HELM

Basics

IBM Cloud

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

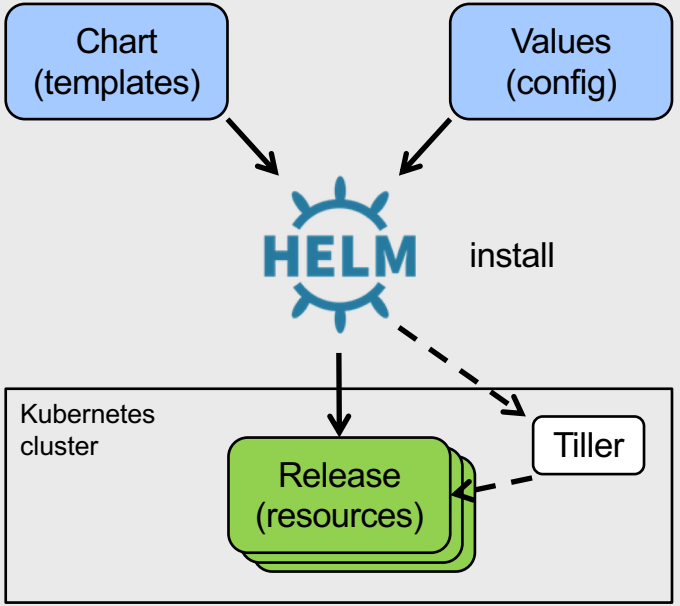
- Storage for Helm charts
- `stable` – The namespace of the hub for official charts

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

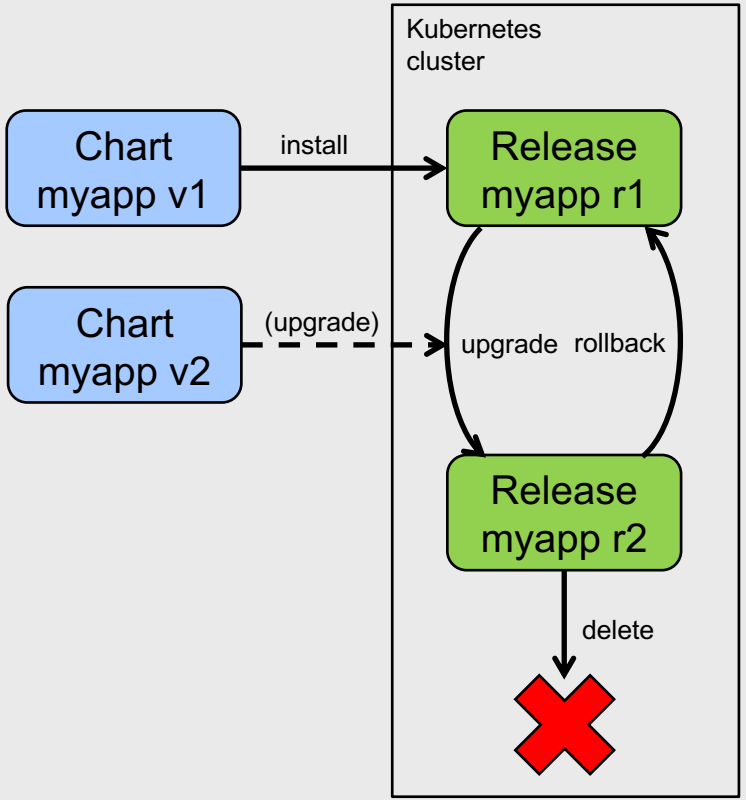
```
$ helm upgrade <release> <chart>
```

