Chapitre 4
Automatisation et publication d'une application

1. Objectifs du chapitre et prérequis

Le chapitre précédent a constitué une introduction à l'utilisation de Kubernetes avec notamment le recours au dashboard pour le déploiement d'une application.

Ce chapitre va reprendre cet exercice. La différence se fera sur le mode opératoire puisque les opérations seront réalisées en ligne de commande.

2. Gestion par kubectl d'une application

2.1 Suppression d'un déploiement

Durant le précédent chapitre, l'application MailHog a été déployée à l'aide du dashboard. Même si vous ne l'avez pas fait, vous pouvez néanmoins suivre les instructions qui suivent.

Premier point, récupérer la liste des déploiements présents dans le serveur. Pour cela, il faut lancer la commande `kubectl` suivie du mot-clé `get` avec le type d'objet `deployment`. Ci-dessous la commande correspondante :

```
$ kubectl get deployment
```

Ci-dessous le résultat renvoyé :

```
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
mailhog   1/1     1            1           8d
```

La suppression du déploiement se fait à l'aide de la commande `kubectl` suivie des éléments suivants :

– Le mot-clé `delete`.
– Le type d'objet à supprimer (`deployment`).
– Le nom de l'objet à supprimer (`mailHog`).

Ci-dessous la commande à lancer :

```
$ kubectl delete deployment mailhog
```

Ci-dessous le résultat renvoyé par cette commande :

```
deployment.extensions "mailhog" deleted
```

La consultation de la liste des pods renverra alors le contenu suivant :

```
NAME                       READY   STATUS        RESTARTS   AGE
mailhog-69bd8f74cb-kl9p5   1/1     Terminating   2          8d
```

Au bout de quelques instants, la commande devrait renvoyer le message suivant :

```
No resources found.
```

2.2 Création d'un déploiement

La commande `kubectl`, outre les opérations de consultation et de suppression, prend en charge la création d'objets. Dans le cas d'un déploiement, la commande doit être lancée avec les options suivantes :

– Le mot-clé `create`.
– Le type d'objet à créer : `deployment` (ou le raccourci `deploy`).
– Le nom du déploiement (`mailhog`).
– Le nom de l'image à déployer avec l'option `--image`.

Ci-dessous la commande correspondant à ces indications :

```
$ kubectl create deployment mailhog --image=mailhog/mailhog
```

Ci-dessous le résultat de cette commande :

```
deployment.apps/mailhog created
```

2.3 État du déploiement

Une fois créé, le déploiement est consultable à l'aide de la commande `kubectl` suivie des options suivantes :

– L'option `get`.

– Le type d'objet : `deployment` (ou `deploy`);

– Le nom de l'objet : `mailhog`.

La commande à lancer :

```
$ kubectl get deployment
```

Le résultat renvoyé :

```
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
mailhog   1/1     1            1           3m
```

L'ajout de l'option `-o wide` permettra de renvoyer des champs supplémentaires. Ci-dessous les informations supplémentaires renvoyées :

– `CONTAINERS` : liste des containers.

– `IMAGES` : liste des images utilisées.

– `SELECTOR` : le label de sélection des pods.

En plus de l'option `get`, la commande `kubectl` propose l'option `describe`. Cette option permet de récupérer tout un ensemble d'informations sur les caractéristiques d'un objet Kubernetes. L'instruction `describe` prend en charge les mêmes options que `get`, à savoir un type d'objet ainsi qu'un nom d'objet.

Ci-dessous la commande à utiliser pour consulter l'état du déploiement de MailHog :

```
kubectl describe deployment mailhog
```

Ci-dessous le résultat renvoyé par cette commande :

```
Name:                   mailhog
Namespace:              default
CreationTimestamp:      Sun, 31 Mar 2019 14:54:03 +0200
Labels:                 app=mailhog
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               app=mailhog
Replicas:               1 desired | 1 updated | 1 total |
1 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
   Labels:  app=mailhog
   Containers:
     mailhog:
       Image:        mailhog/mailhog
       Port:         <none>
       Host Port:    <none>
       Environment:  <none>
       Mounts:       <none>
   Volumes:        <none>
Conditions:
   Type           Status  Reason
   ----           ------  ------
   Available      True    MinimumReplicasAvailable
   Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  <none>
NewReplicaSet:   mailhog-5dbc9c86c4 (1/1 replicas created)
Events:
  Type    ... Message
  ----    ... -------
  Normal  ... Scaled up replica set mailhog-5789447cbf to 1
```

Cette commande renvoie de nombreuses informations parmi lesquelles :

– le nom et l'espace de noms du déploiement,
– la date de création,
– les labels et annotations,
– le mécanisme de sélecteur,
– le nombre de réplicats de l'application (le nombre de pods),
– le mécanisme de mise à jour (ici RollingUpdate),
– le template permettant la création des pods associés,
– l'état du déploiement (conditions),
– le réplicat actuel ainsi que les anciens réplicats,
– les événements autour de la création de ce déploiement (ici le passage à un réplicat sur le déploiement de MailHog).

