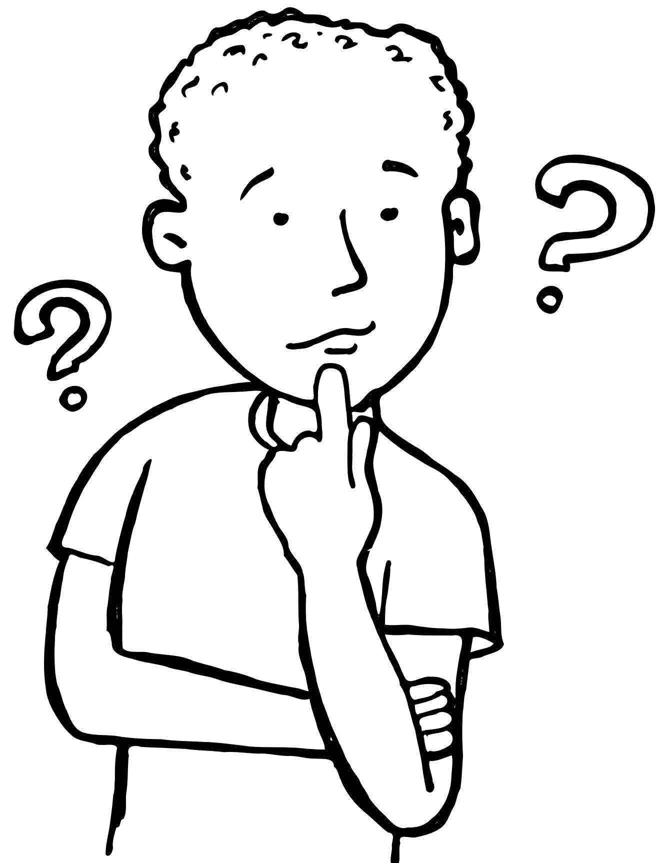
Introduction to Helm

What is Helm?

- Helm is considered as a package manager for Kubernetes.
- Helm automates the maintenance of YAML manifest file for Kubernetes object.
- Helm keeps track of version history of every chart installation.
- It helps developers and operators to package, configure and deploy application and services onto Kubernetes cluster easily.
- We can deploy versions of the chart.



Why do we need Helm?

- At some point, writing manifest YAML file for any application in Kubernetes would result in complexity since that application would have config yaml file, pod yaml file, service yaml file and many more.

- Helm helps us in managing these tedious time consuming task by allowing us to create charts where we define Kubernetes object as package.

- In short every Kubernetes object would need a separate YAML file and if we want to make any changes or upgrade our applications we would result in opening each manifest file and making those changes which is a time consuming task.

What are Helm charts?

- Helm chart can be considered as instruction manual.
- Charts can be considered as collection of files.

Concept of Helm chart:

- Charts: Template of kubernetes resources.
- Release: A chart deployed to a Kubernetes cluster.
- Repository: Storage for Helm chart.

Difference between Helm2 and Helm3?

Helm 2:

- An extra component (Tiller) was installed in the Kubernetes cluster while using helm2.
- Any action to be performed by the helm, it had to communicate with Tiller and that would communicate with kubernetes cluster.
- Roll back to older change: Helm2 compares the current revision(chart) with the previous version(chart) and if the current revison is manual change, helm2 wont find any revision and it lack in roll back to older changes.

Helm 3:

- Any user who had tiller access, could perform any action in cluster, which lead to security concern hence it was removed.
- Roll back to older change: Helm3 uses 3-way strategic merge patch to get the changes and revert back.

Folder Structure:

```
CHART_FOLDER_NAME
|
| - - - .helmignore
|
| - - - templates
                 |
                 | - - config.yaml
                 | - - service.yaml
                 | - - env.yaml
|
| - - - charts/
||
| - - -chart.yaml
|
| - - -values.yaml
```

Cont.

Folder Structure:

- Chart.yaml - All the information about the chart that we are packaging will be added in chart.yaml file
- values.yaml - These files are like the input files where configurable values are stored.
- .helmignore: This is similiar to .gitignore file that we use in git.
- Templates: Here we place our manifest files such as deployment.yaml, service.yaml,config.yaml.
- Charts: Here we store other charts.

Thank you!