Rollback a running release to a previous revision

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

Delete a running release

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



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

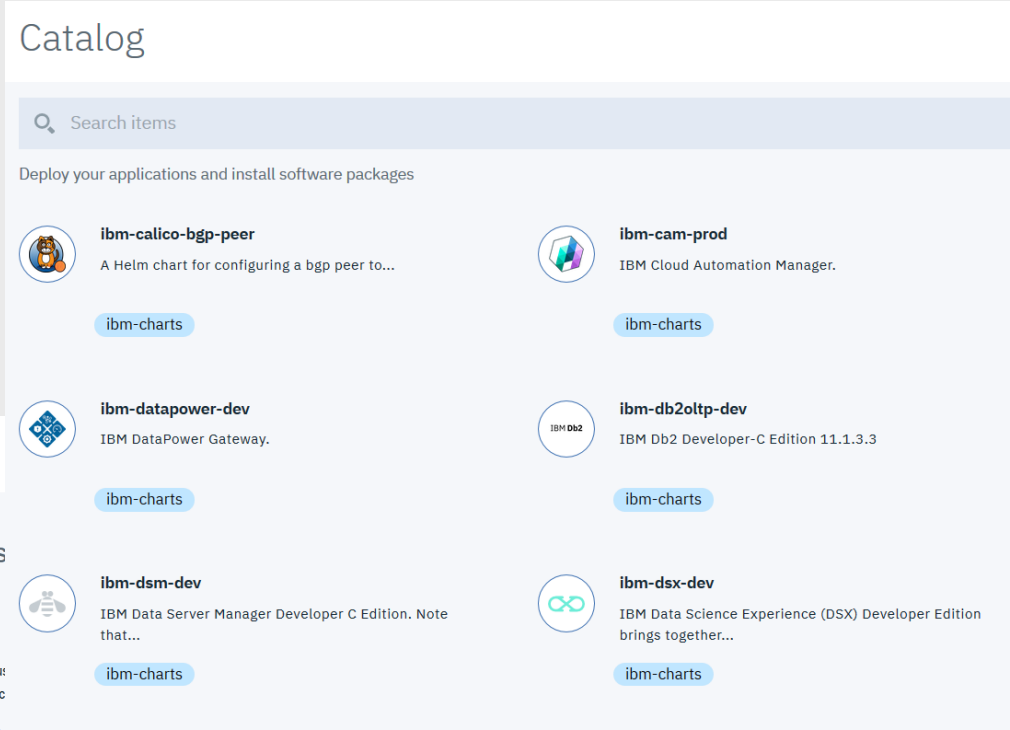
```
$ helm repo list

    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.



