

Empaquetage et déploiement d'applications avec Helm

Helm ajoute un langage de création de modèles au-dessus du YAML standard de Kubernetes. Vous transformez vos spécifications d'objet en modèles avec des variables pour les valeurs qui doivent changer entre les versions ou les environnements, comme la balise d'image à utiliser ou le nombre de répliques. Helm dispose de sa propre CLI que vous utilisez pour installer et mettre à niveau des applications, mais les objets déployés sont des ressources Kubernetes standard que vous pouvez gérer avec Kubectl.

Les packages d'application dans Helm sont appelés charts et vous pouvez installer un chart à partir d'un dossier local, d'une archive compressée ou d'un référentiel de charts distant (similaire à Docker Hub, mais pour les applications). Les charts contiennent simplement les modèles YAML, ce sont donc de petits téléchargements - les images de conteneur sont toujours extraites du registre d'images.

Installer Helm CLI

Helm utilise la même configuration de contexte que Kubectl pour se connecter à vos clusters Kubernetes. Pour commencer, vous devez installer la CLI Helm :

- Utilisez les instructions d'installation <https://helm.sh/docs/intro/install/>

OU

La méthode la plus simple, si vous avez un gestionnaire de paquets installé :

```
curl https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3 |  
bash
```

Vérifiez que la CLI fonctionne :

```
helm version --short
```

Vous devriez voir un numéro de version.

Vous ne devez pas utiliser de versions de Helm antérieures à la v3. Les versions plus anciennes nécessitaient un composant serveur exécuté dans Kubernetes, ce qui constituait un problème de sécurité. À partir de la v3, Helm est un outil purement côté client qui n'a besoin que de la CLI.

Déployer un chart avec des valeurs par défaut

Voici un diagramme Helm simple pour l'application whoami. Le nom de la version est utilisé dans les noms d'objet, de sorte que la même application peut être déployée plusieurs fois.

- Chart.yaml - décrit l'application ; ce sont des champs Helm standard

```
apiVersion: v2  
name: whoami  
description: Simple whoami HTTP server  
type: application  
version: 0.1.0  
appVersion: 1.0.0
```

- values.yaml - définit les valeurs par défaut des champs personnalisés utilisés dans les modèles

```
# service  
servicePort: 8020  
serviceType: NodePort  
serviceNodePort: 30028  
  
# deployment
```

```
replicaCount: 2
imageTag: sixeyed/whoami:21.04
serverMode: q
```

- `templates/deployment.yaml` - l'objet de déploiement basé sur un modèle, utilisant des variables pour les valeurs personnalisées (par exemple `.Values.imageTag`) et les objets standard (par exemple `.Release.Name`)

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: {{ .Release.Name }}-server
spec:
  replicas: {{ .Values.replicaCount }}
  selector:
    matchLabels:
      app: {{ .Release.Name }}
      component: server
  template:
    metadata:
      labels:
        app: {{ .Release.Name }}
        component: server
    spec:
      automountServiceAccountToken: false
      containers:
        - name: app
          image: {{ .Values.imageTag }}
          env:
            - name: WHOAMI_MODE
              value: {{ .Values.serverMode }}
          ports:
            - containerPort: 80
              name: http
```

- `templates/service.yaml` - le service basé sur un modèle

```
apiVersion: v1
kind: Service
metadata:
  name: {{ .Release.Name }}-server
  labels:
    app: {{ .Release.Name }}
spec:
  ports:
    - port: {{ .Values.servicePort }}
      targetPort: http
      {{- if eq .Values.serviceType "NodePort" }}
      nodePort: {{ .Values.serviceNodePort }}
      {{- end }}
  selector:
    app: {{ .Release.Name }}
    component: server
  type: {{ .Values.serviceType }}
```

- ✓ Utilisez Helm pour installer le chart à partir du `labs/helm/charts/whoamidossier`, en appelant la version `whoami-default`.

Pour installer un chart avec des valeurs par défaut, donnez-lui simplement un nom et l'emplacement du chart :


```
helm install whoami-default labs/helm/charts/whoami
```

Répertoriez vos versions Helm et vérifiez les objets Kubernetes :

```
helm ls
```

```
kubectl get all -l app.kubernetes.io/managed-by=Helm
```

Vous avez installé une version qui a créé le service et le déploiement Kubernetes. Les noms d'objet sont basés sur le nom de la version Helm.

- ✓  Confirmez que ce chart peut être déployé à nouveau avec un nouveau nom de version : vérifiez les étiquettes appliquées au Pod et le sélecteur d'étiquettes utilisé par le service.

