SAUVEGARDER ET RESTAURER VOTRE CLUSTER KUBERNETES

Introduction

Dans ce chapitre nous examinerons les **différentes méthodologies de sauvegarde et de restauration de votre cluster Kubernetes**. Jusqu'à présent, nous avons déployé différentes ressources Kubernetes à l'aide de la commande `kubectl` mais aussi grâce aux différents fichiers manifest au format YAML.

Il existe **différentes façons pour sauvegarder et restaurer un cluster Kubernetes**, chaque méthode comporte ses propres avantages et inconvénients.

Nous avons tout d'abord l'approche déclarative qui consiste à créer et à sauvegarder toutes vos ressources Kubernetes depuis un fichier manifest au format YAML, et par la suite exécuter la commande `kubectl apply -f` ou `kubectl create -f`. C'est la méthode classique, vous pouvez en plus de ça, sauvegarder vos fichiers de configuration directement dans votre propre outil sauvegarde, dans ce cas vous pouvez facilement les **réutiliser à un moment ultérieur** ou les partager avec d'autres personnes. Vous aurez donc une copie de ces fichiers enregistrés à tout moment.

Information

Une autre bonne pratique consiste à **commiter** vos fichiers Manifest, afin de revenir plus facilement sur une version précédente mais aussi redéployer vos applications sur le cluster en exécutant simplement ces fichiers de configuration.

Cependant, il se peut que des membres de votre équipe créent un objet Kubernetes de manière impérative sans documenter cette information, en utilisant directement la commande `kubectl create <object name>`. Cette méthode ne garantit pas donc une sauvegarde complète de toutes vos données de votre cluster Kubernetes.

ETCD

Description

Dans ce cas, la **meilleure solution** consiste à utiliser le composant ETCD. Pour rappel ETCD est un composant du master Kubernetes qui est l'endroit où toutes les informations relatives au cluster sont stockées. Dont, des informations sur le cluster lui-même, ainsi que les nœuds et toutes les autres ressources créées dans le cluster.

Comme nous l'avions vu au début de notre cours, le composant ETCD est hébergé sur les nœuds maîtres. Lors de la configuration d'ETCD, un emplacement est spécifié par défaut où toutes les données k8s seraient stockées. Vous retrouverez cet emplacement à l'aide de la commande suivante :

```
kubectl describe pod etcd-master -n kube-system | grep data-dir
```

Résultat :

```
--data-dir=/var/lib/etcd
```

Backup

ETCD est également livré avec une **solution de Snapshot intégrée**, que nous utiliserons ainsi pour créer notre première backup. Pour cela, nous devons

récupérer quelques informations.

Premièrement on récupère, l'adresse et le port utilisé par le master pour communiquer avec le composant ETCD. Pour ce faire, lancez donc la commande suivante :

```
kubectl describe pod etcd-master -n kube-system | grep  listen-client-urls
```

Résultat :

```
--listen-client-urls=https://127.0.0.1:2379
```

Deuxièmement, on récupère le fichier de certificat du serveur ETCD :

```
kubectl describe pod etcd-master -n kube-system | grep cert-file
```

Résultat :

```
--cert-file=/etc/kubernetes/pki/etcd/server.crt
```

Enfin, on récupère, la location du fichier CA (Autorité de Certification) du serveur ETCD :

```
kubectl describe pod etcd-master -n kube-system | grep trusted-ca
```

Résultat :

```
--trusted-ca-file=/etc/kubernetes/pki/etcd/ca.crt
```

Une fois toutes ces informations récupérées, on peut commencer à utiliser l'**outil de backup ETCD** à savoir `ETCDCTL_API` afin de créer la première backup de notre cluster Kubernetes :

```
ETCDCTL_API=3 etcdctl --endpoints=https://[127.0.0.1]:2379 --cacert=/etc/kubernetes/p
    --cert=/etc/kubernetes/pki/etcd/server.crt --key=/etc/kubernetes/pki/etcd/server
```

```
    snapshot save /tmp/snapshot-etcd.db
```

Dans la commande, nous précisons que notre backup sera sauvegardée sous un fichier de base de données sous le nom de **snapshot-etcd.db** qui sera placé dans le dossier **/tmp/**. Ce fichier est très important car nous le réutiliserons plus tard pour la restauration de notre cluster Kubernetes.

Réstauration

Afin de restaurer notre sauvegarde, Nous devons tout d'abord initialiser un nouveau répertoire de restauration de notre base de données et par la suite modifier notre configuration de ETCD en changeant l'emplacement où le composant cherche sa nouvelle base de données.

Nous allons commencer par générer notre nouveau répertoire de sauvegarde en se basant sur notre fichier de sauvegarde toujours en utilisant l'outil ETCDCTL_API :

```
ETCDCTL_API=3 etcdctl --endpoints=https://[127.0.0.1]:2379 --cacert=/etc/kubernetes/p
    --cert=/etc/kubernetes/pki/etcd/server.crt --key=/etc/kubernetes/pki/etcd/server
    --data-dir /var/lib/etcd-backup \
    snapshot restore /tmp/snapshot-etcd.db
```

En exécutant cette commande, un nouveau répertoire de données est créé à l'emplacement **/var/lib/etcd-backup** qui contient toutes les données de notre cluster sauvegardées précédemment.

L'étape suivante consiste à mette à jour la configuration du pod statique ETCD pour utiliser le nouveau répertoire de données en modifiant le fichier **/etc/kubernetes/manifests/etcd.yaml**. Il faut mettre à jour la valeur de `--data-dir` et d'y insérer notre nouveau chemin de sauvegarde, dans notre cas ça nous donnera :

Information

Les pods statiques sont une sorte de pod qui ne sont pas gérés via le composant kube-apiserver et sont directement liés au démon kubelet sur un nœud spécifique. C'est le kubelet qui crée et gère le cycle de vie de ce type de pods. Pour retrouver l'emplacement des pods statiques, il suffit de lancer la commande `grep -i static /var/lib/kubelet/config.yaml`. Une fois le path récupéré, vous y trouverez tous les fichiers manifests YAML de vos statiques pods (vous pouvez d'ailleurs créer votre propre statique pod en déposant votre fichier yaml dans ce répertoire).

```
--data-dir=/var/lib/etcd-backup
```

Enfin, toujours dans le même fichier, il ne faut pas oublier de modifier les volumes afin qu'ils pointent vers le nouveau path :

```
volumeMounts:
- mountPath: /var/lib/etcd-backup
  name: etcd-data
- mountPath: /etc/kubernetes/pki/etcd
  name: etcd-certs
hostNetwork: true
priorityClassName: system-cluster-critical
volumes:
- hostPath:
  path: /var/lib/etcd-backup
  type: DirectoryOrCreate
name: etcd-data
- hostPath:
  path: /etc/kubernetes/pki/etcd
  type: DirectoryOrCreate
name: etcd-certs
```

Vous n'avez pas besoin de lancer d'autres commandes, puisque les statiques pods sont automatiquement mis à jour par le démon kubelet.