**What is Helm?**

HELM is a package manager for Kubernetes. It makes it very easy to manage the kubernetes.

**Write write 5 reason why we need helm?**

Best package manager

Using templates and variable we can reuse the same ymls again

It makes it very easy for sharing

It reduces the complexity to maintain multiple yml files and handling different environments and usecases.

With this we can easily upgrade

**How Helm works?**

HELM help by automating the kubernetes tasks. It automates the packaging, deployment of kubernetes application by defining a structured pattern. It has templates where we keep the yml file which as logic and action and during the run phase it integrates with the specified value files and executes the final action.

**What are the components of helm eco systems?**

Client and library

**What are parallel tools of helm for another platform and programming Language?**

Jinja

template

**Explained a Diretory structure of helm**

templates(Directory)  
 yml files for the job

Chart.yml

Values.yml

.helmignore

**HELM Slides**

<https://devopsschool.com/slides/helm/1.Helm-Introduction.html>

<https://www.bestdevops.com/helm-advance-notes-day-1/>

<https://www.bestdevops.com/helm-advance-notes-day-2/>

HELM commands

helm get all demo

helm template “templateName”

helm template outchart -f values.yaml > out.yaml

helm dependency

helm dependency update chart-dep/  
helm dependency build chart-dep  
helm install release-name ./chart-dep/

helm include

tree

more filename

# Helm Lab – 1: Configure Kubernetes – Install Helm – Deploy Chart

[**RAJESH KUMAR**](https://www.devopsschool.com/blog/author/rajeshkumar/)**JUNE 26, 2023**[**LEAVE A COMMENT**](https://www.devopsschool.com/blog/helm-lab-1-configure-kubernetes-install-helm-deploy-chart/#respond)

## Step 1 – create one ubuntu 20.x in aws

Software Image (AMI) - Canonical, Ubuntu, 20.04 LTS,ami-0261755bbcb8c4a84

Virtual server type (instance type) - t2.large

Firewall (security group) - default

Storage (volumes) - 1 volume(s) - 30 GiB

## Step 2 – Setting up a K8 Cluster

Master

Worker

Workstation - kubectl

======================

1 Server

1 git clone https://github.com/certifications-tutorials/cks-course-environment/

2 ls

3 cd cks-course-environment

4 ls

5 cd cluster-setup

6 ls

7 clear

8 ls

9 cd latest

10 ;s

11 ls

12 chmod 755 install\_master.sh

13 ./install\_master.sh

14 clear

15 kubectl get nodes

16 history

Make master as Worker

kubectl taint node ip-172-31-94-14 node-role.kubernetes.io/master:NoSchedule-

- plz change ip-172-31-94-14

## Step 3 – Install Helm

- https://www.devopsschool.com/blog/helm-tutorials-how-to-install-helm/

- https://helm.sh/docs/intro/install/

## Step 4 – Deploy one chart

- Install a chart

- uninstall a chart

37 kubectl get pods

38 helm repo list

39 helm repo -h

40 helm repo list

41 helm repo add bitnami https://charts.bitnami.com/bitnami

42 helm repo list

43 helm install my-nginx bitnami/nginx --version 15.0.2

44 kubectl get all

45 kubectl list

46 helm list

47 helm -h

48 helm uninstall -h

49 helm list

50 helm un my-nginx

51 kubectl get all

## Install Helm Tarfile Manually

$ wget https://get.helm.sh/helm-v3.12.1-linux-amd64.tar.gz

$ tar -zxvf helm-v3.12.1-linux-amd64.tar.gz

## From Script

$ 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

## From Script

$ curl https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3 | bash

## From Homebrew (macOS)

$ brew install helm

## From Chocolatey (Windows)

$ choco install kubernetes-helm

## From Apt (Debian/Ubuntu)

$ curl https://baltocdn.com/helm/signing.asc | gpg --dearmor | sudo tee /usr/share/keyrings/helm.gpg > /dev/null

$ sudo apt-get install apt-transport-https --yes

$ echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/helm.gpg] https://baltocdn.com/helm/stable/debian/ all main" | sudo tee /etc/apt/sources.list.d/helm-stable-debian.list