```
kubectl get po -o wide --show-labels
```

```
kubectl describe svc whoami-default-server
```

Deux pods se trouvent dans les points de terminaison du service. L'étiquette du sélecteur provient du nom de la version, donc une deuxième version n'interférerait pas avec celle-ci.

Essayez l'application :

```
curl localhost:30028
```

Si vous répétez l'appel, vous verrez des réponses équilibrées entre les pods. Le nombre de répliques et le mode serveur sont des variables, utilisant actuellement les paramètres par défaut dans le fichier de valeurs.

Installer une version avec des valeurs personnalisées

N'importe quel champ du fichier de valeurs peut être remplacé lorsque vous installez ou mettez à niveau une version, à l'aide de set flag avec la CLI Helm.

- valeurs.yaml - contient tous les noms de variables pour l'application whoami, ainsi que les valeurs par défaut
- ✓ Installez une nouvelle version à partir du même chart whoami, appelée whoami-custom. Définissez le nombre de répliques sur 1 et le port de service sur 30038.

Vous pouvez utiliser plusieurs setindicateurs, en fournissant le nom et la valeur de la variable :

```
helm install whoami-custom --set replicaCount=1 --set serviceNodePort=30038 labs/helm/charts/whoami
```

Validez que votre nouvelle version de l'application est déployée :

```
helm ls
```

```
kubectl get pods -l component=server -L app
```

Vous devriez voir un Pod avec l'étiquette d'application whoami-custom et deux avec l'étiquette whoami-default.

Votre nouveau service doit écouter sur le port spécifié :

```
curl localhost:30038
```

Mettre à jour une version avec des valeurs personnalisées

Vous pouvez mettre à niveau une version avec Helm CLI. Vous pouvez le faire pour mettre à jour vers une nouvelle version de chart ou utiliser le même chart et modifier les valeurs déployées.

Essayez de mettre à jour la version personnalisée, en définissant une nouvelle valeur pour le mode serveur :

Ceci va échouer:

```
helm upgrade whoami-custom --set serverMode=V labs/helm/charts/whoami
```

Les valeurs personnalisées de l'installation ne sont pas réutilisées. La mise à niveau tente de modifier la valeur du port de la valeur personnalisée à la valeur par défaut, qui est déjà utilisée dans l'autre version.

- ✓ Répétez la commande de mise à niveau, mais ajoutez un indicateur pour que Helm réutilise les valeurs de la commande d'installation d'origine.

Les options de mise à niveau du Helm fournissent le flag `reuse-values` :

```
helm upgrade whoami-custom --reuse-values --set serverMode=V labs/helm/charts/whoami
```

Désormais, le port personnalisé de l'installation est réutilisé et seul le mode serveur est modifié.

Essayez l'application maintenant :

```
curl localhost:30038
```

Vérifiez les ReplicaSets pour l'installation personnalisée et vous pouvez voir que Helm apporte simplement des modifications aux objets Kubernetes - le déploiement a été mis à jour et il a déployé le changement de la manière habituelle :

```
kubectl get rs -l app=whoami-custom
```

- ✓ Vous pouvez également utiliser Helm pour restaurer les versions précédentes. Vérifiez l'historique de l'application personnalisée et revenez à la première révision.

La commande `history` répertorie toutes les révisions de la version, et la commande `rollback` revient à une révision précédente.

