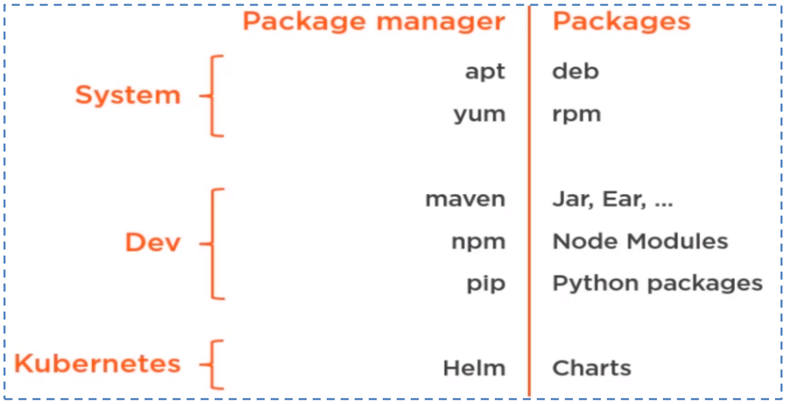
Kubernetes Package administration using Helm

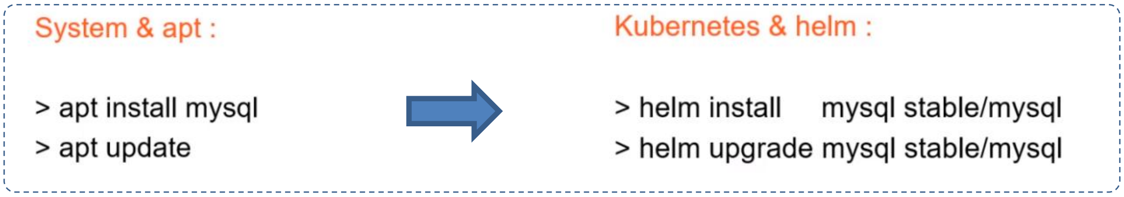
1) Package manager for Kubernetes

A simple comparison….



	Package manager	Packages
System	apt	deb
	yum	rpm
Dev	maven	Jar, Ear, …
	npm	Node Modules
	pip	Python packages
Kubernetes	Helm	Charts

2) Packages are combination of Charts, YAML files and the Charts define the Application to be deployed in Kubernetes cluster.



System & apt :

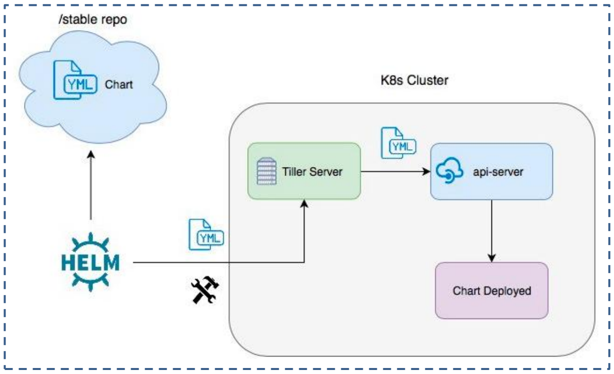
```
> apt install mysql
> apt update
```

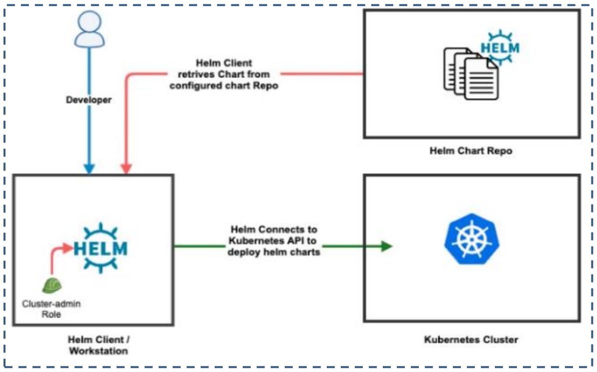
Kubernetes & helm :

```
> helm install     mysql stable/mysql
> helm upgrade mysql stable/mysql
```

3) With Helm we can easily manage,
   a. Deployment of complex applications.
   b. Ease of update / rollback of application.
   c. Charts can be shared in repositories, enabling sharing deployment code across organization.