$ sudo apt-get update

$ sudo apt-get install helm

## From dnf/yum (fedora)

$ sudo dnf install helm

# Helm Tutorials: helm pull command

[**RAJESH KUMAR**](https://www.devopsschool.com/blog/author/rajeshkumar/)**JUNE 27, 2023**[**LEAVE A COMMENT**](https://www.devopsschool.com/blog/helm-tutorials-helm-pull-command/#respond)

The helm pull command is used to download a chart from a Helm repository and save it to your local machine. Here is the basic syntax of the helm pull command:

helm pull [flags] [chart]

Here’s an example of how to use the helm pull command:

$ helm repo add stable https://charts.helm.sh/stable

$ helm pull stable/mysql --version 1.6.1

$ helm env

$ ls /root/.cache/helm

$ ls /root/.cache/helm/repository/

In the above example, the command pulls the latest version of the mysql chart from the default repository (assuming it’s configured) and saves it to the current directory. The chart will be downloaded as a compressed archive file (e.g., mysql-<version>.tgz).

You can also provide additional flags to customize the behavior of the helm pull command. Here are some common flags:

* --version: Specify a specific version of the chart to download.
* --untar: Extract the chart archive after downloading.
* --untardir: Specify a directory to extract the chart into.
* --repo: Specify the URL of a repository to download the chart from.
* --username and --password: Provide credentials for accessing a private repository.

## Helm Pull Commands Example

Pull a specific chart version from a remote repository:

$ helm repo add stable https://charts.helm.sh/stable

$ helm pull stable/mysql --version 1.6.1

$ helm env

$ ls /root/.cache/helm

$ ls /root/.cache/helm/repository/

Pull a chart and save it with a specific name and version:

$ helm pull stable/nginx-ingress --version 1.2.3 --untar --untardir ./nginx-ingress

Pull a chart and save it without extracting the archive:

$ helm pull stable/redis --version 7.2.1 --untar=false

Pull a chart from a specific repository using a custom configuration file:

$ helm --kubeconfig=/path/to/kubeconfig.yaml --repository-config=/path/to/repo.yaml pull stable/mariadb

Pull a private chart from a repository using credentials:

$ helm pull private-chart --repo https://example.com/charts --username myuser --password mypassword

A screenshot of a computer

Description automatically generated

**How to check helm Configuration?**

root@ip-172-31-54-213:~/.config/helm# helm env

HELM\_BIN="helm"

HELM\_BURST\_LIMIT="100"

HELM\_CACHE\_HOME="/root/.cache/helm"

HELM\_CONFIG\_HOME="/root/.config/helm"

HELM\_DATA\_HOME="/root/.local/share/helm"

HELM\_DEBUG="false"

HELM\_KUBEAPISERVER=""

HELM\_KUBEASGROUPS=""

HELM\_KUBEASUSER=""

HELM\_KUBECAFILE=""

HELM\_KUBECONTEXT=""

HELM\_KUBEINSECURE\_SKIP\_TLS\_VERIFY="false"

HELM\_KUBETLS\_SERVER\_NAME=""

HELM\_KUBETOKEN=""

HELM\_MAX\_HISTORY="10"

HELM\_NAMESPACE="default"

HELM\_PLUGINS="/root/.local/share/helm/plugins"

HELM\_REGISTRY\_CONFIG="/root/.config/helm/registry/config.json"

HELM\_REPOSITORY\_CACHE="/root/.cache/helm/repository"

HELM\_REPOSITORY\_CONFIG="/root/.config/helm/repositories.yaml"

**Default directories depend on the Operating System in Helm**

A close-up of a computer screen

Description automatically generated

| Operating System | Cache Path | Configuration Path | Data Path |

|------------------|---------------------------|--------------------------------|-------------------------|

| Linux | $HOME/.cache/helm | $HOME/.config/helm | $HOME/.local/share/helm |

| macOS | $HOME/Library/Caches/helm | $HOME/Library/Preferences/helm | $HOME/Library/helm |

| Windows | %TEMP%\helm | %APPDATA%\helm | %APPDATA%\helm |

**Environment variables:**

| $HELM\_CACHE\_HOME | set an alternative location for storing cached files. |

| $HELM\_CONFIG\_HOME | set an alternative location for storing Helm configuration. |

