HELM

A package manager for Kubernetes

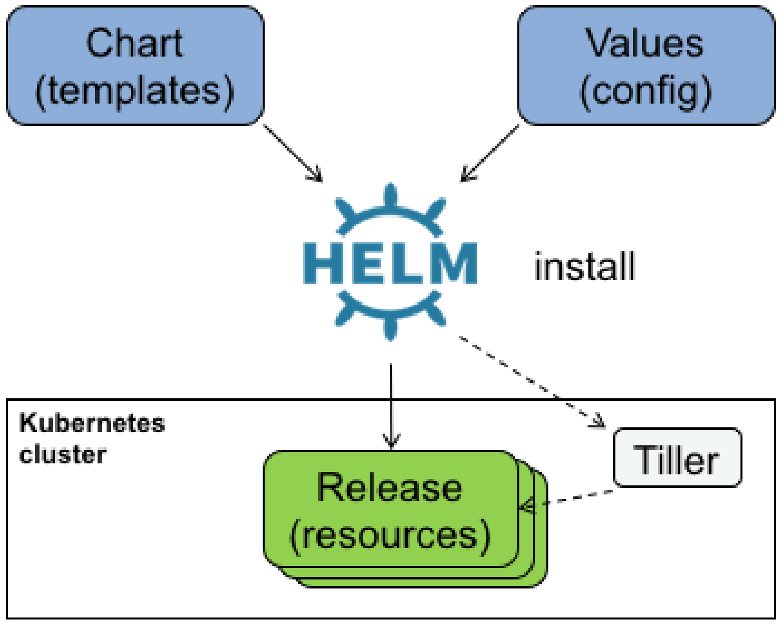
Helm – A package manager for Kubernetes

- What is a package manager?
  - Automates the process of installing, configuring, upgrading, and removing computer programs
  - Examples: Red Hat Package Manager (RPM), Homebrew …
- Helm enables multiple Kubernetes resources to be created with a single command
  - Deploying an application often involves creating and configuring multiple resources
  - A **Helm chart** defines multiple resources as a set

- An application in Kubernetes typically consists of (at least) two resource types
  - **Deployment** – Describes a set of pods to be deployed together
  - **Services** – Endpoints for accessing the APIs in those pods
  - Could also include **ConfigMaps, Secrets, Ingress**, etc.
- A default **chart** for an application consists of a deployment template and a service template
  - The chart creates all of these **resources** in a Kubernetes cluster as a set
  - Rather than manually having to create each one separately via **`kubectl`**

Helm Terminology

- Helm
  - Helm installs charts into Kubernetes, creating a new release for each installation
  - To find new charts, search Helm chart repositories
- Chart
  - Templates for a set of resources necessary to run an application
  - The chart includes a values file that configures the resources
- Repository
  - Storage for Helm charts
  - `stable` – The namespace of the hub for official charts
- Release
  - An instance of a chart running in a Kubernetes cluster
  - The same chart installed multiple times creates many releases
- Tiller
  - Helm templating engine, runs in a pod in a Kubernetes cluster
  - Tiller processes the chart to generate the resource manifests, then installs the release into the cluster
  - Tiller stores each release as a Kubernetes config map



Advantages of Using Helm

- Deploy all of the resources for an application with a single command
  - Makes deployment easy and repeatable

```
$ helm install <chart>
```

- Separates configuration settings from manifest formats
  - Edit the values without changing the rest of the manifest
  - **`values.yaml`** – Update to deploy the application differently

- Upgrade a running release to a new chart version

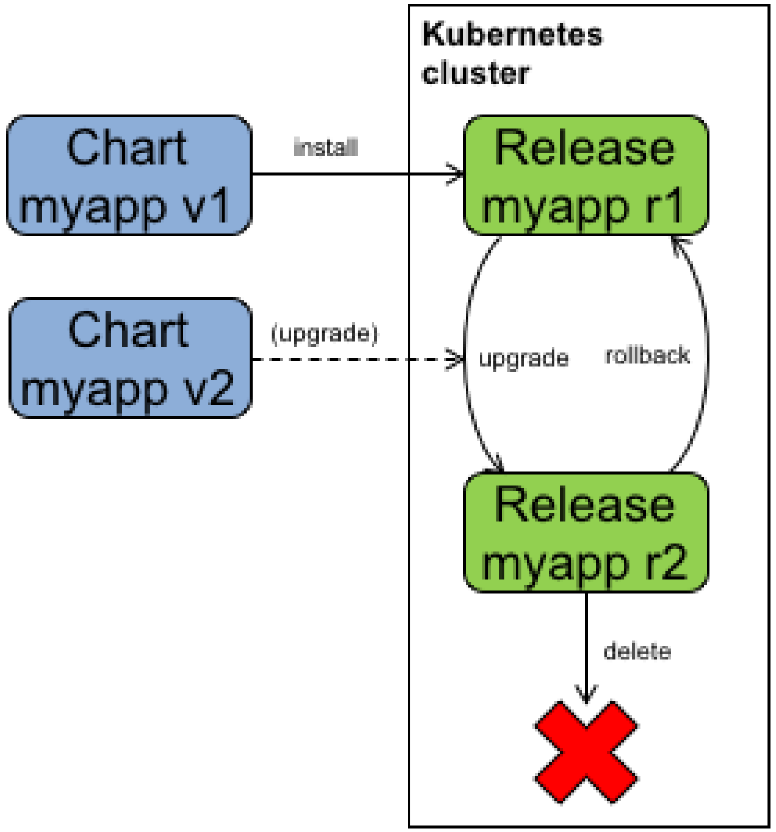
```
$ helm upgrade <release> <chart>
```

