Take the Wheel, Don’t Reinvent It! - Deploying Apps With Helm in 5 Minutes

Christopher Hanson, Senior Associate Consultant RX-M Enterprises LLC
In 5 Minutes !!!

Why do I need Helm?

What is Helm?

How do I use Helm?
Fine grained parts

- Kubernetes allows users to describe any application configuration they require.
- This flexibility requires Kubernetes application building blocks to be fine grained.
- Combined with the fact that cloud native applications are typically decomposed into small services, even simple applications can require many Kubernetes resources.
A simple application
A simple application
A simple application
A simple application
A simple application
A simple application
A simple application
A simple application

Deployment
ReplicaSet
Pod
Container

Service

Service Account

Roll Binding

Roll
A simple application
A simple application

Deployment

ReplicaSet

Pod

Container

Service

secret

ConfigMap

Roll

Roll Binding

Service Account
A simple application
A simple application
A simple application
can tame the madness
What is Helm?

The package manager for Kubernetes!

A great way to find, share, and use software built for Kubernetes

Helm collects all of the resources required to run an application into a single artifact: a Chart

Charts let you define, install, and upgrade complex Kubernetes applications as a single unit

Helm is a CNCF project with significant contributions from Microsoft, Google and Bitnami

https://helm.sh/
How Helm works

Simple Parameters

- `replicaCount: 1`
- `restartPolicy: Never`

# Evaluated by the post-install hook

sleepyTime: "10"

Index: ->

<h1>Hello</h1>

<p>This is a test</p>

image: repository: nginx

tag: 1.11.0

pullPolicy: IfNotPresent

Complex Charts

Powerful Applications
Installing Helm

From Snap (**Linux**)
$ sudo snap install helm

From Homebrew (**macOS**)
$ brew install kubernetes-helm

From Chocolatey (**Windows**)
C:\> choco install kubernetes-helm

From Script (**GitHub**)
https://raw.githubusercontent.com/helm/helm/master/scripts/get
Initializing Helm

$ helm init

Adding stable repo with URL: https://kubernetes-charts.storage.googleapis.com
Adding local repo with URL: http://127.0.0.1:8879/charts
$HELM_HOME has been configured at /home/user/.helm.
Tiller (the Helm server-side component) has been installed into your Kubernetes Cluster.
Please note: by default, Tiller is deployed with an insecure 'allow unauthenticated users' policy.
To prevent this, run `helm init` with the --tiller-tls-verify flag.
For more information on securing your installation see: https://docs.helm.sh/using_helm/#securing-your-helm-installation
Happy Helming!

$ kubectl create serviceaccount --namespace kube-system tiller
serviceaccount/tiller created

$ kubectl create clusterrolebinding tiller-cluster-rule \ 
   --clusterrole=cluster-admin --serviceaccount=kube-system:tiller
clusterrolebinding.rbac.authorization.k8s.io/tiller-cluster-rule created

$ kubectl patch deploy --namespace kube-system tiller-deploy \ 
   -p '{"spec":{"template":{"spec":{"serviceAccount":"tiller"}}}'}
deployment.extensions/tiller-deploy patched

helm init uses the current kubectl context from $HOME/<username>/.kube/config
to configure the target Kubernetes cluster for use with Helm

To enable Helm's server to function in an RBAC system the Helm deployment (tiller-deploy) must be given a service account (typically named “tiller”) bound to the cluster admin role
The helm CLI init command launches Helm’s server, Tiller on the configured cluster.

Tiller runs as a single pod deployment with the “tiller-deploy” service in the kube-system namespace.
A **Chart** is a Helm package

Charts contain **all of the resource definitions** necessary to run an application

A **Repository** is the place where charts can be collected and shared

A **Release** is an instance of a chart running in a Kubernetes cluster
Running applications

The **helm install** command is used to launch applications on the configured Kubernetes cluster.

Charts can be installed from **Repositories**; curated charts for many common applications can be installed from the public repo: https://kubernetes-charts.storage.googleapis.com

**helm install** “releases” the chart, combining it with configuration parameters (from the source repo in this case).

The example release here is: `full-joey`

**NOTES:**

MySQL can be accessed via port 3306 on the following DNS name from within your cluster:

```
full-joey-mysql.default.svc.cluster.local
```

To get your root password run:

```
MYSQL_ROOT_PASSWORD=$(kubectl get secret --namespace default full-joey-mysql -o jsonpath='{.data.mysql-root-password}') | base64 --decode; echo
```

To connect to your database:

1. Run an Ubuntu pod that you can use as a client:
   ```
kubectl run -i --tty ubuntu --image=ubuntu:16.04 --restart=Never -- bash
   ```
2. Install the mysql client:
   ```
   $ apt-get update & apt-get install mysql-client -y
   ```
3. Connect using the mysql cli, then provide your password:
   ```
   mysql -h full-joey-mysql -p
   ```

