

Helm Overview

**What is Helm?**

Helm is a package manager in Kubernetes. Helm is a collection of tools that automates the process of installing, updating, upgrading, configuring, and removing software. It allows you to easily install an application without having prior knowledge of its setup.

For more details regarding helm, please click [**here**](https://helm.sh/docs/intro/quickstart/).

Your application can have multiple versions, it can run in multiple environments. In Kubernetes, an application is made of several files known as manifests such as **Deployments, Services, ConfigMaps, Ingress***,*etc*.* Thus, Helm is used to simplify the installation and management of Kubernetes applications by using the below concepts.

1. A**chart** is a Helm package. It groups all of the necessary manifests to run an application within the Kubernetes Cluster.

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

3. A **Values**is a file where you can define the different values related to your service, deployment, ingress, etc.

4. A **templates** is a directory in which you can define the manifest file for your deployments and services.

5. A **Release** is an instance of a chart running in a Kubernetes cluster. One chart can often be installed many times into the same cluster and each time it is installed, a new release is created.

With helm, application manifests are packaged into a single Chart. This Chartcan be put into a Repositoryand from there you can pick the chart and install that in your cluster.

**Helm Installation:**

* Helm can be install in the multiple ways. One of the ways is to install via script.
* Run the below commands and it will install helm.

curl -fsSL -o get\_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3  
chmod 700 get\_helm.sh  
./get\_helm.sh

* For other options, you can check [this](https://helm.sh/docs/intro/install/).

Once helm gets installed you can follow this below steps. For this tutorial I will show a basic example of helm chart but in the production the chart will be complex.

**Prerequisites:**

* Overview of Kubernetes
* A running kubernetes cluster
* Helm installed

1. **Create Helm Chart**

* Run the below command and create a helm chart

C02CR96VMD6M:Desktop dhruvins$ helm create my-web-app  
Creating my-web-app  
C02CR96VMD6M:Desktop dhruvins$

* The above command will create below files & directories

C02CR96VMD6M:Desktop dhruvins$ ls -la my-web-app/  
total 24  
drwxr-xr-x@ 7 dhruvins staff 224 Nov 3 18:35 .  
drwx------@ 18 dhruvins staff 576 Nov 3 18:35 ..  
-rw-r--r--@ 1 dhruvins staff 349 Nov 3 18:35 .helmignore  
-rw-r--r--@ 1 dhruvins staff 1146 Nov 3 18:35 Chart.yaml  
drwxr-xr-x@ 2 dhruvins staff 64 Nov 3 18:35 charts  
drwxr-xr-x@ 10 dhruvins staff 320 Nov 3 18:35 templates  
-rw-r--r--@ 1 dhruvins staff 1877 Nov 3 18:35 values.yaml  
C02CR96VMD6M:Desktop dhruvins$

* Chart.yaml : A YAML file that contains information about the chart. This file contains the metadata related to the chart.

apiVersion: The chart API version (required)  
name: The name of the chart (required)  
description: A single-sentence description of this project (optional)  
type: The type of the chart (i.e application) (optional)  
version: A SemVer 2 version (required)  
kubeVersion: A SemVer range of compatible Kubernetes versions (optional)  
appVersion: The version of the app that this contains (optional). Needn't be SemVer. Quotes recommended.  
maintainers: # (optional)  
 - name: The maintainers name (required for each maintainer)  
 email: The maintainers email (optional for each maintainer)  
 url: A URL for the maintainer (optional for each maintainer)

1. **The apiVersion Parameter**

* The apiVersion field should be v2 for Helm charts that require at least Helm 3. Charts supporting previous Helm versions have an apiVersion set to v1 and are still installable by Helm 3.

2. **The version Parameter**

* Every chart must have a version number. A version must follow the [SemVer 2](https://semver.org/spec/v2.0.0.html" \t "_blank) standard.

3. **The appVersion Parameter**

* Note that the appVersion field is not related to the version field. It is a way of specifying the version of the application.

4. **The kubeVersion Field**

* The optional kubeVersion field can define semver constraints on supported Kubernetes versions. Helm will validate the version constraints when installing the chart and fail if the cluster runs an unsupported Kubernetes version.
* values.yaml : A YAML file that contains configuration values for your chart.
* templates : You need to create the deployment and service file for your application and put those inside this directory.
* charts : A directory contains any charts upon which this chart depends.