Helm 2 v/s Helm 3



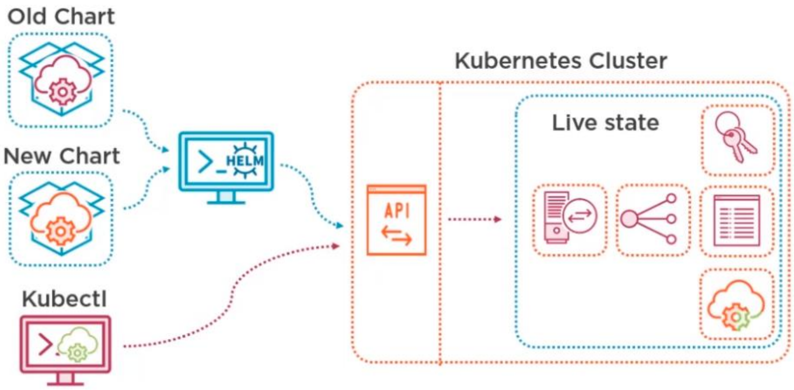

Developer
Helm Client retrives Chart from configured chart Repo
HELM
Helm Chart Repo
HELM
Cluster-admin Role
Helm Connects to Kubernetes API to deploy helm charts
Helm Client / Workstation
Kubernetes Cluster

Kubernetes Package administration using Helm

Creating a Patch Update

- Helm also helps in updating a change to live environment using chart update.
- In case if there a change done to the live environment using '`Kubectl`' and not by updating a chart, then the chart definition will not match with the live environment updates.

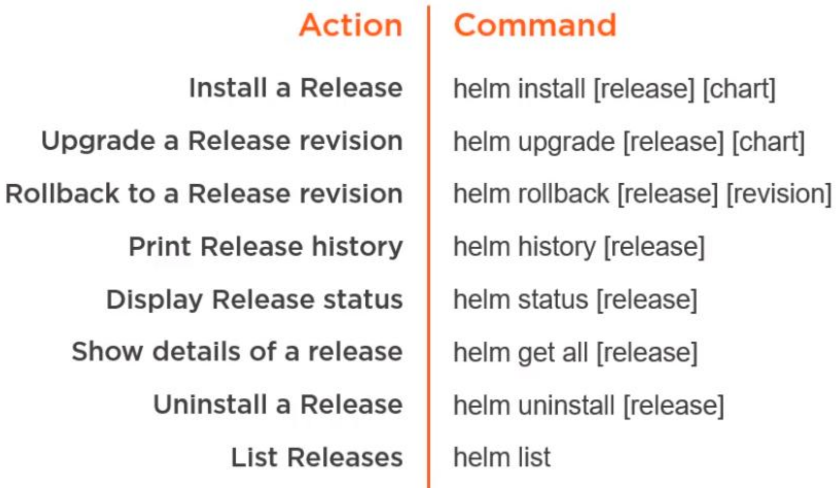
  In such scenario as well, Helm helps in creating a patch of updates and deploy it to the cluster by creating a three way Merge Patch update.



To start install Helm on windows using Chocolatey package manager

```
# choco install kubrenetes-helm  ….
```

Some helm commands



Action	Command
Install a Release	helm install [release] [chart]
Upgrade a Release revision	helm upgrade [release] [chart]
Rollback to a Release revision	helm rollback [release] [revision]
Print Release history	helm history [release]
Display Release status	helm status [release]
Show details of a release	helm get all [release]
Uninstall a Release	helm uninstall [release]
List Releases	helm list

`# helm version` .. displays the version for 'helm'

`# helm repo add` .. Adds a chart repo.

Kubernetes Package administration using Helm

`# helm search repo` .. To look for charts.

`# helm install` … to use the charts to deploy resources to cluster. … Can use `--dry-run` option to validate and test the charts.

`# helm list` .. to list the release done using charts.

To uninstall a release we can use,

`# helm uninstall <release name>`

`# helm upgrade`

`# helm rollback`

`# helm history`

`# helm create` … to create charts with default YAML files

`# helm package` --- create a package from the charts

`# helm repo add stable https://charts.helm.sh/stable` ... Adds a chart repo where from chart can be downloaded and used