| $HELM\_DATA\_HOME | set an alternative location for storing Helm data. |

| $HELM\_DEBUG | indicate whether or not Helm is running in Debug mode |

| $HELM\_DRIVER | set the backend storage driver. Values are: configmap, secret, memory, sql. |

| $HELM\_DRIVER\_SQL\_CONNECTION\_STRING | set the connection string the SQL storage driver should use. |

| $HELM\_MAX\_HISTORY | set the maximum number of helm release history. |

| $HELM\_NAMESPACE | set the namespace used for the helm operations. |

| $HELM\_NO\_PLUGINS | disable plugins. Set HELM\_NO\_PLUGINS=1 to disable plugins. |

| $HELM\_PLUGINS | set the path to the plugins directory |

| $HELM\_REGISTRY\_CONFIG | set the path to the registry config file. |

| $HELM\_REPOSITORY\_CACHE | set the path to the repository cache directory |

| $HELM\_REPOSITORY\_CONFIG | set the path to the repositories file. |

| $KUBECONFIG | set an alternative Kubernetes configuration file (default "~/.kube/config") |

| $HELM\_KUBEAPISERVER | set the Kubernetes API Server Endpoint for authentication |

| $HELM\_KUBECAFILE | set the Kubernetes certificate authority file. |

| $HELM\_KUBEASGROUPS | set the Groups to use for impersonation using a comma-separated list. |

| $HELM\_KUBEASUSER | set the Username to impersonate for the operation. |

| $HELM\_KUBECONTEXT | set the name of the kubeconfig context. |

| $HELM\_KUBETOKEN | set the Bearer KubeToken used for authentication. |

| $HELM\_KUBEINSECURE\_SKIP\_TLS\_VERIFY | indicate if the Kubernetes API server's certificate validation should be skipped (insecure) |

| $HELM\_KUBETLS\_SERVER\_NAME | set the server name used to validate the Kubernetes API server certificate |

| $HELM\_BURST\_LIMIT | set the default burst limit in the case the server contains many CRDs (default 100, -1 to disabl e)|

# Helm Lab – 3: Deploying & Retrieving & Upgrading & Rolling Back a Helm

[**RAJESH KUMAR**](https://www.devopsschool.com/blog/author/rajeshkumar/)**JUNE 27, 2023**[**LEAVE A COMMENT**](https://www.devopsschool.com/blog/helm-lab-3-deploying-retrieving-upgrading-rolling-back-a-helm/#respond)

## Deploying a Chart to Kubernetes Cluster

|  |  |
| --- | --- |
|  | # Confirm Helm version |
|  | $ helm version |
|  |  |
|  | # Add a helm repository to use the charts in the following demos |
|  | # using this custom repository as the charts in the stable repository have been deprecated. |
|  | $ helm repo add stable https://charts.helm.sh/stable |
|  |  |
|  | # Search for a chart |
|  | $ helm search repo stable/mysql |
|  |  |
|  | # confirm current context |
|  | $ kubectl config current-context |
|  |  |
|  | # test connection to cluster |
|  | $ kubectl get nodes |
|  |  |
|  | # list helm repositories |
|  | $ helm repo list |
|  |  |
|  | # search repository for a mysql chart |
|  | $ helm search repo stable/mysql |
|  |  |
|  | # show chart definition |
|  | $ helm show chart stable/mysql |
|  |  |
|  | # show chart README |
|  | $ helm show readme stable/mysql |
|  |  |
|  | # pipe README to file |
|  | $ helm show readme stable/mysql > C:\Temp\README.txt |
|  |  |
|  | # show chart values |
|  | $ helm show values stable/mysql > C:\Temp\values.txt |
|  |  |
|  | # test deployment of chart |
|  | $ helm install mysql stable/mysql --dry-run --debug |
|  |  |
|  | # deploy chart |
|  | $ helm install mysql stable/mysql |
|  |  |
|  | # confirm deployment |
|  | $ helm list |

