Hey Techies… 👋

In this session , we're going to see an overview of Helm and how to create a Helm Chart..

Before going to the hands-on lab , let's have some basic ideas on Helm and helm Chart

What is Helm

Helm is a package manager for Kubernetes. Helm is the K8s equivalent of yum or apt. Helm deploys charts, which you can think of as a packaged application. It is a collection of all your versioned, pre-configured application resources which can be deployed as one unit. You can then deploy another version of the chart with a different set of configuration.

What is Helm Chart

Helm Charts are simply Kubernetes YAML manifests combined into a single package that can be advertised to your Kubernetes clusters.Creating a Helm chart for your application simplifies reproducible deployments into a Kubernetes

cluster. Users can install the whole chart with one command, instead of manually applying individual component manifests with Kubectl.

Goal of Helm

Helm is an open-source project which was originally created by DeisLabs and donated to CNCF, which now maintains it. The original goal of Helm was to provide users with a better way to manage all the Kubernetes YAML files we create on Kubernetes projects.

Benefits of Helm

Boosts productivity

Reduces duplication & complexity of deployments

Ability to leverage Kubernetes with a single CLI command
Implementation of cloud-native applications
Ability to re-use Helm charts across multiple environments

Below are the steps which we are going to perform on this hands-on activity:

1. Prerequisites - GKE
2. Check all nodes are in steady state
3. Download the helm file
4. Create Helm
5. Change the service type
6. Install Helm chart
7. Verify application
8. Delete Helm chart

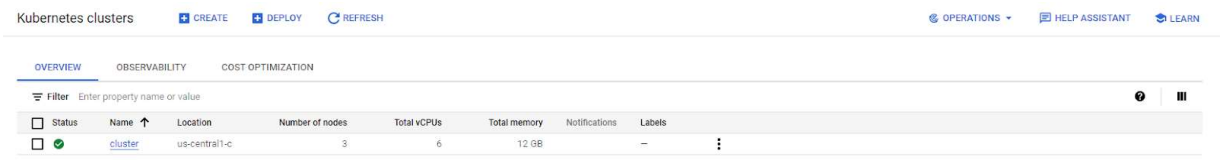
Let's get started……

1 - Prerequisites - GKE - Create Google Kubernete cluster engine in GCP

Search for Kubernetes Engine in Google Cloud console and create a cluster.

Select Standard Cluster and create with default settings.

From the image below we can confirm with green tick mark that the cluster is activated.



2 - Check all nodes are in steady state

Connect to the command-line access and check all nodes are in ready state
Kubectl get nodes

```
athirakk1827@cloudshell:~ (teak-spot-394123)$ kubectl get nodes
NAME                                          STATUS   ROLES    AGE     VERSION
gke-cluster-default-pool-8fbbc187-16rl   Ready    <none>   2m48s   v1.27.3-gke.100
gke-cluster-default-pool-8fbbc187-8nvx   Ready    <none>   2m48s   v1.27.3-gke.100
gke-cluster-default-pool-8fbbc187-dchr   Ready    <none>   2m48s   v1.27.3-gke.100
athirakk1827@cloudshell:~ (teak-spot-394123)$
```

3 -

Now we're going to download the helm file by running below command:

wget https://get.helm.sh/helm-v3.4.1-linux-amd64.tar.gz

As we can see it is a compressed file, extract it using below command and copy it into location - /usr/local/bin

tar xvf helm-v3.4.1-linux-amd64.tar.gz
sudo mv linux-amd64/helm /usr/local/bin

```
athirakk1827@cloudshell:~ (teak-spot-394123)$ wget https://get.helm.sh/helm-v3.4.1-linux-amd64.tar.gz
--2023-09-08 02:47:10--  https://get.helm.sh/helm-v3.4.1-linux-amd64.tar.gz
Resolving get.helm.sh (get.helm.sh)... 152.199.39.108, 2606:2800:247:1cb7:261b:1f9c:2074:3c
Connecting to get.helm.sh (get.helm.sh)|152.199.39.108|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 13323294 (13M) [application/x-tar]
Saving to: 'helm-v3.4.1-linux-amd64.tar.gz.1'

helm-v3.4.1-linux-amd64.tar.gz.1              100%[=========================================================================================>]  12.71M  --.-KB/s    in 0.05s

2023-09-08 02:47:11 (239 MB/s) - 'helm-v3.4.1-linux-amd64.tar.gz.1' saved [13323294/13323294]

athirakk1827@cloudshell:~ (teak-spot-394123)$
athirakk1827@cloudshell:~ (teak-spot-394123)$ tar xvf helm-v3.4.1-linux-amd64.tar.gz
linux-amd64/
linux-amd64/LICENSE
linux-amd64/README.md
linux-amd64/helm
athirakk1827@cloudshell:~ (teak-spot-394123)$
athirakk1827@cloudshell:~ (teak-spot-394123)$ sudo mv linux-amd64/helm /usr/local/bin
```