2.4 Mécanisme des réplicats

2.4.1 Consultation des réplicats

Parmi les informations remontées par la commande `kubectl describe` se trouve le ReplicaSet actuel ainsi que – en cas de mise à jour – la liste des anciens ReplicaSet.

Cet objet va prendre en charge la création d'un nombre donné de réplicats pour une application donnée. C'est lui qui va indiquer au cluster Kubernetes qu'il faut redémarrer un pod en cas de suppression d'un pod.

Il va également être associé à l'ensemble des caractéristiques d'une application à un instant donné, comme par exemple :

– les caractéristiques d'une image (numéro de version, emplacement, nom),
– la réservation d'une quantité de mémoire et CPU,
– certaines caractéristiques du déploiement (variables d'environnement).

En cas de mise à jour de l'objet Deployment, un nouvel objet ReplicaSet sera créé automatiquement et le ReplicaSet précédent sera ajouté à la liste des anciens objets ReplicaSet.

Remarque

Le mécanisme de retour arrière de Kubernetes s'appuie sur ces objets.

Tout comme pour les déploiements, il est possible de consulter les réplicats à l'aide de la commande `kubectl` et de l'option `get` suivie du type à consulter (`replicaset` ou son raccourci `rs`). Ci-dessous la commande correspondante :

```
$ kubectl get replicaset
```

Ci-dessous le résultat de cette commande :

```
NAME                  DESIRED   CURRENT   READY   AGE
mailhog-5dbc9c86c4    1         1         1       5h33m
```

Le nom du réplicat est constitué d'une partie fixe (reprenant le déploiement duquel il descend) suivie d'une partie aléatoire.

2.4.2 Description des réplicats

Tout comme pour un objet Deployment, il est possible de décrire un objet ReplicaSet. Ci-dessous la commande à lancer dans le cas du réplicat de MailHog précédemment remonté :

```
$ kubectl describe rs mailhog-5dbc9c86c4
```

Ci-dessous la sortie renvoyée par cette commande :

```
Name:           mailhog-5dbc9c86c4
Namespace:      default
Selector:       app=mailhog,pod-template-hash=5dbc9c86c4
Labels:         app=mailhog
                pod-template-hash=5dbc9c86c4
Annotations:    deployment.kubernetes.io/desired-replicas: 1
                deployment.kubernetes.io/max-replicas: 2
                deployment.kubernetes.io/revision: 3
                deployment.kubernetes.io/revision-history: 1
Controlled By:  Deployment/mailhog
Replicas:       1 current / 1 desired
```

```
Pods Status:    1 Running / 0 Waiting / 0 Succeeded / 0 Failed
Pod Template:
   Labels:  app=mailhog
            pod-template-hash=5dbc9c86c4
   Containers:
     mailhog:
     Image:        mailhog/mailhog
     Port:         <none>
     Host Port:    <none>
     Environment:  <none>
     Mounts:       <none>
   Volumes:        <none>
Events:
   Type    Reason            Age   From                   Message
   ----    ------            ----  ----                   -------
   Normal  SuccessfulDelete  62m   replicaset-controller  Deleted
pod: mailhog-5dbc9c86c4-22lzr
   Normal  SuccessfulCreate  59m   replicaset-controller  Created
pod: mailhog-5dbc9c86c4-8sjcs
   Normal  SuccessfulDelete  59m   replicaset-controller  Deleted
pod: mailhog-5dbc9c86c4-8sjcs
   Normal  SuccessfulDelete  58m   replicaset-controller  Deleted
pod: mailhog-5dbc9c86c4-49n9v
   Normal  SuccessfulCreate  57m   replicaset-controller  Created
pod: mailhog-5dbc9c86c4-vdgj6
```

Les informations sont sensiblement les mêmes que pour un objet Deployment au niveau de l'en-tête. En revanche, la partie `Events` est plus verbeuse, avec toutes les opérations qui ont été réalisées sur les pods du déploiement de l'application MailHog.

Le champ `Events` peut être une source d'informations intéressante à consulter en cas d'anomalies lors d'un déploiement.