB_PASSWORD
      value: mysqlpwd
    - name: WORDPRESS_DB_HOST
      value: 127.0.0.1
  - image: mysql:5.7
    name: mysql
    env:
    - name: MYSQL_ROOT_PASSWORD
      value: mysqlpwd
```

Note: le Pod défini par la spécification ci-dessus ne permet pas de découpler les données gérées par le container *mysql* avec le cycle de vie de ce même container.
Comme nous le verrons un peu plus loin dans ce cours, nous pourrions définir un volume dans

la spécification du Pod et le monter dans le container *mysql* comme cela est illustré dans la spécification ci-dessous.

```
apiVersion: v1
kind: Pod
metadata:
  name: wp
spec:
  containers:
  - image: wordpress:4.9-apache
    name: wordpress
    env:
    - name: WORDPRESS_DB_PASSWORD
      value: mysqlpwd
    - name: WORDPRESS_DB_HOST
      value: 127.0.0.1
  - image: mysql:5.7
    name: mysql
    env:
    - name: MYSQL_ROOT_PASSWORD
      value: mysqlpwd
    volumeMounts:
    - name: data
      mountPath: /var/lib/mysql
  volumes:
  - name: data
    emptyDir: {}
```

2. Lancement du Pod

Le Pod peut être lancé avec la commande suivante:

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

3. Vérification du status du Pod

La commande suivante permet de voir l'état du Pod *wp*

```
$ kubectl get po/wp
```

Vous dévriez obtenir un Pod dans l'état *ContainerCreating* pendant quelques secondes, le temps que les images des containers soient téléchargées du DockerHub.

```
$ kubectl get po/wp
NAME      READY     STATUS              RESTARTS   AGE
wp        0/2       ContainerCreating   0          49s
```

Rapidement, le Pod devrait apparaitre avec le status *Running*

```
$ kubectl get pod/wp
NAME      READY     STATUS    RESTARTS   AGE
wp        2/2       Running   0          2m
```

4. Accès à l'application

Le container *wordpress* tournant sur le port *80*, la commande suivante permet de forwarder ce port sur le port *8080* de l'hôte.

```
$ kubectl port-forward wp 8080:80
Forwarding from 127.0.0.1:8080 -> 80
```

5. Suppression du Pod

Le Pod peut être supprimé avec la commande suivante:

```
$ kubectl delete po/wp
```