TP – HELM

Helm est le **gestionnaire de paquets** officiel de **Kubernetes**, écrit en Go, qui permet de gérer le cycle de vie d'une **application** au sein d'un **cluster Kubernetes**

Table des matières

Fonctionnement d'un gestionnaire de package

Pour rappel, un gestionnaire de package est un outil permettant gérer le cycle de vie des applications dans un système informatique. Ce cycle de vie comprend les phases d'installation, de mise à jour (upgrade), de retour arrière (downgrade) et de désinstallation d'une application. Helm est en quelque sorte le `yum` ou l'`apt` de Kubernetes.

Le fonctionnement d'un gestionnaire de paquets est simple. Tout d'abord, l'utilisateur passe le nom d'un progiciel en argument. Le gestionnaire de packages effectue une recherche dans un référentiel de packages pour voir si ce package existe. S'il est trouvé, le gestionnaire de packages installe l'application définie par le package et ses dépendances aux emplacements spécifiés sur le système.

Quelques définitions

Les Charts Helm

Un Chart est le package Helm. Il contient toutes les définitions des ressources nécessaires pour installer un objet à l'intérieur d'un cluster Kubernetes.

Un dépot Helm

Un `dépot` ou `repository` est l'endroit où les **charts** sont stockés et mis à disposition.

Une release Helm

Une Release est une instance d'un **chart Helm** installé sur un **cluster Kubernetes**. Un chart Helm peut être installé à plusieurs reprises sur le même **cluster Kubernetes**, mais devra avoir un release name différent.

Installation de Helm

L'installation est assez simple :

```
ludo@kubernetes:$ curl ttps://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3 | bash
```

Afficher l'aide de Helm

La cli d'Helm est très bien documentée. Pour afficher l'aide il suffit d'ajouter `--help`. Ce fonctionne vaut pour toutes le commandes de Helm `helm <commande> --help` :

```
ludo@kubernetes:$ helm ls --help
This command lists all of the releases for a specified namespace (uses current
namespace context if namespace not specified).

By default, it lists only releases that are deployed or failed. Flags like
'--uninstalled' and '--all' will alter this behavior. Such flags can be combined:
'--uninstalled --failed'.

...
```

Installation de Package Helm

Gérer les repositories Helm

Une fois Helm installé vous ne possédez aucun repository configurer sur votre session. Vous pouvez ajouter le premier repo avec la commande `repo add` :

```
ludo@kubernetes:$ helm repo add stable https://charts.helm.sh/stable
"stable" has been added to your repositories
ludo@kubernetes:$ helm repo add bitnami https://charts.bitnami.com/bitnami
"bitnami" has been added to your repositories
```

Pour les repositories configuré sur votre session il suffit d'utiliser la commande suivante :

```
ludo@kubernetes:$ helm repo list
NAME    URL
stable   https://charts.helm.sh/stable
bitnami  https://charts.bitnami.com/bitnami
```

Pour trouver d'autres repos, par exemple celui fournissant l'ingress controller nginx, il suffit de se rendre sur le Hub Artifact.

Comme pour les repository de packages de systèmes d'exploitation il est possible de mettre à jour l'index. Cela permet de bénéficier des dernières versions de package

```
ludo@kubernetes:$ helm repo update
```

Recherche de package Helm

Il existe deux commandes qui permettent de rechercher des packages Helm dans les repository public : `helm search hub` et `helm search repo`.

La première commande lance une recherche sur l'Artifact Hub qui regroupe un grand nombre de repositories :

```
ludo@kubernetes:$ helm search hub loki | grep 2.4.2
URL                                                 CHART VERSION   APP VERSION
DESCRIPTION
https://artifacthub.io/packages/helm/wenerme/loki       2.9.0           v2.4.2
Loki: like Prometheus, but for logs.
https://artifacthub.io/packages/helm/grafana/loki       2.9.0           v2.4.2
Loki: like Prometheus, but for logs.
https://artifacthub.io/packages/helm/wenerme/lo...      0.40.0          2.4.2
Helm chart for Grafana Loki in microservices mode
https://artifacthub.io/packages/helm/grafana/lo...      0.40.0          2.4.2
Helm chart for Grafana Loki in microservices mode
https://artifacthub.io/packages/helm/grafana/lo...      0.2.0           2.4.2
Helm chart for Grafana Loki in simple, scalable...
```