`# helm repo list`

`# helm search repo stable/mysql` … here `mysql` is a chart in the stable repo

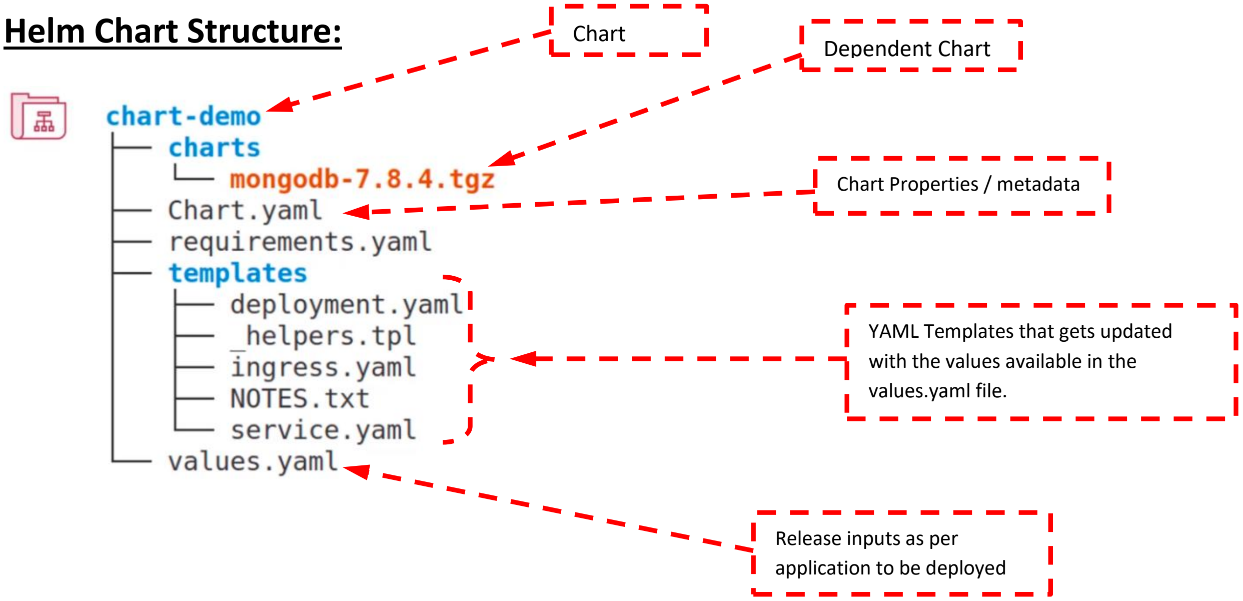
`# helm show chart stable/mysql ….` To get information on a chart before deploying it to the cluster

`# helm show readme stable/mysql` … to get readme file, if available in the chart template

`# helm show values stable/mysql` … get values available in the chart..

Kubernetes Package administration using Helm

Helm Chart Structure:



```
chart-demo
├── charts
│   └── mongodb-7.8.4.tgz
├── Chart.yaml
├── requirements.yaml
├── templates
│   ├── deployment.yaml
│   ├── _helpers.tpl
│   ├── ingress.yaml
│   ├── NOTES.txt
│   └── service.yaml
└── values.yaml
```

Chart
Dependent Chart
Chart Properties / metadata
YAML Templates that gets updated with the values available in the values.yaml file.
Release inputs as per application to be deployed

TO start, create a directory inside which the chart structure would reside.

```
$ helm create webapp
```

This command will generate a directory named <chart_name> with the following structure:

- Chart.yaml: Contains metadata about the chart, such as name, version, and description.
- values.yaml: Defines the default values for the chart's templates.
- templates/: Contains the Kubernetes resource manifests that will be rendered by Helm.
- charts/: (Optional) This directory can be used to store dependencies or subcharts.
- .helmignore: Specifies files or patterns to be ignored during the packaging of the chart.
- tests/: (Optional) This directory can be used to store tests for the chart.
- charts/: charts folder is where Helm stores its dependencies. *helm dependency list / helm dependency update chart* commands are used to work with dependencies.
-

Kubernetes Package administration using Helm

```
# vim chart.yml

    apiVersion: v2
    name: webapp
    appVersion: "1.0"
    description: helm chart for webapp
    version: 0.1.0
    type: application
```

```
# helm install webapp-release1 webapp
```

Release-name

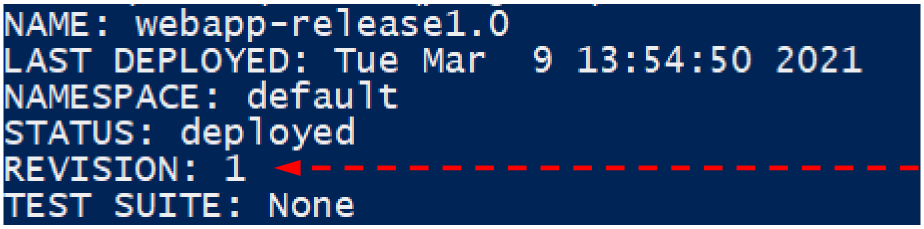
Chart - Name

If we want to deploy a new release, like a new image

1) The changes are to be done to deployment.yml file and
2) Change the version to 1.1 (next version)
3) And run command,

```
# helm upgrade <release name> <chart-name>
# helm status <release name>
```

… to check the metadata of deployed release.



```
NAME: webapp-release1.0
LAST DEPLOYED: Tue Mar  9 13:54:50 2021
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
```

Released version / revision

If we need to rollback the version to earlier revision..

```
# helm rollback <release-name>  <revision number>
```

Revision number to revert to

```
# helm history <release_name>
```