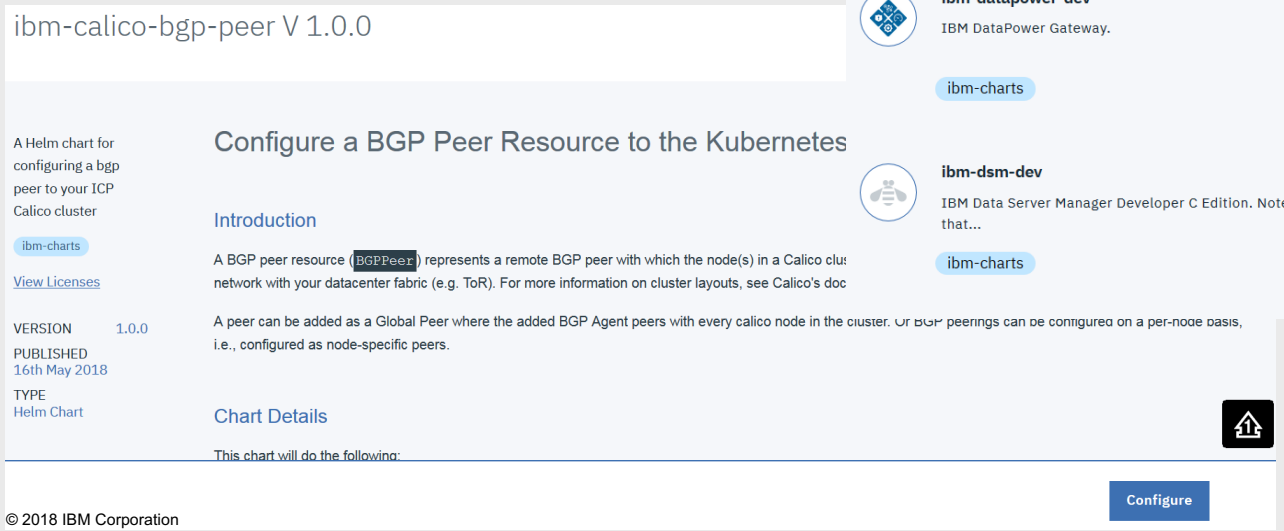


Chart repository

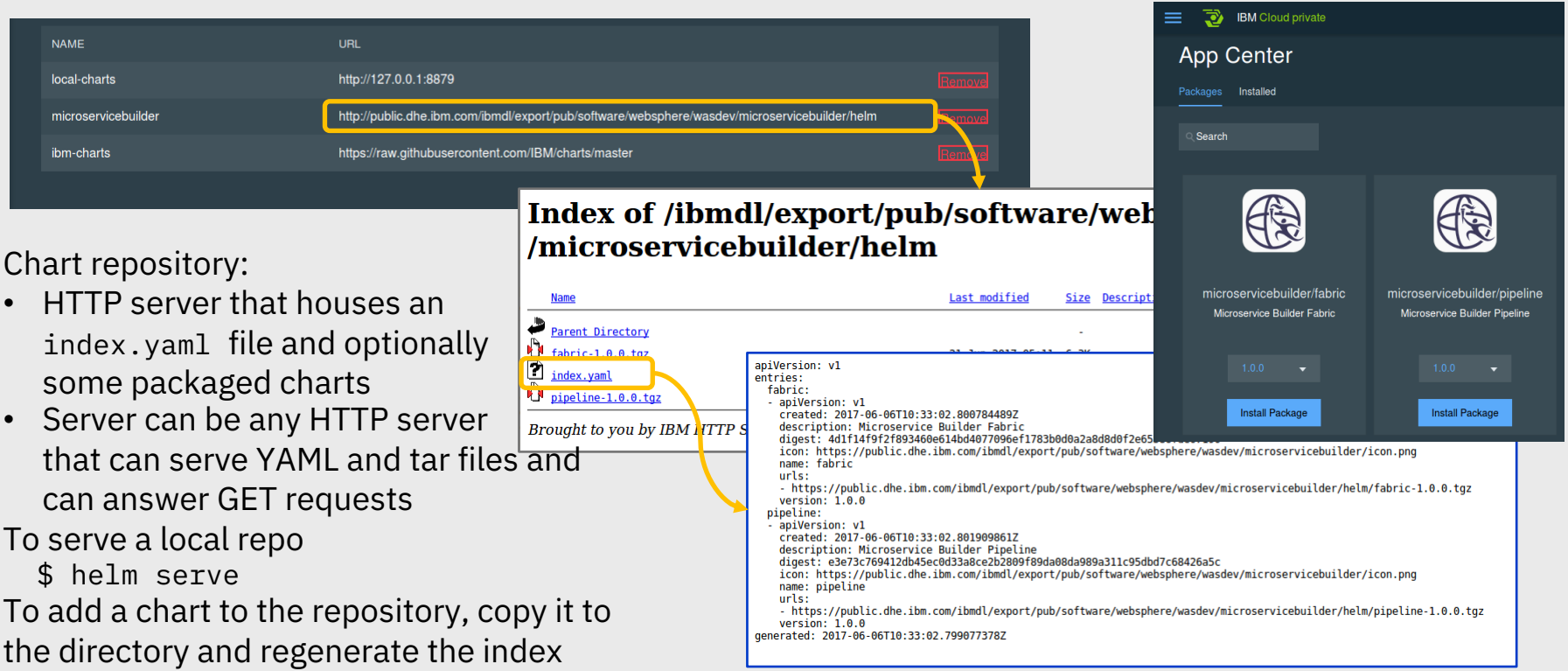


Chart repository:

- 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>
```

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

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`

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
```