Getting Started with Helm Chart

Nov 7, 2020 · 8 min read · HELM CHART

· Last Modified : Nov 7, 2020 Share on: Author : Rahul Wagh

Helm charts are configuration ymls which are used for managing the Kubernetes resources.

In the production environment where you are managing lots of Kubernetes resources then **Helm Chart** can be very helpful to manage those Kubernetes resources because managing each Kubernetes resource can be a little cumbersome and daunting task.

In this article, we will start from very basic -

Table of Content

How I install Helm chart and prepared my first "Hello World" chart - Part 2
Copy link
HELM
Install Helm Chart & "HelloWorld" Helm Chart
Part - 2 (Demo)
Watch on YouTube

1. Install Helm Chart

Installing the Helm Chart pretty easy but there is a pre-requisite of setting up Kubernetes Cluster.

If you do not have a Kubernetes cluster

> Follow this guide for setting up Kubernetes cluster - Setup you Kubernetes cluster

Tags

Recent Posts

1.1: Install Helm Chart Using Script

If you like doing everything from scratch then I would suggest you to install the Helm Chart Using script.

Run the following scripts -

```bash
curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3
```

```bash
chmod 700 get_helm.sh
```

```bash
./get_helm.sh
```

Verify the Installation - You can verify the installation by running the following command

```bash
helm version
```

```bash
WARNING: Kubernetes configuration file is group-readable. This is insecure. Location: /home/vagra
version.BuildInfo{Version:"v3.4.0", GitCommit:"7090a89efc8a18f3d8178bf47d2462450349a004", GitTree
```

1.2: Install Helm Chart Using Binary

- Can Terraform be used to provision on-premises infrastructure?
- Fixing the Terraform Error creating IAM Role. MalformedPolicyDocument Has prohibited field Resource
- In terraform how to handle null value with default value?
- Terraform use module output variables as inputs for another module?

Rahul Wagh

Its all about Open Source and DevOps, here I talk about Kubernetes, Docker, Java, Spring boot and practices.

READ MORE

The other option would be to download the complete binary and do the installation be yourself

Step 1 : Download the Binary

Step 2 : Extract the binary using the command

```bash
tar -zxvf helm-vxxx-xxxx-xxxx.tar.gz
```

Step 3 : Move it to

```bash
mv linux-amd64/helm /usr/local/bin/helm
```

Verify the Installation - You can verify the installation by running the following command

```bash
helm version
```

```bash
WARNING: Kubernetes configuration file is group-readable. This is insecure. Location: /home/vagra
version.BuildInfo{Version:"v3.4.0", GitCommit:"7090a89efc8a18f3d8178bf47d2462450349a004", GitTree
```

1.3: Install Helm Chart with package Manager

If you like package manager then you use the following install command based on your preference -

Homebrew

```bash
brew install helm
```

Chocolatey

```bash
choco install kubernetes-helm
```

Scoop

```bash
scoop install helm
```

GoFish

```bash
gofish install helm
```

Snap

```bash
sudo snap install helm --classic
```

And do not forget to verify the installation

Verify the Installation - You can verify the installation by running the following command

```bash
helm version
```

```bash
WARNING: Kubernetes configuration file is group-readable. This is insecure. Location: /home/vagra
version.BuildInfo{Version:"v3.4.0", GitCommit:"7090a89efc8a18f3d8178bf47d2462450349a004", GitTree
```

2. Writing your first Helm Chart for "Hello World"

Now after you have done your Helm Chart installation, we can write our first **"Hello World"** Helm Chart.

To begin with -

2.1: Create your first Helm Chart

We are going to create our first **helloworld** Helm Chart using the following command

```bash
helm create helloworld
```

It should create a directory `helloworld`, you can verify it by using the following `ls -lart` command

```bash
ls -lart | grep helloworld
```

It should return you with -

```bash
drwxr-xr-x 4 vagrant vagrant     4096 Nov  7 19:57 helloworld
```

To verify the complete directory structure of the HelmChart please do run the command

```bash
tree helloworld
```

```bash
helloworld
├── charts
├── Chart.yaml
├── templates
│   ├── deployment.yaml
│   ├── _helpers.tpl
│   ├── hpa.yaml
│   ├── ingress.yaml
│   ├── NOTES.txt
│   ├── serviceaccount.yaml
│   ├── service.yaml
│   └── tests
│       └── test-connection.yaml
└── values.yaml
```

Great now you created your first Helm Chart - **helloworld**.

In the next steps we are going to run the **helloworld** Helm Chart.

2.2: Update the service.type from `ClusterIP` to `NodePort` inside the values.yml

Before you run your **helloworld** Helm Chart we need to update the `service.type` from `ClusterIP` to `NodePort`.

The reason for this change is - After installing/running the **helloworld** Helm Chart we should be able to access the service outside of the kubernetes cluster. And if you do not change the `service.type` then you will only be able to access the service withing kubernetes cluster.

To update the `values.yml`, first go inside the directory `helloworld`

```bash
cd helloworld
```

2.2.1: Open `values.yml` in `vi`

After that open the `values.yml` in `vi`

```bash
vi values.yaml
```

2.2.2: Update `service.type` from `ClusterIP` to `NodePort`

Look for the `service.type` block and update its value to `NodePort`

```yaml
service:
  type: NodePort
  port: 80
```

2.3: Install the Helm Chart using command - `helm install`

Now after updating the `values.yml`, you can install the Helm Chart.