La seconde commande recherche les charts dans les repos que vous avez configuré. La configuration d'Helm se trouve dans le répertoire $HOME/.config/helm.:

```
ludo@kubernetes:$ helm search repo mariadb
NAME                    CHART VERSION   APP VERSION     DESCRIPTION
bitnami/mariadb         12.2.4          10.11.3         MariaDB is an open source,
bitnami/mariadb-galera  8.2.4           10.11.3         MariaDB Galera is a multi-
bitnami/phpmyadmin      11.1.2          5.2.1           phpMyAdmin is a free …
```

Installation d'un package Helm

Pour installer un nouveau package, utilisez la commande `helm install`. Par exemple pour installer le chart nginx du repo bitnami:

```
ludo@kubernetes:$ helm repo add bitnami https://charts.bitnami.com/bitnami
ludo@kubernetes:$ helm repo update
ludo@kubernetes:$ helm search nginx
NAME                                    CHART VERSION   APP VERSION     DESCRIPTION
bitnami/nginx                           9.7.1           1.21.5          Chart for
the nginx server
ludo@kubernetes:$ kubectl create ns nginx
ludo@kubernetes:$ kubectl config set-context --current --namespace=nginx
ludo@kubernetes:$ helm install my-nginx bitnami/nginx
NAME: my-nginx
LAST DEPLOYED: Sat Jan 15 15:01:57 2022
NAMESPACE: nginx
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
CHART NAME: nginx
CHART VERSION: 9.7.1
APP VERSION: 1.21.5

** Please be patient while the chart is being deployed **

NGINX can be accessed through the following DNS name from within your cluster:

    nginx.nginx.svc.cluster.local (port 80)

To access NGINX from outside the cluster, follow the steps below:

1. Get the NGINX URL by running these commands:

  NOTE: It may take a few minutes for the LoadBalancer IP to be available.
        Watch the status with: 'kubectl get svc --namespace nginx -w nginx-1-21-5'

    export SERVICE_PORT=$(kubectl get --namespace nginx -o
jsonpath="{.spec.ports[0].port}" services nginx-1-21-5)
    export SERVICE_IP=$(kubectl get svc --namespace nginx nginx-1-21-5 -o
jsonpath='{.status.loadBalancer.ingress[0].ip}')
    echo http://${SERVICE_IP}:${SERVICE_PORT}

ludo@kubernetes:$ kubectl get svc
ludo@kubernetes:$ kubectl get svc --namespace nginx my-nginx
NAME       TYPE           CLUSTER-IP    EXTERNAL-IP   PORT(S)        AGE
my-nginx   LoadBalancer   10.106.2.64   10.0.0.191    80:30123/TCP   12s
```

Lister les releases installer

Pour lister les **releases** installées dans le namespace actuel de votre cluster Kubernetes, il faut utiliser la commande `list` ou `ls` :

```
ludo@kubernetes:$ helm ls -A
NAME            NAMESPACE       REVISION        UPDATED
STATUS          CHART           APP VERSION
my-nginx        nginx           1               2022-01-15 15:10:31.134375044 +0000
UTC deployed        nginx-9.7.1     1.21.5
```

La commande `ls` prends plein de paramètres permettant de les filtrer. Par exemple pour afficher toutes les **releases** installées sur tous les namespaces du cluster l'option -A sera parfaite.

Afficher le status d'une release

Maintenant que vous savez lister les **releases** présente au sein de votre **cluster**, vous pouvez demander à afficher le status de celui-ci :

```
ludo@kubernetes:$ helm status -n nginx my-nginx
NAME: my-nginx
LAST DEPLOYED: Sat Jan 15 15:10:31 2022
NAMESPACE: nginx
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
CHART NAME: nginx
CHART VERSION: 9.7.1
APP VERSION: 1.21.5

...
```

Vous avez remarqué qu'il s'agit de la même sortie de commande que celle lors de l'installation du chart Helm. On pourrait l'utiliser dans un programme en modifiant le type de sortie : `helm status my-nginx -o json` avec un petit `jq` derrière pour sortie une information en particulier.

Désinstallation d'une release

C'est assez simple il suffit d'utiliser la commande uninstall :

```
ludo@kubernetes:$ helm uninstall my-nginx
release "my-nginx" uninstalled
```

Personnaliser une Release

Par exemple, nous aimerions plutôt que le service de notre release my-nginx soit de Type `NodePort`. Pour cela il faut récupérer les values du chart et créer un fichier contenant ces valeurs :

```
ludo@kubernetes:$ helm show values bitnami/nginx > values-my-nginx.yaml
```

Éditez le et modifiez `type: LoadBalancer` en `type: NodePort`.