```
helm history whoami-custom
```

```
helm rollback whoami-custom 1
```

Après la restauration, vérifiez les ReplicaSets et vous verrez que l'original a été réajusté :

```
kubectl get rs -l app=whoami-custom
```

Et l'application fonctionne avec le mode serveur « silencieux » :

```
curl localhost:30038
```

Utilisation des référentiels de charts

Certaines équipes utilisent Helm pour empaqueter leurs propres applications - d'autres s'en tiennent aux fichiers YAML et n'utilisent Helm que pour déployer des applications tierces.

Des projets comme Prometheus et Nginx Ingress Controller publient des packages sous forme de charts Helm, ce qui facilite l'installation d'une version de production.

Les charts sont publiés dans des référentiels, qui peuvent être publics ou privés. Commencez par ajouter un référentiel simple :

```
helm repo ls
```

```
helm repo add kiamol https://kiamol.net
```

```
helm repo update
```

L'ajout et la mise à jour des référentiels de charts sont similaires aux gestionnaires de packages comme APT et APK sous Linux.

- ✓ Recherchez dans les référentiels un chart appelé vweb, et répertoriez les valeurs par défaut pour le 2.0.0 chart de version.

La commande de recherche recherche dans tous les dépôts :

```
helm search repo vweb --versions
```

Il existe deux numéros de version pour chaque ligne : la version de l'application et la version du chart. Les charts peuvent évoluer indépendamment, de sorte qu'une même version d'application peut comporter plusieurs charts.

Les valeurs par défaut sont regroupées dans le chart et vous pouvez les imprimer à partir de la CLI, en utilisant le nom du référentiel et les détails du chart :

```
helm show values kiamol/vweb --version 2.0.0
```

Le fichier de valeurs est YAML, il peut donc contenir des commentaires - très utiles pour les utilisateurs.

- ✓ Installez une version appelée vweb à partir du kiamol/vweb chart à la version 2.0.0, en utilisant un service NodePort à l'écoute sur le port 30039.

Il s'agit de la même commande d'installation, spécifiant la version du chart et l'emplacement incluant le nom du référentiel :

```
helm install --set replicaCount=1 --set serviceType=NodePort --set servicePort=30039 vweb kiamol/vweb --version 2.0.0
```

Répertoriez les services pour confirmer le déploiement :

```
kubectl get svc -l app.kubernetes.io/instance=vweb
```

Vous devriez pouvoir accéder à l'application à l'adresse <http://localhost:30039>. Ce n'est pas très excitant.

Les mises à niveau ne nécessitent pas nécessairement l'utilisation d'une version plus récente. Vous pouvez revenir au 1.0.0 tableau des versions de cette application, mais il se peut qu'elle ne fasse pas ce que vous pensez.

Lab 7 – Empaquetage & déploiement avec Helm

- ✓ Vérifiez les valeurs par défaut de la 1.0.0 version et effectuez la mise à niveau vers cette version. Comment accéder au site maintenant ?

```
helm show values kiamol/vweb --version 1.0.0
```

the v1 chart doesn't let you choose the service type, only the port

```
helm upgrade --reuse-values vweb kiamol/vweb --version 1.0.0
```

La « mise à niveau » fonctionne, mais le chart v1 n'a pas de variable pour le type de service, il est fixé à LoadBalancer. L'application est toujours disponible à l'adresse <http://localhost:30039>, mais uniquement si votre cluster prend en charge les services LoadBalancer.

Lab non guidé

Vous pouvez utiliser un fichier de valeurs locales pour remplacer les valeurs par défaut dans un chart, au lieu d'utiliser de nombreux arguments `set`.

Ce fichier de valeurs convient au chart du contrôleur d'entrée Nginx dans un environnement local :

- laboratoires/helm/ingress-nginx/dev.yaml

```
controller:
  kind: DaemonSet
  service:
    type: NodePort
    nodePorts:
      http: 30040
      https: 30033
```

Installez le contrôleur Nginx Ingress à partir du chart Helm public, en utilisant au moins la version 1.0.1 1.3.0 de l'application. Utilisez un nouvel espace de noms appelé `ingress`. Accédez au point de terminaison HTTP et confirmez que vous recevez une réponse de Nginx.

Conseils

Vous trouverez ici les instructions pour installer le contrôleur à partir du dépôt Helm :

<https://kubernetes.github.io/ingress-nginx/deploy/#using-helm>

L'interface de ligne de commande Helm utilise les mêmes arguments d'espace de noms que Kubectl, et pour la commande d'installation, vous pouvez la configurer pour créer l'espace de noms s'il n'existe pas.

Bloqué ? Essayez [les indices](#) ou consultez la [solution](#) .

Nettoyage

Supprimer toutes les versions de Helm :

```
helm uninstall vweb whoami-custom whoami-default
```

```
helm uninstall ingress-nginx -n ingress
```

```
kubectl delete ns ingress
```