[view raw](https://gist.github.com/devops-school/2963a8a8565bc6e36ad87fef251367b4/raw/3faccaa149ad47ee3ea43bbc9bd2cb380efd00ba/deploy.sh)[deploy.sh](https://gist.github.com/devops-school/2963a8a8565bc6e36ad87fef251367b4#file-deploy-sh)hosted with  by [GitHub](https://github.com/)

## Retrieving information about a Release

|  |  |
| --- | --- |
|  | # view deployments |
|  | $ helm list |
|  |  |
|  | # view status of release |
|  | $ helm status mysql |
|  |  |
|  | # get release manifests |
|  | $ helm get manifest mysql > C:\Temp\manifests.txt |
|  |  |
|  | # get release values - no user values were supplied so file will be blank |
|  | $ helm get values mysql > C:\Temp\values.txt |
|  |  |
|  | # get release notes |
|  | $ helm get notes mysql > C:\Temp\notes.txt |
|  |  |
|  | # get all from release |
|  | $ helm get all mysql > C:\Temp\all.txt |
|  |  |
|  | # view kubernetes objects |
|  | $ kubectl get all |
|  |  |
|  | # view release history |
|  | $ helm history mysql |
|  |  |
|  | # uninstall a release |
|  | $ helm uninstall mysql --keep-history |
|  |  |
|  | # confirm |
|  | $ helm list |
|  |  |
|  | # confirm with --all flag |
|  | $ helm list --all |
|  |  |
|  | # delete release |
|  | $ helm delete mysql |
|  |  |
|  | # confirm with --all flag |
|  | $ helm list --all |

[view raw](https://gist.github.com/devops-school/2c7b5b4372e7e49d07f8b9e6aa95200c/raw/b1fd6e5c1c4c86db53348bdfbb89aecf4642de96/helm.sh)[helm.sh](https://gist.github.com/devops-school/2c7b5b4372e7e49d07f8b9e6aa95200c#file-helm-sh)hosted with  by [GitHub](https://github.com/)

## Upgrading a Release

|  |  |
| --- | --- |
|  | # search the repository for a chart |
|  | $ helm search repo stable/mysql |
|  |  |
|  | # search the stable repository for a chart and retrieve the different versions (2 in the repo) |
|  | $ helm search repo stable/mysql --versions |
|  |  |
|  | # install a specific version of the chart |
|  | $ helm install mysql stable/mysql --version 1.6.3 |
|  |  |
|  | # confirm release deployed |
|  | $ helm list |
|  |  |
|  | # view kubernetes objects |
|  | $ kubectl get all |
|  |  |
|  | # upgrade the release |
|  | $ helm upgrade mysql stable/mysql --version 1.6.4 |
|  |  |
|  | # confirm release upgraded |
|  | $ helm list |
|  |  |
|  | # view history of release |
|  | $ helm history mysql |
|  |  |
|  | # view kubernetes objects |
|  | $ kubectl get all |

[view raw](https://gist.github.com/devops-school/03babce88ac377474f377dd4f60fbd52/raw/bd2785e0bd9d486ce4147a9d4308e545b672956a/helm.sh)[helm.sh](https://gist.github.com/devops-school/03babce88ac377474f377dd4f60fbd52#file-helm-sh)hosted with  by [GitHub](https://github.com/)