Nous allons installer à nouveau le chart mais en lui indiquant d'utiliser notre fichier de valeur :

```
ludo@kubernetes:$ helm install -f values-my-nginx.yaml my-nginx bitnami/nginx
NAME: my-nginx
LAST DEPLOYED: Sat Jan 15 15:40:48 2022
NAMESPACE: nginx
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
CHART NAME: nginx
CHART VERSION: 9.7.1
APP VERSION: 1.21.5

ludo@kubernetes:$ kubectl get svc my-nginx
NAME       TYPE       CLUSTER-IP     EXTERNAL-IP   PORT(S)        AGE
my-nginx   NodePort   10.98.52.185   <none>        80:30205/TCP   36s
```

Upgrade d'une release

Nous allons changer la version de l'image nginx. Éditez le fichier et modifier la ligne tag :

```
image:
  registry: docker.io
  repository: bitnami/nginx
  tag: 1.21.5-debian-10-r3
```

En

```
image:
  registry: docker.io
  repository: bitnami/nginx
  tag: 1.21.5-debian-10-r6
```

Appliquons nos modifications :

```
ludo@kubernetes:$ helm upgrade my-nginx bitnami/nginx -f values-my-nginx.yaml
Release "my-nginx" has been upgraded. Happy Helming!
NAME: my-nginx
LAST DEPLOYED: Sat Jan 15 15:47:31 2022
NAMESPACE: nginx
```

```
STATUS: deployed
REVISION: 3
TEST SUITE: None
NOTES:
CHART NAME: nginx
CHART VERSION: 9.7.1
APP VERSION: 1.21.5
```

Vérifions la version de l'image du pod :

```
kubectl describe pod my-nginx-d6d8b459f-h4lcc | grep Image:
    Image:          docker.io/bitnami/nginx:1.21.5-debian-10-r16CHART NAME: nginx
```

Afficher l'historique

Pour afficher l'historique de nos modifications sur notre release, il sufffit d'utiliser la commande `history` :

Vérifions la version de l'image du pod :

```
helm history my-nginx
REVISION        UPDATED                         STATUS          CHART           APP
VERSION     DESCRIPTION
1               Sat Jan 15 15:40:48 2022        superseded      nginx-9.7.1
1.21.5          Install complete
2               Sat Jan 15 15:46:40 2022        superseded      nginx-9.7.1
1.21.5          Upgrade complete
3               Sat Jan 15 15:47:31 2022        deployed        nginx-9.7.1
1.21.5          Upgrade complete
```

Rollback d'une release

Mince notre application ne fonctionne plus il faut vite restaurer la version précédente. Pour cela il suffit d'utiliser la commande `rollback` en indiquant la version désirée, l'index des versions (première colonne de la commande history) :

```
helm rollback my-nginx 1
Rollback was a success! Happy Helming!
kubectl describe pod my-nginx-7cffd85f5-njfc2 | grep Image:
    Image:          docker.io/bitnami/nginx:1.21.5-debian-10-r3

helm history
1               Sat Jan 15 15:40:48 2022        superseded      nginx-9.7.1
1.21.5          Install complete
2               Sat Jan 15 15:46:40 2022        superseded      nginx-9.7.1
1.21.5          Upgrade complete
3               Sat Jan 15 15:47:31 2022        superseded      nginx-9.7.1
1.21.5          Upgrade complete
4               Sat Jan 15 15:55:29 2022        deployed        nginx-9.7.1
1.21.5          Rollback to 1-r16CHART NAME: nginx
```

Télécharger un chart

Je n’aime pas installer quelque chose sans en connaître le contenu. Pour récupérer le code d’un chart il faut utiliser la commande pull

```
helm pull bitnami/nginx --untar

ls nginx/
Chart.lock  charts  Chart.yaml  ci  README.md  templates  values.schema.json
values.yaml
```

Pour instancier un chart depuis son code source :

```
helm install my-nginc --dry-run --debug ./ -f values.yaml
```

`--dry-run` accompagné de `--debug` permet de générer le manifest sans l’appliquer.

Cela va vous permettre de vous inspirer des charts existants pour écrire les vôtres. Vous pouvez aussi les stocker dans votre repository git.

Conclusion

Helm est vraiment plus qu’un outil de **templating**, c’est un vrai **gestionnaire d’applications** dédié aux **cluster Kubernetes**. Mais comment créer ses propres chart?