Next, the important point is to check the helm version which we installed:

helm version

```
athirakk1827@cloudshell:~ (teak-spot-394123)$ helm version
version.BuildInfo{Version:"v3.4.1", GitCommit:"c4e74854886b2efe3321e185578e6db9be0a6e29", GitTreeState:"clean", GoVersion:"go1.14.11"}
athirakk1827@cloudshell:~ (teak-spot-394123)$
```

4 - Create Helm

Now we can begin creating a Helm chart for our application. Use the below command to create a new helm in our working directory:

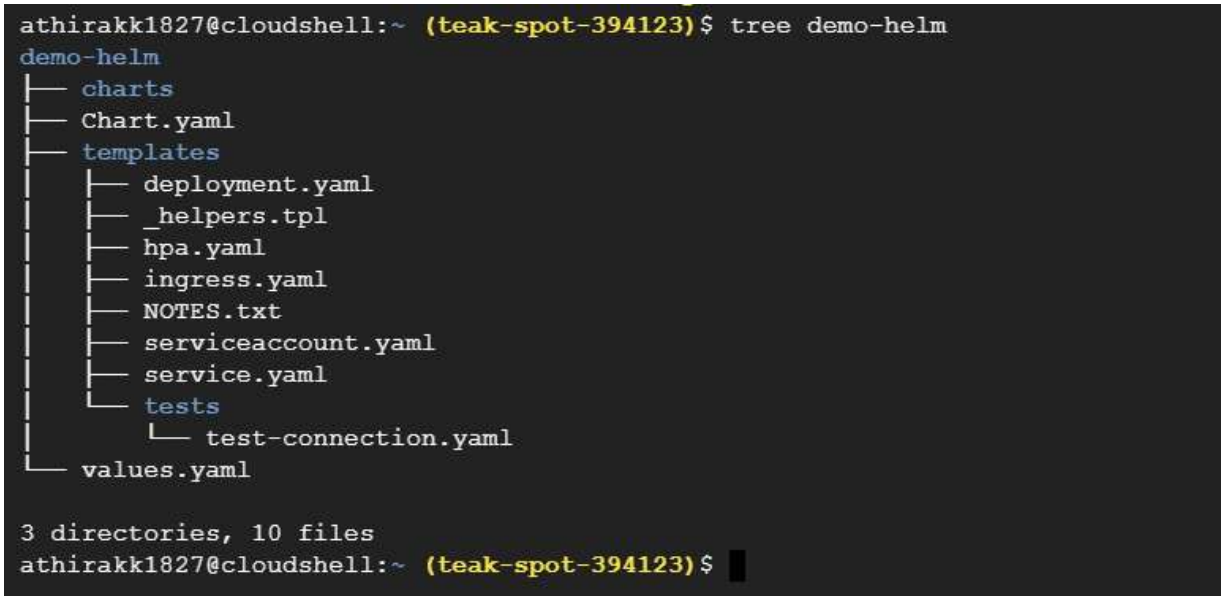
Helm create demo-helm

```
athirakk1827@cloudshell:~ (teak-spot-394123)$ helm create demo-helm
Creating demo-helm
athirakk1827@cloudshell:~ (teak-spot-394123)$
```

We can verify the helm project is created by running below command, a directory should be created with the name of demo-helm
Ls -ltr

```
athirakk1827@cloudshell:~$ ls -ltr
total 13024
-rw-r--r-- 1 athirakk1827 athirakk1827 13323294 Nov 11  2020 helm-v3.4.1-linux-amd64.tar.gz
drwxr-xr-x 2 athirakk1827 athirakk1827     4096 Sep  8 02:47 linux-amd64
drwxr-xr-x 4 athirakk1827 athirakk1827     4096 Sep  8 03:01 demo-helm
-rw-r--r-- 1 athirakk1827 athirakk1827      913 Sep  8 11:57 README-cloudshell.txt
athirakk1827@cloudshell:~$
```

We can also check the tree structure of demo-helm like below:



So once we create a helm project by running " helm create" command, we will get these yaml files by default.