## Rolling back a Release

|  |  |
| --- | --- |
|  | # view release |
|  | helm list |
|  |  |
|  | # view release history |
|  | helm history mysql |
|  |  |
|  | # view kubernetes objects |
|  | kubectl get all |
|  |  |
|  | # view replicasets |
|  | kubectl get replicasets |
|  |  |
|  | # rollback release |
|  | helm rollback mysql 1 |
|  |  |
|  | # view release history |
|  | helm history mysql |
|  |  |
|  | # view kubernetes objects |
|  | kubectl get all |
|  |  |
|  | # view replicasets |
|  | kubectl get replicasets |
|  |  |
|  | # view secrets |
|  | kubectl get secrets |
|  |  |
|  |  |
|  | ################################################################################################### |
|  | # Rolling back without the old replicaset |
|  | ################################################################################################### |
|  |  |
|  | # clean up previous release |
|  | helm delete mysql |
|  |  |
|  | # deploy specific version of chart |
|  | helm install mysql stable/mysql --version 1.6.3 |
|  |  |
|  | # upgrade release |
|  | helm upgrade mysql stable/mysql --version 1.6.4 |
|  |  |
|  | # confirm history |
|  | helm history mysql |
|  |  |
|  | # view kubernetes objects |
|  | kubectl get all |
|  |  |
|  | # view replicasets |
|  | kubectl get replicaset |
|  |  |
|  | # delete old replicaset - replace NAME with the old replicaset name |
|  | kubectl delete replicaset NAME |
|  |  |
|  | # confirm deletion |
|  | kubectl get replicaset |
|  |  |
|  | # try a rollback with kubectl - will fail as we have deleted the old replicaset |
|  | kubectl rollout undo deployment/mysql |
|  |  |
|  | # but we still have the history of the release in Helm |
|  | helm history mysql |
|  |  |
|  | # rollback with helm |
|  | helm rollback mysql 1 |
|  |  |
|  | # confirm status of release |
|  | helm list |
|  |  |
|  | # confirm rollback |
|  | helm history mysql |
|  |  |
|  | # view replicaset |
|  | kubectl get replicaset |
|  |  |
|  | # the old replicaset is back! |
|  | # this is due to the release history being stored as secrets in the k8s cluster |
|  | kubectl get secrets |
|  |  |
|  | # let's a look at one of those secrets |
|  | kubectl get secret sh.helm.release.v1.mysql.v1 -o yaml |
|  |  |
|  | $ kubectl get secret sh.helm.release.v1.mysql.v1 -o jsonpath="{ .data.release }" | base64 -d | base64 -d | gunzip -c | jq '.chart.templates[].data' | tr -d '"' | base64 -d |
|  |  |
|  |  |
|  | # the secret is encoded. For further information on decoding these secrets, see here: - |
|  | # https://dbafromthecold.com/2020/08/10/decoding-helm-secrets/ |

# Helm Lab – 4: Creating and Packaging a Helm Chart

[**RAJESH KUMAR**](https://www.devopsschool.com/blog/author/rajeshkumar/)**JUNE 27, 2023**[**LEAVE A COMMENT**](https://www.devopsschool.com/blog/helm-lab-4-creating-and-packaging-a-helm-chart/#respond)

