How To Create A Helm Chart

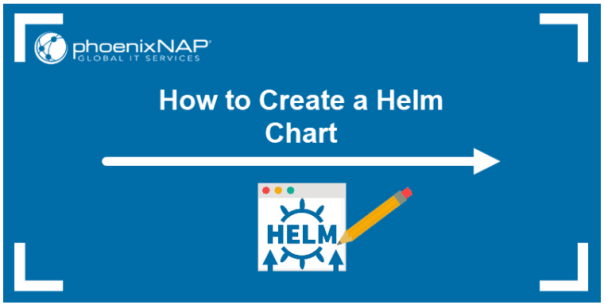
February 3, 2021 HELM KUBERNETES

Introduction

Helm charts are one of the best practices for building efficient clusters in Kubernetes. It is a form of packaging that uses a collection of Kubernetes resources. Helm charts use those resources to define an application.

Helm charts use a template approach to deploy applications. Templates give structure to projects and are suitable for any type of application.

This article provides step-by-step instructions to create and deploy a Helm chart.



Prerequisites

- Access to a CLI
- Minikube cluster installed and configured. (For assistance, follow our guides How to Install Minikube on Ubuntu and How to Install Minikube on CentOS.)
- Helm installed and configured.

Note: To confirm Helm installed properly, run `which helm` in the terminal. The output should return a path to Helm.

Create Helm Chart

Creating a Helm chart involves creating the chart itself, configuring the image pull policy, and specifying additional details in the *values.yaml* file.

Step 1: Create a New Helm Chart

1. To create a new Helm chart, use:

```
helm create <chart name>
```

For example:

```
helm create phoenixnap
```

```
~ $ helm create phoenixnap
Creating phoenixnap
```

2. Using the ls command, list the chart structure:

```
ls <chart name>
```

The Helm chart directory contains:

- **Directory *charts*** – Used for adding dependent charts. Empty by default.
- **Directory *templates*** – Configuration files that deploy in the cluster.
- YAML **file** – Outline of the Helm chart structure.
- ***YAML* file** – Formatting information for configuring the chart.

Step 2: Configure Helm Chart Image Pull Policy

1. Open the *values.yaml* file in a text editor. Locate the *image* values:

There are three possible values for the *pullPolicy*:

- `IfNotPresent` – Downloads a new version of the image if one does not exist in the cluster.
- `Always` – Pulls the image on every restart or deployment.
- `Latest` – Pulls the most up-to-date version available.

2. Change the image *pullPolicy* from `IfNotPresent` to `Always`:

Step 3: Helm Chart Name Override

To override the chart name in the *values.yaml* file, add values to the *nameOverride* and *fullnameOverride*:

For example:

```
image:
  repository: nginx
  pullPolicy: Always
  # Overrides the image tag whose default is the chart appVersion.
  tag: ""

imagePullSecrets: []
nameOverride: "phoenix-app"
fullnameOverride: "phoenix-chart"

```

Overriding the Helm chart name ensures configuration files also change.

Step 4: Specify Service Account Name

The service account name for the Helm chart generates when you run the cluster. However, it is good practice to set it manually.

The service account name makes sure the application is directly associated with a controlled user in the chart.

1. Locate the *serviceAccount* value in the *values.yaml* file:

2. Specify the *name* of the service account:

```

serviceAccount:
  # Specifies whether a service account should be created
  create: true
  # Annotations to add to the service account
  annotations: {}
  # The name of the service account to use.
  # If not set and create is true, a name is generated using the fullname template
  name: "phoenixnap"
```

Step 5: Change Networking Service Type

The recommended networking service type for Minikube is `NodePort`.

1. To change the networking service type, locate the *service* value:

2. Change the *type* from `ClusterIP` to `NodePort`:

Deploy Helm Chart

After configuring the *values.yaml* file, check the status of your Minikube cluster and deploy the application using Helm commands.

Step 1: Check minikube Status

If Minikube isn't running, the install Helm chart step returns an error.

1. Check Minikube status with:

```
minikube status
```

The status shows up as *Running*.

```
~ $ minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured
timeToStop: Nonexistent
```

2. If the status shows *Stopped*, run:

```
minikube start
```

The output shows *Done* and the status changes to *Running*.

Step 2: Install the Helm Chart

Install the Helm chart using the `helm install` command:

```
helm install <full name override> <chart name>/ --values <chart name>/values.ya
ml
```

For example:

```
helm install phoenix-chart phoenixnap/ --values phoenixnap/values.yaml
```

The `helm install` command deploys the app. The next steps are printed in the *NOTES* section of the output.

Step 3: Export the Pod Node Port and IP Address

1. Copy the two `export` commands from the `helm install` output.

2. Run the commands to get the Pod node port and IP address:

```
~ $ export NODE_PORT=$(kubectl get --namespace default -o jsonpath="{.spec.ports[0].nodePort}" serv
ices phoenix-chart)

~ $   export NODE_IP=$(kubectl get nodes --namespace default -o jsonpath="{.items[0].status.address
es[0].address}")
```

Step 4: View the Deployed Application

1. Copy and paste the `echo` command and run it in the terminal to print the IP address and port:

```
~ $ echo http://$NODE_IP:$NODE_PORT
http://192.168.49.2:30230
```

2. Copy the link and paste it into your browser, or press **CTRL+click** to view the deployed application:

Note: Learn how to delete a Helm deployment and namespace to get rid of unwanted or multiple copies of Helm deployments.

Conclusion

After following the outlined step-by-step instructions, you have a Helm chart created, set up, and deployed on a web server. Helm charts simplify application deployment on a Kubernetes cluster.

Now that you have created a Helm chart, learn How to Pull And Push Helm Charts.

Add Helm chart repositories to create more complex applications, learn how to use environment variables with Helm, or learn about other Kubernetes tools next.

Milica Dancuk

Milica Dancuk is a technical writer at phoenixNAP who is passionate about programming. Her background in Electrical Engineering and Computing combined with her teaching experience give her the ability to easily explain complex technical concepts through her content.

Next you should read

DevOps and DevOps and DevOps and DevOps and

Development, Virtualization

How to Add, Update or Remove Helm Repositories

December 21, 2020

Helm is Kubernetes' equivalent to apt and yum. It is a package manager used for deploying applications...

READ MORE

Development, Virtualization

How to Install Helm on Ubuntu, Mac and Windows

December 10, 2020

Helm is a package manager for Kubernetes that simplifies deployment process. Follow this step-by-step...

READ MORE

Development, SysAdmin, Virtualization

19 Kubernetes Best Practices for Building Efficient Clusters

September 23, 2020

Kubernetes is a feature-rich orchestration tool. The best practices outlined in this article are going to...

READ MORE

Development, Virtualization

How to Install Minikube on Ubuntu 18.04 / 20.04

April 30, 2020

Follow this step-by-step tutorial to install Minikube on your Ubuntu 18.04. Minikube allows you to work in...

READ MORE

Live Chat ↗