Note : The `helm install` command take two arguments -

1. First argument - Release name that you pick
2. Second argument - Chart you want to install

It should look like -

```bash
helm install <FIRST_ARGUMENT_RELEASE_NAME> <SECOND_ARGUMENT_CHART_NAME>
```

Your final command would be

```bash
helm install myhelloworld helloworld
```

After running the above command it should return you with -

```bash
NAME: myhellworld
LAST DEPLOYED: Sat Nov  7 21:48:08 2020
NAMESPACE: default
STATUS: deployed
REVISION: 1
NOTES:
1. Get the application URL by running these commands:
  export NODE_PORT=$(kubectl get --namespace default -o jsonpath="{.spec.ports[0].nodePort}" ser
  export NODE_IP=$(kubectl get nodes --namespace default -o jsonpath="{.items[0].status.addresse
  echo http://$NODE_IP:$NODE_PORT
```

2.4: Verify the helm install command

Now you need to verify your helm release .i.e. `myhelloworld` and which can be done by running the `helm list` command.

```bash
helm list -a
```

It should return you withe release name which you have just installed .i.e. `myhelloworld`

```bash
NAME         NAMESPACE       REVISION        UPDATED                              STATUS
myhelloworld default         1               2020-11-07 21:48:08.8550677 +0000 UTC    deployed
```

2.5: Get kubernetes Service details and port

Lets run the `kubectl get service` command to get the `NodePort`.

```bash
kubectl get service
```

And the above command should return you -

```bash
NAME                       TYPE        CLUSTER-IP      EXTERNAL-IP    PORT(S)        AGE
kubernetes                 ClusterIP   10.233.0.1      <none>         443/TCP        14d
```

```
myhellworld-helloworld   NodePort   10.233.14.134   <none>        80:30738/TCP   7m10s
```

Note: Keep in mind the `NodePort` number can vary in the range **30000-32767**, so you might get different `NodePort`.

Since my **cluster ip** is `100.0.0.2` and **NodePort** is `30738`, so I can access my Nginx page of my `myhelloworld` Helm Chart

helm chart helloworld

3. Helm: Adding upstream repositories

We have `apt`,`yum`,`dnf` package manager in Linux distros, similarly Helm relies on bitnami chart repositories and Chart Developer can create `YAML` configuration file and package them into charts and publish it as chart repositories.

For Example - You want to deploy Redis in-memory cache inside your kubernetes cluster from Helm repository, so you can simply run the following command -

```bash
helm install redis bitnami/redis
```

Note: The above command will search for the `redis` chart inside bitnami chart repository and then it will install the `redis` chart inside your kubernetes cluster.

3.1 How to ADD upstream Helm chart repository

There are five `repo` commands provided by Helm which can be used for `add`,`list`,`remove`,`update`,`index` the chart repository.

1. `add` : Add chart repository
2. `list` : List chart repository
3. `update` : Update the chart information locally
4. `index` : For generating the index file
5. `remove` : Remove chart repository

3.1.1: 'add' Helm Chart repository

To `add` any chart repository you should know the **name** and **repository url**.

Example - We are going to add `bitnami` repository.

So our command should look like

```bash
helm repo add <REPOSITORY_NAME> <REPOSITORY_URL>
```

Here is the final command

```bash
helm repo add bitnami https://charts.bitnami.com/bitnami
```

z

Verify the repository

```bash
helm search repo bitnami
```

It should return you back with all the charts which are available inside bitnami repository

```bash
NAME                                    CHART VERSION   APP VERSION     DESCRIPTION
bitnami/bitnami-common                  0.0.8           0.0.8           Chart with custom templates
bitnami/airflow                         6.7.1           1.10.12         Apache Airflow is a platfor
bitnami/apache                          7.6.0           2.4.46          Chart for Apache HTTP Serve
bitnami/aspnet-core                     0.3.3           3.1.9           ASP.NET Core is an open-sou
bitnami/cassandra                       6.0.6           3.11.8          Apache Cassandra is a free
bitnami/common                          0.10.0          0.10.0          A Library Helm Chart for gr
bitnami/consul                          8.0.4           1.8.4           Highly available and distri
```

3.1.2: 'list' Helm Chart repository

In the previous step we have added the `bitnami` repository, lets run the `list` command for listing the repositories we have added so far.

```bash
helm repo list
```

It should return you back with

```bash
NAME            URL
bitnami         https://charts.bitnami.com/bitnami
```

3.1.3: 'update' Helm Chart repository

In the previous two steps we have seen - How to `add` and `list` the Helm Repositories.

Lets see how you can `update` your helm repositories. (*The update command is necessary if haven't updated your*

Helm chart repository in a while, so might miss some recent changes)

Here is the command to `update` Helm repository

```bash
helm repo update
```

Once your update has completed you should see following message on your console

```bash
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "bitnami" chart repository
Update Complete. □Happy Helming!□
```

3.1.4: 'index' Helm Chart repository

The `index` command can be used for generating the index file of given directory which contains the packaged charts.

So in our case we have created a chart named `helloworld`, now we are going to create `index.yaml` for it.

Run the following command -

```bash
helm repo index helloworld
```

The above command should create `index.yaml` inside your packaged charts directory.

```bash
cat helloworld/index.yaml
```

```bash
apiVersion: v1
entries: {}
generated: "2020-11-08T10:28:18.544761158Z"
```

3.1.5: 'remove' Helm Chart repository

If in case you want to remove certain repositories then Helm provides `remove` command which can be used for removing the repositories.

In the previous steps we have added `bitnami` repositories, so now we are going to remove the same repositories using the `remove` command

```bash
helm repo remove bitnami
```

After the successful removal you should see the following message

```bash
"bitnami" has been removed from your repositories
```

Read More -

1. Helm chart - How to Add/Install plugins

Posts in this series