**2. Define the metadata of in Chart.yaml file.**

* For this tutorial I have defined the metadata in the file like below. You can modify as per your need.

apiVersion: v2  
name: my-web-app  
description: A Helm chart for my portfolio website  
type: application  
version: 1.0.0  
appVersion: 1.0.0  
maintainers:  
 - name: Dhruvin Soni  
 email: dksoni4530@gmail.com

**3. Define the configuration in the values.yaml file**

* I have defined the configuration in values.yaml as per below.

replicaCount: 3  
image:  
 repository: dhruvin30/dhsoniweb  
 tag: latest  
 pullPolicy: IfNotPresent  
service:  
 name: my-web-app  
 type: LoadBalancer  
 port: 80  
 nameSpace: mywebapp

* replicaCount is set to 3 and it will create 3 replica of our pod.
* Under the image section there are 3 parameters. dhruvin30/dhsoniweb is the name of my repository and image which is present on the docker hub. tag is set to latest so it will fetch the latest image.
* pullPolicy is set to IfNotPresent so it will only fetch the image if it’s already not present in the cluster.
* Under the service section I have defined below parameters.
* name is for the metadata and label part.
* type is set to LoadBalancer so it will create the loadbalancer service.
* port is set to 80 so service will listen on port 80
* nameSpace is set to mywebapp so all the resources will create in this namespace.

**3. Create the deployment and service file for your application**

* Now, In the templates folder create below file for the deployment and service.

1. **deployment.yaml**

apiVersion: apps/v1  
kind: Deployment  
metadata:  
 name: {{ .Chart.Name }}  
 namespace: {{ .Values.service.nameSpace }}  
spec:  
 replicas: {{ .Values.replicaCount }}  
 selector:  
 matchLabels:  
 app: {{ .Chart.Name }}  
 template:  
 metadata:  
 labels:  
 app: {{ .Chart.Name }}  
 spec:  
 containers:  
 - name: {{ .Chart.Name }}  
 image: "{{ .Values.image.repository }}:{{ .Values.image.tag }}"  
 imagePullPolicy: {{ .Values.image.pullPolicy }}  
 ports:  
 - name: http  
 containerPort: 80

* The above file will take the values from the values.yaml file and create the deployment for your application.

**2. service.yaml**

apiVersion: v1  
kind: Service  
metadata:  
 name: {{ .Values.service.name }}  
 namespace: {{ .Values.service.nameSpace }}  
spec:  
 type: {{ .Values.service.type }}  
 ports:  
 - port: {{ .Values.service.port }}  
 protocol: TCP  
 selector:  
 app: {{ .Chart.Name }}

* The above file will take the values from the values.yaml file and create the service of type loadbalancer.

Now, you code it ready. You need to package your chart using the below command.

helm package .

* The above command will all the files and dependencies and create a tar file like below.

-rw-r--r--@ 1 dhruvins staff 14331 Nov 3 15:37 my-web-app-1.0.0.tgz

* Now, in order to install the chart in your cluster and make your application running, you need to run the below command.

helm install my-web-app my-web-app-1.0.0.tgz

* The above command will create a new release for your application. You can check this using below command.

C02CR96VMD6M:Helm-Demo dhruvins$ helm list  
NAME NAMESPACE REVISION UPDATED STATUS CHART APP VERSION  
my-web-app default 1 2023-11-03 15:37:59.491666 +0530 IST deployed my-web-app-1.0.0 1.0.0   
C02CR96VMD6M:Helm-Demo dhruvins$

* In order to check the values of your release you can run the below command.

C02CR96VMD6M:Desktop dhruvins$ helm show values my-web-app/  
replicaCount: 3  
image:  
 repository: dhruvin30/dhsoniweb  
 tag: latest  
 pullPolicy: IfNotPresent  
service:  
 name: my-web-app  
 type: LoadBalancer  
 port: 80  
 nameSpace: mywebapp  
  
C02CR96VMD6M:Desktop dhruvins$

* You can validate your chart using the below command.

C02CR96VMD6M:Desktop dhruvins$ helm lint my-web-app/  
==> Linting Helm-Demo/  
[INFO] Chart.yaml: icon is recommended  
  
1 chart(s) linted, 0 chart(s) failed  
C02CR96VMD6M:Desktop dhruvins$

**Update the release**

* Let’s say now you have to update your application and need to make a new release of it. You can follow the below steps for that.
* In this tutorial I will use the the image of tag v1 for my application. So, I have changed the content in the values.yaml file as per below.

C02CR96VMD6M:Helm-Demo dhruvins$ cat values.yaml   
replicaCount: 3  
image:  
 repository: dhruvin30/dhsoniweb  
 tag: v1  
 pullPolicy: IfNotPresent  
service:  
 name: my-web-app  
 type: LoadBalancer  
 port: 80  
 nameSpace: mywebapp  
C02CR96VMD6M:Helm-Demo dhruvins$

* I have updated the tag from latest to v1
* Now, you need to run helm upgrade command in order to create the new release.

