

HELM

Basics



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, Windows Pkgmgr/PackageManagement

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 – The CLI

- Helm installs charts into Kubernetes, creating a new release for each installation

Chart – The application package

- Templates for a set of resources necessary to run an application
- Includes a values file to configure resources

Repository – The library

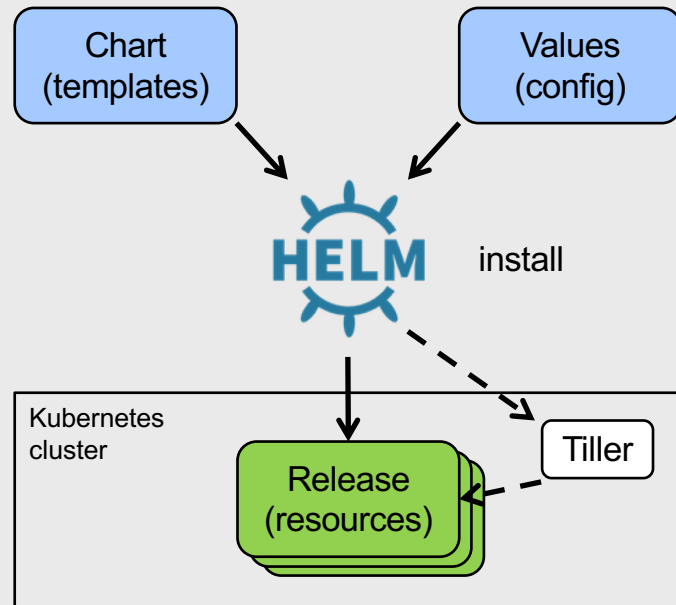
- Storage for Helm charts
- `stable` – The namespace of the hub for official charts

Release – The application runtime

- An instance of a chart running in a Kubernetes cluster

Tiller – The server-side engine

- Helm templating engine, runs in a pod in a Kubernetes cluster
- Processes the chart to generate the resource manifests, then installs the release into the cluster
- Stores each release as a Kubernetes config map



Advantages of using Helm

Deploy all 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>
```

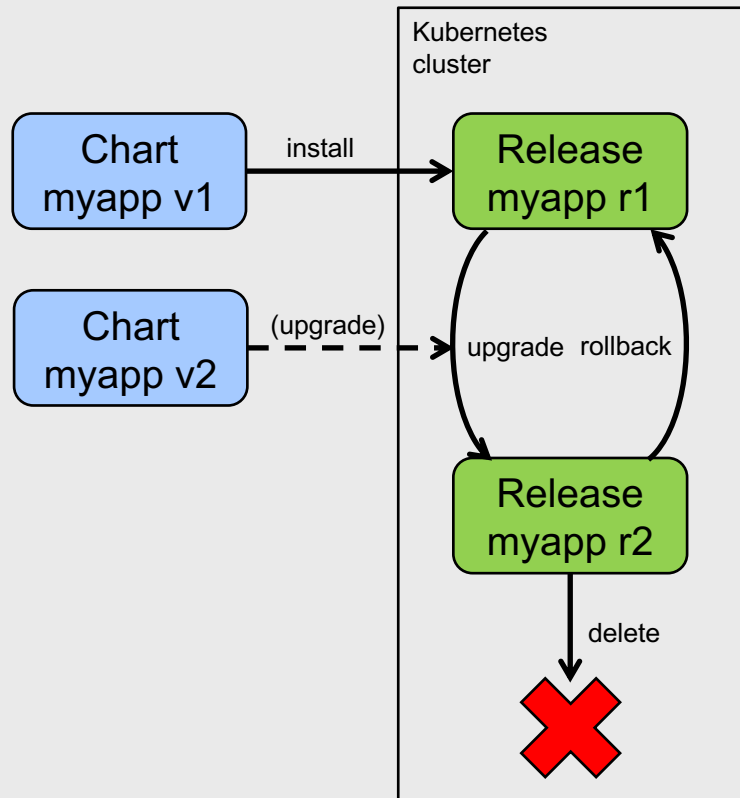
Rollback a running release to a previous revision

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

Delete a running release

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

© 2018 IBM Corporation



Installing Helm (1/2)

Helm runs as a CLI client

- Typically installed on your laptop
- https://docs.helm.sh/using_helm/#installing-helm

Installing Helm (2/2)

Options for installing Helm

1. Download the release, including the binary from:

<https://github.com/kubernetes/helm/releases>

2. Homebrew on MacOS

```
brew install kubernetes-helm
```

3. Installer script

```
curl https://raw.githubusercontent.com/kubernetes/helm/master/scripts/get  
> get_helm.sh
```

4. Install from ICP Image

https://www.ibm.com/support/knowledgecenter/en/SSBS6K_2.1.0.3/app_center/create_helm_cli.html

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>
```

Working with repositories

```
$ helm repo list
```

| NAME | URL |
|--------|---|
| stable | https://kubernetes-charts.storage.googleapis.com/ |

```
$ helm search jenkins
```

| NAME | VERSION | DESCRIPTION |
|----------------|---------|--------------------------------------|
| stable/jenkins | 0.1.14 | A Jenkins Helm chart for Kubernetes. |

```
$ helm repo add my-charts https://my-charts.storage.googleapis.com
```

```
$ helm repo list
```

| NAME | URL |
|-----------|---|
| stable | https://kubernetes-charts.storage.googleapis.com/ |
| my-charts | https://my-charts.storage.googleapis.com |

Helm and IBM Cloud Private

Catalog entries are Helm charts that can be deployed from the chart repositories.

ibm-calico-bgp-peer V 1.0.0

A Helm chart for configuring a bgp peer to your ICP Calico cluster

[ibm-charts](#)

[View Licenses](#)

VERSION 1.0.0
PUBLISHED 16th May 2018
TYPE Helm Chart

Configure a BGP Peer Resource to the Kubernetes

Introduction

A BGP peer resource (`BGPPeer`) represents a remote BGP peer with which the node(s) in a Calico cluster network with your datacenter fabric (e.g. ToR). For more information on cluster layouts, see Calico's doc

A peer can be added as a Global Peer where the added BGP Agent peers with every calico node in the cluster. Or BGP peerings can be configured on a per-node basis, i.e., configured as node-specific peers.

