UNIVERSITÉ CAEN NORMANDIE

Helm, un gestionnaire de package pour Kubernetes

Virtualisation avancée - CM

IUT GRAND OUEST NORMANDIE

Maxime Lambert

2024 / 2025

BUT Informatique - Semestre 5 - Ressource R5.09

Présentation de Helm

C'est quoi un gestionnaire de package pour Kubernetes ?

- Helm permet de déployer des Charts (*package*). Un chart contient tous les fichiers nécessaires pour déployer une application Kubernetes.
- Les charts permettent de gérer les déploiements de vos applications de manière reproductible dans différents clusters ou namespaces.
- Lorsque l'on déploie un chart dans un cluster Kubernetes, on parle de *release*. Une *release* possède un nom unique dans le cluster.
- A l'instar des images de conteneurs, il est possible de stocker et partager les charts dans un registre.
- Helm utilise un moteur de modèles pour générer vos manifestes Kubernetes à partir de fichiers (*templates*) dans lesquels seront interprétées des valeurs.
- Il est aussi possible d'exécuter des actions à des moments spécifiques du cycle de vie d'une *release*, comme par exemple appliquer une migration sur une base de données après le déploiement.

Présentation de Helm

Organisation typique d'un chart

- Voici l'organisation typique d'un répertoire contenant un chart, qui dans ce cas se nommerait `mon-chart`.

- **charts/**
  - Répertoire contenant des charts dépendants, utilisés par le chart principal.

- **Chart.yaml**
  - Fichier contenant le nom du chart, sa version, sa description, et éventuelles dépendances

- **templates/**
  - Répertoire dans lequel il est possible de placer des modèles de manifestes kubernetes au format YAML.

- **values.yaml**
  - Fichier contenant les valeurs par défaut pour les variables utilisées dans les modèles de manifeste.

```
mon-chart/
|-- charts/
|-- Chart.yaml
|-- templates/
|-- values.yaml
```

1. A la découverte de Helm

Travaux à réaliser

- En vous aidant de la documentation (https://helm.sh/), installez Helm.
- Créez un répertoire de travail se nommant `vs-code-chart`. Il vous est demandé à l'intérieur de ce répertoire, de reprendre les travaux du TP2 afin de déployer votre application via un chart Helm.
- Pour vous aider, vous trouverez le contenu du fichier Chart.yaml sur cette slide.
- Packagez votre chart avec `helm package <chart-path>`
- Installez votre chart sur votre cluster dans un namespace `td3` avec `helm upgrade <name> <chart> --namespace <namespace>`

Chart.yaml

```
apiVersion: v2
name: vs-code-chart
description: A Helm chart for my application
version: 0.1
```

2. Améliorer la modularité d'un chart

Travaux à réaliser

- Le stockage nécessaire pour notre application peut également servir à d'autres applications. De plus, il peut être intéressant de modifier la ressource de stockage sans avoir à modifier le chart de l'application. Cela fait partie des bonnes pratiques d'avoir une séparation claire des responsabilités.
- Reprenez votre chart, vous allez créer un chart dépendant qui sera utilisé par le chart principal. Dans ce chart dépendant, vous vous occuperez des ressources de stockage nécessaire à votre application.
- Pensez à mettre à jour la version de votre chart, ce n'est pas obligatoire, mais c'est une bonne pratique.
- Packagez et déployez une nouvelle révision de votre chart embarquant cette modification.
- Constatez-vous un changement sur votre cluster ? Votre modification a-t-elle été appliquée ?
  - Aidez-vous des commandes `helm status`, `helm list`, `kubectl get pvc`

`Chart.yaml`

```
apiVersion: v2
name: vs-code-chart
description: A Helm chart for my application
version: 0.2
dependencies:
  - name: ma-dependance
    version: 0.1
    repository: "file://chemin-vers-tgz"
```

3. Variabilisation d'un Chart

Travaux à réaliser

- Certaines valeurs ne devrait pas être figées dans un Chart, car elles peuvent dépendre des usages. Identifiez les valeurs pouvant être variabilisées et si cela est judicieux, proposez des valeurs par défaut dans le fichier values.yaml

- Lorsque vous souhaitez référencer une variable dans un modèle de manifeste, utilisez la syntaxe suivante:
  `{{ .Values.cle.de.ma.variable }}`

Exemple de values.yaml

```
cle:
  de:
    ma:
      variable: "une valeur"
```

- Vous pouvez ensuite initialiser ou surcharger la valeur de ces variables en vous inspirant des syntaxes suivantes:
  - `helm upgrade <name> -f values.yaml`
  - `helm upgrade <name> --set cle.de.ma.variable='une autre valeur'`

- Déployez une nouvelle révision de votre chart en vous appuyant sur les nouvelles possibilités offertes par la variabilisation.

4. Eviter la redondance dans les charts

Travaux à réaliser

- Notre étiquette `app: code-server` est un moyen relativement peu discriminant d'identifier notre application. Imaginons que nous partageons notre cluster avec d'autres personnes, il est nécessaire de spécifier plus d'étiquettes afin d'éviter les collisions.

- Toutefois cela peut rapidement devenir redondant dans l'écriture des charts. Et si nous devons en ajouter, nous devrons repasser sur tous nos charts.

`_helpers.yaml`

```
{{-define "common.labels" -}}
orga: "IUT-C3"
res: "R5-09"
{{- end }}
```

- Vous pouvez définir des fonctions dans un fichier `_helpers.tpl`, le contenu de ce fichier est fourni ci-contre.

- Créez un nouveau chart nommé `common-chart` dont votre chart principal sera dépendant, placer le fichier `_helpers.tpl`. dans le répertoire `templates`.

- Modifiez vos modèles de manifeste afin de prendre en compte ces nouvelles étiquettes
  - Voici comment les intégrer dans vos fichiers :
    `{{ include "common.labels" . | nindent <nb_espace> }}`

- Déployer une nouvelle révision de votre chart à l'aide de `helm upgrade`

Bilan sur l'utilisation de Helm

- Helm avec son concept de "Chart", permet d'empaqueter une application Kubernetes. C'est un moyen cohérent, fiable et efficient pour gérer le déploiement d'une application complexe.
- Les variables (fichiers values) permettent de paramétrer les manifestes Kubernetes sans modifier directement les fichiers et ainsi personnaliser l'applications en fonction de vos besoins sur différents environnements.
- La gestion des versions d'un Chart vous permet de suivre l'évolution de la manière dont vous allez déployer et exploiter votre application. Vous pouvez très bien ne pas modifier fonctionnellement une application (tag d'une image d'un conteneur) mais modifier son intégration dans votre infrastructure.
- En décrivant votre infrastructure sous forme de code (les modèles de manifeste), vous pouvez considérer Helm comme un outil d'infrastructure en tant que code (IaC).
- Il est tout à fait possible d'intégrer Helm dans les pipelines CI/CD pour vous permettre de déployer et mettre à jour de manière automatisée vos applications Kubernetes.

Merci de votre attention

Maxime Lambert

maxime.lambert@unicaen.fr