To connect to your database directly from outside the K8s cluster:

```
MYSQL_HOST=127.0.0.1
MYSQL_PORT=3306
```

```
# Execute the following command to route the connection:
kubectl port-forward svc/full-joey-mysql 3306
mysql -h $MYSQL_HOST -p$MYSQL_PORT -u root -p$MYSQL_ROOT_PASSWORD
```

**When you install a chart, a new release is created.**

**One chart can be installed many times in the same cluster.**

**Each release is given a name and can be independently parameterized, managed and upgraded.**
Controlling applications

Helm has many commands for working with applications:

- `$ helm status`
- `$ helm inspect`
- `$ helm list`
- `$ helm upgrade`
- `$ helm rollback`
- `$ helm delete`

MySQL can be accessed via port 3306 on the following DNS name from within your cluster:
```
full-joey-mysql.default.svc.cluster.local
```

To get your root password run:
```
MYSQL_ROOT_PASSWORD=${kubectl get secret --namespace default full-joey-mysql -o jsonpath="{.data.mysql-root-password}" | base64 -d | echo)
```

To connect to your database:

1. Run an Ubuntu pod that you can use as a client:
   ```
kubectl run -i --tty ubuntu --image=ubuntu:16.04 --restart=Never -- bash -l
   ```
2. Install the mysql client:
   ```
   $ apt-get update && apt-get install mysql-client -y
   ```
3. Connect using the mysql cli, then provide your password:
   ```
   $ mysql -h full-joey-mysql -p
   ```

To connect to your database directly from outside the K8s cluster:

- Set the environment variables:
  ```
  MYSQL_HOST=127.0.0.1
  MYSQL_PORT=3306
  ```
- Execute the following command to route the connection:
  ```
  kubectl port-forward svc/full-joey-mysql 3306
  ```
- Connect to the database:
  ```
  mysql -h $[MYSQL_HOST] -P$[MYSQL_PORT] -u root -p$[MYSQL_ROOT_PASSWORD]
  ```
A Chart is a collection of files defining the resources necessary to launch a Kubernetes application.

Charts can be packaged into versioned archives to be deployed.

Charts have a required set of files and directory structure:

```
wordpress/
  Chart.yaml   # A YAML file containing information about the chart
  LICENSE      # OPTIONAL: A plain text file containing the license for the chart
  README.md    # OPTIONAL: A human-readable README file
  requirements.yaml # OPTIONAL: A YAML file listing dependencies for the chart
  values.yaml  # The default configuration values for this chart
  charts/      # A directory containing any charts upon which this chart depends.
  templates/   # A directory of templates that, when combined with values,
                  # will generate valid Kubernetes manifest files.
  templates/NOTES.txt # OPTIONAL: A plain text file containing short usage notes
```
The Chart.yaml file is required and provides high level information about the Chart application.

Chart.yaml file fields:

- **apiVersion**: The chart API version, always "v1"
- **name**: The name of the chart (required)
- **version**: A SemVer 2 version (required)
- **appVersion**: The version of the app that this contains (optional)
- **description**: A single-sentence description of this project (optional)
- **keywords**: A list of keywords about this project (optional)
- **home**: The URL of this project's home page (optional)
- **icon**: A URL to an SVG or PNG image (optional)
- **sources**: A list of URLs to source code for this project (optional)
- **maintainers**: # (optional)
  - **name**: The maintainer's name (required for each maintainer)
  - **email**: The maintainer's email (optional for each maintainer)
  - **url**: A URL for the maintainer (optional for each maintainer)
- **engine**: Name of the template engine (optional, defaults to gotpl)
- **deprecated**: Whether this chart is deprecated (optional, boolean)
Templates

Based Go templates with a few add-ons

Stored in a chart’s templates/ folder

Contain place holders for values that can be supplied by users

When Helm installs a chart it passes every file in templates/ through the template engine resolving the template variables

Values for template variables can be supplied two ways:

Chart developers may supply a file called values.yaml in chart repo

Chart users may supply a YAML file that contains values on the command line or use the --set flag to pass a variable imperatively
Why do I need Helm?
Because K8s apps require coordinated deployment of large sets of complex resources

What is Helm?
A tool for coordinating deployment of large sets of complex resources!!!

How do I use Helm?
`helm install` pre-built charts from the public repo !!!
Beyond 5 Minutes…

What's Next?

**Security** – the default installation doesn't apply any security config

**Multitenancy** – Tiller has a one-to-many relationship with kubernetes namespaces (multi-Tiller? Tillerless Helm?)

**Create your own charts** – helm makes it easy to create new charts, find common problems in chart code, and package charts into tarballs for deployment and installation

**Private Repository** – HTTP server that houses an index.yaml file and packaged charts (ChartMuseum!)