Chart Details

This chart will do the following:



Configure

Catalog

Search items

Deploy your applications and install software packages



ibm-calico-bgp-peer

A Helm chart for configuring a bgp peer to...

[ibm-charts](#)



ibm-datapower-dev

IBM DataPower Gateway.

[ibm-charts](#)



ibm-dsm-dev

IBM Data Server Manager Developer C Edition. Note that...

[ibm-charts](#)



ibm-cam-prod

IBM Cloud Automation Manager.

[ibm-charts](#)



ibm-db2oltp-dev

IBM Db2 Developer-C Edition 11.1.3.3

[ibm-charts](#)



ibm-dsx-dev

IBM Data Science Experience (DSX) Developer Edition brings together...

[ibm-charts](#)

Chart repository

| NAME | URL | |
|---------------------|---|------------------------|
| local-charts | http://127.0.0.1:8879 | Remove |
| microservicebuilder | http://public.dhe.ibm.com/ibmdl/export/pub/software/websphere/wasdev/microservicebuilder/helm | Remove |
| ibm-charts | https://raw.githubusercontent.com/IBM/charts/master | Remove |

Chart repository:

- HTTP server that houses an index.yaml file and optionally some packaged charts
- Server can be any HTTP server that can serve YAML and tar files and can answer GET requests

To serve a local repo

```
$ helm serve
```

To add a chart to the repository, copy it to the directory and regenerate the index

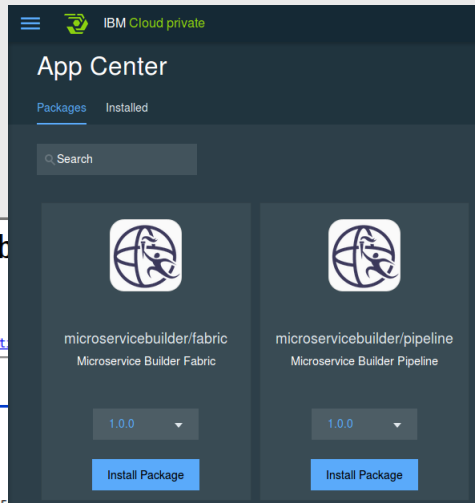
```
$ helm repo index <charts-path>
```

Index of /ibmdl/export/pub/software/web /microservicebuilder/helm

| Name | Last modified | Size | Description |
|------------------------------------|-------------------|------|-------------|
| Parent Directory | | | |
| fabric-1.0.0.tgz | 21 Jul 2017 05:31 | 6.3K | |
| index.yaml | | | |
| pipeline-1.0.0.tgz | | | |

Brought to you by IBM HTTP S

```
apiVersion: v1
entries:
  fabric:
    - apiVersion: v1
      created: 2017-06-06T10:33:02.800784489Z
      description: Microservice Builder Fabric
      digest: 4d1f14f9f2f893460e614bd4877096ef1783b0d0a2a8d8d0f2e65
      icon: https://public.dhe.ibm.com/ibmdl/export/pub/software/websphere/wasdev/microservicebuilder/icon.png
      name: fabric
      urls:
        - https://public.dhe.ibm.com/ibmdl/export/pub/software/websphere/wasdev/microservicebuilder/helm/fabric-1.0.0.tgz
      version: 1.0.0
  pipeline:
    - apiVersion: v1
      created: 2017-06-06T10:33:02.801909861Z
      description: Microservice Builder Pipeline
      digest: e3e73c769412db45ec0d33a8ce2b2809f89da08da989a311c95dbd7c68426a5c
      icon: https://public.dhe.ibm.com/ibmdl/export/pub/software/websphere/wasdev/microservicebuilder/icon.png
      name: pipeline
      urls:
        - https://public.dhe.ibm.com/ibmdl/export/pub/software/websphere/wasdev/microservicebuilder/helm/pipeline-1.0.0.tgz
      version: 1.0.0
generated: 2017-06-06T10:33:02.799077378Z
```



Deploying an application with its Helm chart

```
$ helm search mysql
```

| NAME | VERSION | DESCRIPTION |
|--------------|---------|-----------------|
| stable/mysql | 0.1.1 | Chart for MySQL |

```
$ helm install stable/mysql
```

```
Fetches 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

Overriding values

Default values are stored in the chart

```
<chart-path>/values.yaml
```

Helm CLI uses Kubernetes CLI's configuration to connect to your current cluster

```
~/.kube/config
```

```
$ kubectl config view
```

To specify a release's name, use the *name* flag

```
$ helm install --name CustomerDB stable/mysql
```

To deploy the release into a particular Kubernetes namespace, use the *namespace* flag

```
$ helm install --namespace ordering-system stable/mysql
```

To override an individual value, use the *set* flag

```
$ helm install --set user.name='student',user.password='passw0rd' stable/mysql
```

To override values with a values file, use the *values* or *f* flag

```
$ helm install --values myvalues.yaml stable/mysql
```