- Rollback a running release to a previous revision

```
$ helm rollback <release> <revision>
```

- Delete a running release

```
$ helm delete <release>
```



Helm Commands

- Install Tiller
```
$ helm init
```
- Create a chart
```
$ helm create <chart>
```
- List the repositories
```
$ helm repo list
```
- Search for a chart
```
$ helm search <keyword>
```
- Info about a chart
```
$ helm inspect <chart>
```
- Deploy a chart (creates a release)
```
$ helm install <chart>
```
- List all releases
```
$ helm list --all
```
- Get the status of a release
```
$ helm status <release>
```
- Get the details about a release
```
$ helm get <release>
```
- Upgrade a release
```
$ helm upgrade <release> <chart>
```
- Rollback a release
```
$ helm rollback <release> <revision>
```
- Delete a release
```
$ helm delete <release>
```

Installing an Application

- To deploy an application into Kubernetes, install that application's Helm chart

```
$ helm search mysql
NAME                VERSION     DESCRIPTION
stable/mysql  0.1.1  Chart for MySQL
$ helm install stable/mysql
Fetched stable/mysql to mysql-0.1.1.tgz
NAME: loping-toad
LAST DEPLOYED: Thu Oct 20 14:54:24 2016
NAMESPACE: default
STATUS: DEPLOYED

RESOURCES:
==> v1/Secret
NAME             TYPE    DATA   AGE
loping-toad-mysql   Opaque 2       3s

==> v1/Service
NAME             CLUSTER-IP    EXTERNAL-IP   PORT(S)       AGE
loping-toad-mysql   192.168.1.5   <none>        3306/TCP      3s

==> extensions/Deployment
NAME             DESIRED       CURRENT       UP-TO-DATE    AVAILABLE     AGE
loping-toad-mysql   1        0        0             0        3s

==> v1/PersistentVolumeClaim
NAME             STATUS VOLUME CAPACITY      ACCESSMODES   AGE
loping-toad-mysql   Pending
```

- Install output
  - Details about the release
  - Details about its resources
- Chart
  - `stable/mysql`
- Release name
  - `loping-toad` (auto generated)
- Resources
  - Four total, one of each type
  - All named `loping-toad-mysql`
  - `Secret`
  - `Service`
  - `Deployment`
  - `PersistentVolumeClaim`

Creating a Chart

- Creating a new chart generates a directory with sample files

```
$ helm create my-chart
$ tree my-chart
my-chart/
    |- Chart.yaml              # Information about the chart
    |- values.yaml                 # The default configuration values for this chart
    |- charts/                # Charts that this chart depends on
    |- templates/              # The template files
        |- NOTES.txt           # OPTIONAL: A plain text file containing short notes
        |- _helpers.tpl            # OPTIONAL: The default location for template
partials
        |- deployment.yaml
        |- service.yaml
```

- By default, a chart starts with sample templates for a Kubernetes deployment and service
  – In the simplest case, just edit the `values.yaml` file

Chart Template for Deployment Manifest

Kubernetes Deployment Manifest

```
apiVersion: apps/v1beta1
kind: Deployment
metadata:
   name: nginx-deployment
spec:
   replicas: 3
   template:
      metadata:
         labels:
            app: nginx
      spec:
         containers:
            - name: nginx
              image: nginx:1.7.9
              ports:
                - containerPort: 80
```

Helm Deployment Template

```
apiVersion: apps/v1beta1
kind: Deployment
metadata:
   name: {{ template "fullname" . }}
   labels:
      app: {{ template "name" . }}
      chart: {{ .Chart.Name }}-{{ .Chart.Version }}
      heritage: {{ .Release.Service }}
      release: {{ .Release.Name }}
spec:
   replicas: {{ .Values.replicaCount }}
   template:
      metadata:
{{- if .Values.podAnnotations }}
         annotations:
{{ toYaml .Values.podAnnotations | indent 8 }}
{{- end }}
         labels:
            app: {{ template "name" . }}
            release: {{ .Release.Name }}
      spec:
         containers:
            - name: {{ template "name" . }}
              image: "{{ .Values.image.repository }}:{{ .Values.image.tag }}"
              imagePullPolicy: {{ .Values.image.pullPolicy }}
               ports:
               - name: http
                 containerPort: 80
                 protocol: TCP
. . .
```

Resources – Introduction

- Helm - The Kubernetes Package Manager
  - https://helm.sh
  - https://docs.helm.sh
  - https://github.com/kubernetes/helm
  - https://github.com/kubernetes/helm/blob/master/docs/index.md
- Taking the Helm: Delivering Kubernetes-Native Applications by Michelle Noorali (KubeCon 2016)
  - https://www.youtube.com/watch?v=zBc1goRfk3k
- Installing Helm
  - https://docs.helm.sh/using_helm/#installing-helm

Resources – Developing Charts

- Helm examples
  - https://github.com/kubernetes/helm/tree/master/docs/examples
- Stable Helm charts
  - https://github.com/kubernetes/charts/tree/master/stable
- Golang templates
  - https://golang.org/pkg/text/template
- Sprig template library
  - https://godoc.org/github.com/Masterminds/sprig
- Getting Started Authoring Helm Charts
  - https://deis.com/blog/2016/getting-started-authoring-helm-charts
- How to Create Your First Helm Chart
  - https://docs.bitnami.com/kubernetes/how-to/create-your-first-helm-chart
- Packaged Kubernetes Deployments – Writing a Helm Chart
  - https://www.influxdata.com/packaged-kubernetes-deployments-writing-helm-chart