GÉRER ET MANIPULER LES REPLICASETS KUBERNETES

Introduction

Dans le chapitre précédent nous avions étudié la plus petite unité de Kubernetes, à savoir les Pods. C'est bien joli tout ça, mais si vous souhaitez allouer plus de puissance à votre Pod, comment allez-vous vous y prendre ?

Une des solutions reste de scaler votre Pod. Mais pensez-vous que nous allons exécuter la commande permettant de créer un Pod autant de fois que le nombre de répliques voulues ?

La réponse est évidemment **NON**. Dans ce cas, nous utiliserons un objet Kubernetes, prévue à cet effet, soit les `ReplicaSets`.

Les `ReplicaSets` garantissent que le nombre de répliques souhaitées de votre Pod, est bel et bien respecté. Il peut être considéré comme un remplacement d'un autre objet Kubernetes nommé le `replication controller` (en FR: "contrôleur de réplication").

La principale **différence entre le ReplicaSets et le replication controller**, est que le `ReplicaSet` prend en charge le sélecteur basé sur sur les labels, ce qui n'est pas le cas avec le `replication controller`.

Manipulation des ReplicaSet

Création du ReplicaSet

Dans cet exemple, nous allons **créer un ReplicaSet contenant trois répliques** du Pod `nginx`. Comme d'habitude pour créer un Pod, nous passerons par la création d'un template au format YAML, ce qui nous donne ;

```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: nginx-rs
  labels:
    app: web
spec:
  replicas: 3
  selector:
    matchLabels:
      app: web
  template:
    metadata:
      labels:
        app: web
    spec:
      containers:
      - name: nginx
        image: nginx
```

Passons à l'explication de ces quelques nouvelles lignes de notre template :

```
apiVersion: apps/v1
```

Dans ce template nous spécifions la version `apps/v1` qui est la version de l'API k8s qui prend en charge le concept des **`ReplicasSets`**.

```
replicas: 3
selector:
  matchLabels:
    app: web
```

Ici, nous indiquons le nombre de répliques voulues, soit trois répliques. Et nous définissons aussi la condition de correspondance qui consiste à répliquer les pods ayant le label `app: web`.

Par la suite, on ne fait que définir la configuration de notre Pod, comme vu dans le chapitre précédent dédié aux Pods.

Une fois le template fini, on peut commencer à **créer notre ReplicaSet avec la commande suivante** :

```
kubectl create -f nginx-rs.yaml
```

Vérifons ensuite la liste des pods disponibles dans notre cluster :

```
kubectl get pods -o wide
```

Résultat :

```
NAME             READY   STATUS    RESTARTS   AGE     IP              NODE      NOMINAT
nginx-rs-f49kh   1/1     Running   0          7m25s   192.168.1.44    worker-1   <none>
nginx-rs-sn9v9   1/1     Running   0          7m25s   192.168.1.42    worker-1   <none>
nginx-rs-xk5g7   1/1     Running   0          7m25s   192.168.1.43    worker-1   <none>
```

D'après le résultat, nous avons bien nos trois répliques du Pod `nginx` avec une adresse IP spécifique à chaque Pod.

Nous pouvons dès lors, accéder à la page d'accueil de nginx depuis ces 3 IPS :

```
ssh vagrant@worker-1 curl -s  http://192.168.1.44
ssh vagrant@worker-1 curl -s  http://192.168.1.42
ssh vagrant@worker-1 curl -s  http://192.168.1.43
```

Résultat :

```
<title>Welcome to nginx!</title>
...
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>
```

```
<p><em>Thank you for using nginx.</em></p>
```

Scaler vos Pods

Il existe différentes manières pour scaler vos Pods. Vous pouvez par exemple **scaler les répliques de vos Pods sans changer votre template YAML** avec les commandes suivantes :

```
kubectl scale --replicas=4 replicaset nginx-rs
```

ou

```
kubectl scale --replicas=4 -f nginx-rs.yaml
```

Après avoir exécuté la commande de scaling , nous allons cette fois-ci, **vérifier la liste des ReplicaSets disponibles dans notre cluster k8s** :

```
kubectl get replicasets
```

ou

```
kubectl get rs
```

Résultat :

```
NAME       DESIRED   CURRENT   READY   AGE
nginx-rs   4         4         4       21m
```

Le nombre de répliques du Pod `nginx` est bien passé de trois à quatre.

Contrairement à la première méthode, celle prochaine commande permet d'un côté de **diminuer le nombre de répliques de notre Pod**, mais nécessite un changement depuis notre template YAML.

Depuis notre template, je vais changer la ligne `replicas: 3` en `replicas: 2`, et lancer la

commande suivante :

```
kubectl apply -f nginx-pod.yaml
```

Si on revérifie la liste de nos ReplicaSet, nous avons bien plus que deux Pods :

```
kubectl apply -f nginx-pod.yaml
```

Résultat :

```
NAME       DESIRED   CURRENT   READY   AGE
nginx-rs   2         2         2       28m
```

Information

Si jamais vous avez perdu votre template, vous pouvez alors utiliser la commande `kubectl edit rs nginx-rs`, ce qui aura pour effet d'ouvrir l'éditeur de texte de votre terminal contenant la configuration de votre ReplicaSet au format YAML. Pour chaque modification et sauvegarde de ce fichier, les changements seront immédiatement appliqués sur votre ReplicaSet.

Supprimer votre ReplicaSet

Voici la commande qui permet de **supprimer votre ReplicaSet** :

```
kubectl delete ReplicaSet nginx-rs
```

ou

```
kubectl delete rs nginx-rs
```

Résultat :

```
replicaset.extensions "nginx-rs" deleted
```

Récupérer des informations sur votre ReplicaSet

Comme pour les Pods, vous pouvez **récupérer des informations détaillées sur votre ReplicaSet**. Plus exactement, ce qui va nous intéresser le plus, ce sont les différents événements subis par notre ReplicaSet, soit la commande suivante :

```
kubectl describe rs nginx-rs
```

Résultat :

```
Name:         nginx-rs
...
    Events:
  Type    Reason            Age   From                   Message
  ----    ------            ----  ----                   -------
  Normal  SuccessfulCreate  28s   replicaset-controller  Created pod
  Normal  SuccessfulCreate  28s   replicaset-controller  Created pod
```

Conclusion

Les `ReplicaSets` sont un bon moyen, pour créer et gérer les répliques de vos Pods. Dans le chapitre suivant nos verrons un autre type d'objet Kubernetes, les `Deployments`.

Comme à mon habitude, voici un **récapitulatif des commandes de manipulation des ReplicaSet**

```
# Afficher la liste des Replicasets
kubectl get replicasets [En option <REPLICASET NAME>]
    --template : récupérer des informations précises de la sortie de la commande
        Ex: Récupérer le nombre de réplique : kubectl get rs  <REPLICASET NAME> --temp

# Créer un ReplicaSet
kubectl create -f <template.yaml>

# Supprimer un ReplicaSet
```

```
kubectl delete replicasets <REPLICASET NAME>

# Appliquer des nouveaux changements à votre ReplicaSet sans le détruire
kubectl apply -f <template.yaml>

# Modifier et appliquer les changements de votre ReplicaSet instantanément sans le dét
kubectl edit rs <REPLICASET NAME>

# Afficher les détails d'un replicaset
kubectl describe replicasets <REPLICASET NAME>
```