Exercice

Dans cet exercice, vous allez créer un Deployment et effectuer un rolling update.

1. Création d'un Deployment

A l'aide de la commande `kubectl run`, créez un Deployment avec les propriétés suivantes:

- nom: *www*
- 3 replicas d'un Pod avec un container basé sur nginx:1.12

Spécifiez l'option *--record=true* à la fin de la commande afin de conserver l'historique des commandes de mises à jour du Deployment.

2. Liste des ressources

Listez les ressources créées par la commande précédente (Deployment, ReplicaSet, Pod).

Note: utilisez une seule fois la commande *kubectl* pour lister l'ensemble des ressources.

3. Mise à jour de l'image

Mettez l'image nginx à jour avec la version 1.14-alpine

4. Liste des ressources

Une nouvelle fois, listes les ressource.

Que constatez vous ?

5. Historique des mises à jour

Listez les mises à jour (= révisions) du Deployment.

Note: utilisez la commande `kubectl rollout...`

Correction

1. Création d'un Deployment

Le Deployment peut être lancé avec la commande suivante:

```
$ kubectl run www --image nginx:1.12 --replicas 3 --record=true
deployment "www" created
```

2. Liste des ressources

La commande suivante permet de lister Les Deployment, ReplicaSet et Pod.

On utilise les raccourcis suivants:

- Deployment => deploy
- ReplicaSet => rs
- Pod => po

```
$ kubectl get deploy,rs,pod
NAME         DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
deploy/www   3         3         3            3           7s

NAME               DESIRED   CURRENT   READY     AGE
rs/www-cd74c7888   3         3         3         7s

NAME                     READY     STATUS    RESTARTS   AGE
po/www-cd74c7888-fjfmr   1/1       Running   0          7s
po/www-cd74c7888-m7db4   1/1       Running   0          7s
po/www-cd74c7888-qxpdl   1/1       Running   0          7s
```

La commande de la première question, à créé:

- 1 Deployment
- 1 ReplicaSet
- 3 Pods

Le ReplicaSet assure que les 3 Pods sont actifs.

3. Mise à jour de l'image

La commande suivante permet de mettre à jour l'image avec la version *nginx:1.14-alpine*.

```
$ kubectl set image deploy/www www=nginx:1.14-alpine
deployment "www" image updated
```

Note: lorsque nous avons créé le Deployment avec la commande `kubectl run`, nous n'avons pas utilisé de spécification détaillée et n'avons donc pas donné un nom au container du Pod. Cependant, le nom *www* que nous avons donné au Deployment a automatiquement été utilisé pour le nom du container. C'est donc le nom de ce container qui est utilisé dans la partie `www=nginx:1.14-alpine` de la commande ci dessus.

4. Liste des ressources

Nous utilisons la même commande que dans la question 2:

```
$ kubectl get deploy,rs,pod
NAME         DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
deploy/www   3         3         3            3           50s

NAME                DESIRED   CURRENT   READY     AGE
rs/www-6b5dfc4699   3         3         3         19s
rs/www-cd74c7888    0         0         0         50s

NAME                        READY     STATUS    RESTARTS   AGE
po/www-6b5dfc4699-krfl2     1/1       Running   0          19s
po/www-6b5dfc4699-w2p7l     1/1       Running   0          18s
po/www-6b5dfc4699-xwqxq     1/1       Running   0          19s
```

Nous pouvons voir ici qu'il y a maintenant 2 ReplicaSet:

- un pour la gestion des Pods utilisant l'image *nginx:1.12*. Il n'est plus actif, comme le montre la valeur *0* des champs *DESIRED*, *CURRENT* et *READY* relatifs aux Pods gérés par le ReplicaSet
- un second qui a été créé lors de la mise à jour de l'image, il gère 3 Pods, chacun ayant un container basé sur l'image *nginx:1.14-alpine*

5. Historique des mises à jour

La commande suivante permet de voir les différentes mises à jour du Deployment et les commandes associées:

```
$ kubectl rollout history deploy/www
deployments "www"
REVISION  CHANGE-CAUSE
```

```
kubectl run www --image=nginx:1.12 --replicas=3 --record=true
kubectl set image deploy/www www=nginx:1.14-alpine
```