C02CR96VMD6M:Helm-Demo dhruvins$ helm upgrade -f values.yaml my-web-app .  
Release "my-web-app" has been upgraded. Happy Helming!  
NAME: my-web-app  
LAST DEPLOYED: Fri Nov 3 19:23:02 2023  
NAMESPACE: default  
STATUS: deployed  
REVISION: 2  
TEST SUITE: None  
C02CR96VMD6M:Helm-Demo dhruvins$

* Now, you can see the release has been updated.

C02CR96VMD6M:Helm-Demo dhruvins$ helm list  
NAME NAMESPACE REVISION UPDATED STATUS CHART APP VERSION  
my-web-app default 2 2023-11-03 19:23:02.168776 +0530 IST deployed my-web-app-1.0.0 1.0.0   
C02CR96VMD6M:Helm-Demo dhruvins$

* Now, you can see the values has been updated.

C02CR96VMD6M:Desktop dhruvins$ helm show values my-web-app/  
replicaCount: 3  
image:  
 repository: dhruvin30/dhsoniweb  
 tag: v1  
 pullPolicy: IfNotPresent  
service:  
 name: my-web-app  
 type: LoadBalancer  
 port: 80  
 nameSpace: mywebapp  
  
C02CR96VMD6M:Desktop dhruvins$

* You can use the below command as well to get the values.

C02CR96VMD6M:Desktop dhruvins$ helm get values my-web-app  
USER-SUPPLIED VALUES:  
image:  
 pullPolicy: IfNotPresent  
 repository: dhruvin30/dhsoniweb  
 tag: v1  
replicaCount: 3  
service:  
 name: my-web-app  
 nameSpace: mywebapp  
 port: 80  
 type: LoadBalancer  
C02CR96VMD6M:Desktop dhruvins$

**Roll back the release**

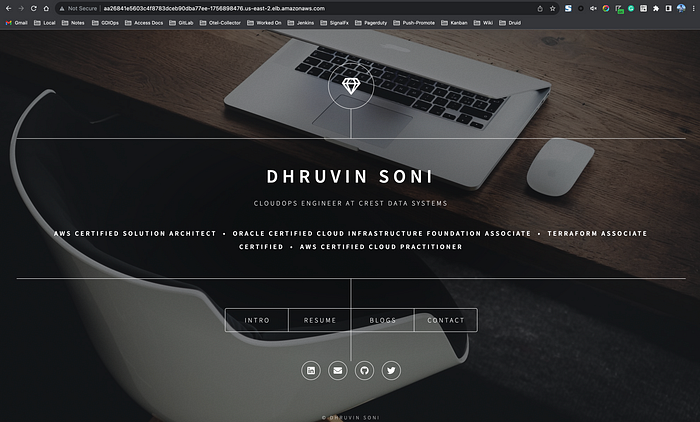
* Now, let’s say for some reason you need to roll back to the previous version then also you can run the below command.

C02CR96VMD6M:Desktop dhruvins$ helm rollback my-web-app 1  
Rollback was a success! Happy Helming!  
C02CR96VMD6M:Desktop dhruvins$

* As you can see I have rolled back to the previous release of my application.
* The above rolls back our release to its very first release version. A release version is an incremental revision. Every time an install, upgrade, or rollback happens, the revision number is incremented by 1. The first revision number is always 1. And we can use helm history [RELEASE] to see revision numbers for a certain release.

REVISION UPDATED STATUS CHART APP VERSION DESCRIPTION   
1 Fri Nov 3 15:37:59 2023 superseded my-web-app-1.0.0 1.0.0 Install complete  
2 Fri Nov 3 19:23:02 2023 superseded my-web-app-1.0.0 1.0.0 Upgrade complete  
3 Fri Nov 3 19:31:37 2023 deployed my-web-app-1.0.0 1.0.0 Rollback to 1   
C02CR96VMD6M:Desktop dhruvins$

* You can verify your application by copying the DNS of the loadbalancer in browser.



Application

**Uninstalling a Chart**

* If you need to uninstall the chart then you can run the below command.

helm uninstall [RELEASE]

**Conclusion**

This article has covered the basic usage patterns of the helm including creating, installing, updating, upgrading, rollingback and uninstalling Helm Charts.

You can find the entire code [**here**](https://github.com/DhruvinSoni30/Helm-Demo).

Feel free to check out my other repositories as well.

Follow me on [**LinkedIn**](https://www.linkedin.com/in/dhruvinksoni/)

Follow for more stories like this 😁