There are two top-level files and two supplementary sub-directories. Here's what each resource is used for:

- Chart.yaml - Your Helm chart's manifest defining metadata properties including its name and version.
- values.yaml - This file stores default values for variables that you can reference in your chart. It's possible to override values that are set here using CLI flags when you install the chart.
- templates - The templates directory contains your chart's Kubernetes object manifests. Installing the chart will apply all these manifests to your cluster. Any valid Kubernetes YAML manifest can be placed here; you can also use extra functionality, such as references to variables defined in your values.yaml file. We'll look at this capability below.

- charts - The charts directory holds other Helm charts which this one depends on. It's used to configure complex parent-child chart relationships. We won't be covering this feature in this article so you can delete the directory if you don't need it.

5 - Change the service type

If we're checking the values.yaml file, we could see a service, it has ClusterIP service type: We need to change it to NodePort.

After installing/running the helm chart , we should be able to access the service from outside of the kubernetes cluster. If we didn't change the service type , we would be able to access the service only within the cluster.

Vim values.yaml

```
service:
  type: ClusterIP
  port: 80
```

So we have changed the service type using vim command and saved it.

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

We can also take the service type as LoadBalnacer , so we will get the external ip and we can access the deployed application through it.

Now we're going to install the helm chart using below command:
6 - Install Helm chart

```
Helm install demo-helmchart demo-helm
```

"demo-helmchart" is the custom name which we're going to deploy inside the kubernetes cluster and "demo-helm" is the directory name of helm chart we created :

Helm install demo-helmchart demo-helm

```
athirakk1827@cloudshell:~ (teak-spot-394123)$ helm install demo-helmchart demo-helm
NAME: demo-helmchart
LAST DEPLOYED: Fri Sep  8 06:51:09 2023
NAMESPACE: default
STATUS: deployed
REVISION: 1
NOTES:
1. Get the application URL by running these commands:
  export NODE_PORT=$(kubectl get --namespace default -o jsonpath="{.spec.ports[0].nodePort}" services demo-helmchart)
  export NODE_IP=$(kubectl get nodes --namespace default -o jsonpath="{.items[0].status.addresses[0].address}")
  echo http://$NODE_IP:$NODE_PORT
athirakk1827@cloudshell:~ (teak-spot-394123)$
athirakk1827@cloudshell:~ (teak-spot-394123)$
```

Let's verify which all helm-charts we're running inside our kubernetes cluster by running below command:

Helm list

```
athirakk1827@cloudshell:~ (teak-spot-394123)$ helm list
NAME            NAMESPACE       REVISION        UPDATED                                 STATUS          CHART           APP VERSION
demo-helmchart  default         1               2023-09-08 06:51:09.213244127 +0000 UTC deployed        demo-helm-0.1.0 1.16.0
athirakk1827@cloudshell:~ (teak-spot-394123)$
```

7 - Verify application

We can verify the application by running this command - " kubectl get all -o wide"

To print the services running in the kubernetes cluster , use below command:
Kubectl get svc

```
athirakk1827@cloudshell:~ (teak-spot-394123)$ kubectl get svc
NAME              TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)          AGE
demo-helmchart    NodePort    10.8.31.42    <none>        80:31825/TCP     4m19s
kubernetes        ClusterIP   10.8.16.1     <none>        443/TCP          4h12m
athirakk1827@cloudshell:~ (teak-spot-394123)$
```

Take the ip of hostname and the port which we had deployed the helm chart (from above screenshot) and paste in the browser like below :

172.18.0.1:31825

```
athirakk1827@cloudshell:~$ hostname -I
172.18.0.1 172.17.0.4
athirakk1827@cloudshell:~$
```

Here is the output…..

8 - Delete Helm chart

Do not forget to uninstall the helm chart once the learning activity is completed:
Helm uninstall demo-helmchart

```
athirakk1827@cloudshell:~$ helm uninstall demo-helmchart
release "demo-helmchart" uninstalled
athirakk1827@cloudshell:~$
```

⭐ ⭐ ⭐ Enjoy your learning….!!! ⭐ ⭐ ⭐