## Windows Commands

|  |  |
| --- | --- |
|  | # create a chart |
|  | helm create ourchart |
|  |  |
|  | # view the chart directory |
|  | ls .\ourchart |
|  |  |
|  | # view the templates directory |
|  | ls .\ourchart\templates |
|  |  |
|  | # remove all yaml files in the templates directory |
|  | rm .\ourchart\templates\\*yaml |
|  |  |
|  | # generate deployment yaml |
|  | kubectl create deployment nginx ` |
|  | --image=nginx ` |
|  | --dry-run=client ` |
|  | --output=yaml |
|  |  |
|  | # create a new deployment yaml file |
|  | kubectl create deployment nginx ` |
|  | --image=nginx ` |
|  | --dry-run=client ` |
|  | --output=yaml > .\ourchart\templates\deployment.yaml |
|  |  |
|  | # create a deployment (in order to run the expose command below) |
|  | kubectl create deployment nginx --image=nginx |
|  |  |
|  | # generate service yaml |
|  | kubectl expose deployment nginx ` |
|  | --type=LoadBalancer ` |
|  | --port=80 ` |
|  | --dry-run=client ` |
|  | --output=yaml |
|  |  |
|  | # create a new service yaml file |
|  | kubectl expose deployment nginx ` |
|  | --type=LoadBalancer ` |
|  | --port=80 ` |
|  | --dry-run=client ` |
|  | --output=yaml > .\ourchart\templates\service.yaml |
|  |  |
|  | # delete the deployment |
|  | kubectl delete deployment nginx |
|  |  |
|  | # replace NOTES.txt file |
|  | rm .\ourchart\templates\NOTES.txt |
|  | echo 'A test Helm Chart' > .\ourchart\templates\NOTES.txt |
|  |  |
|  | # remove the charts and tests directory within the chart (they're not needed) |
|  | rm -r .\ourchart\charts |
|  | rm -r .\ourchart\templates\tests |
|  |  |
|  | # deploy the chart |
|  | helm install ourchart .\ourchart |
|  |  |
|  |  |
|  | ################################################################################################### |
|  | ################################################################################################### |
|  | # If you are running Powershell v5.1 and run into the following error: - |
|  | # |
|  | # Error: unable to build kubernetes objects from release manifest: error parsing : |
|  | # error converting YAML to JSON: yaml: invalid leading UTF-8 octet |
|  | # |
|  | # Tomica Kaniski posted a work around here: - |
|  | # https://blog.kaniski.eu/2020/09/having-fun-with-helm-and-file-encoding/ |
|  | # |
|  | # If you then run into this error: - |
|  | # Error: unable to build kubernetes objects from release manifest: error validating "": |
|  | # error validating data: apiVersion not set |
|  | # |
|  | # Check the version of Helm you are using. Upgrading to Helm v3.3.1 will fix this issue: - |
|  | # choco upgrade kubernetes-helm |
|  | ################################################################################################### |
|  | ################################################################################################### |
|  |  |
|  | # view release |
|  | helm list |
|  |  |
|  | # view kubernetes objects |
|  | kubectl get all |
|  |  |
|  | # delete the release |
|  | helm delete ourchart |
|  |  |
|  | # update the deployment yaml to use template directives |
|  | # copy {{ .Release.Name }} in for the deployment name |
|  |  |
|  | # also update the value.yaml file, removing the old file |
|  | rm .\ourchart\values.yaml |
|  |  |
|  | # and add in a custom value |
|  | echo 'containerImage: nginx:1.17' > .\ourchart\values.yaml |
|  |  |
|  |  |
|  | # update the deployment yaml to use the new default value |
|  | # copy {{ .Values.containerImage }} in for the container image name |
|  |  |
|  | # now redeploy the chart |
|  | helm install ourchart .\ourchart |
|  |  |
|  | # view release |
|  | helm list |
|  |  |
|  | # view kubernetes objects |
|  | kubectl get all |
|  |  |
|  | # view the container image in the deployment |
|  | kubectl get deployment -o jsonpath='{ .items[\*].spec.template.spec.containers[\*].image }' |
|  |  |
|  | # now upgrade the chart, overriding the image name in the values file |
|  | helm upgrade ourchart .\ourchart --set containerImage=nginx:1.18 |
|  |  |
|  | # view the container image in the deployment |
|  | kubectl get deployment -o jsonpath='{ .items[\*].spec.template.spec.containers[\*].image }' |
|  |  |
|  | # package the chart |
|  | helm package .\ourchart --destination C:\Charts |
|  |  |
|  | # view packaged chart |
|  | ls C:\Charts\ |

[view raw](https://gist.github.com/devops-school/d5441d439436d4da97a666b2ab6b7bf8/raw/6e356538e413c8ac5f3a844572cd5dfb02f3beb6/helm.sh)[helm.sh](https://gist.github.com/devops-school/d5441d439436d4da97a666b2ab6b7bf8#file-helm-sh)hosted with  by [GitHub](https://github.com/)

## Linux Commands

$ mkdir demo2

$ cd demo2

$ helm create ourchart

$ ls .\ourchart

$ ls ourchart

$ ls ourchart/templates/

$ rm -rf ourchart/templates/\*.yaml

$ ls ourchart/templates/

$ kubectl create deployment nginx --image=nginx --dry-run=client --output=yaml

$ kubectl create deployment nginx --image=nginx --dry-run=client --output=yaml > ourchart/templates/deployment.yaml

$ kubectl create deployment nginx --image=nginx

$ kubectl expose deployment nginx --type=LoadBalancer --port=80 --dry-run=client --output=yaml

$ kubectl expose deployment nginx --type=LoadBalancer --port=80 --dry-run=client --output=yaml > ourchart/templates/service.yaml

$ kubectl delete deployment nginx

$ rm ourchart/templates/NOTES.txt

$ echo 'A test Helm Chart' > ourchart/templates/NOTES.txt

$ rm -rf ourchart/charts/

$ rm -rf ourchart/templates/tests/

$ tree

$ kubectl deploy

$ helm list

$ helm uninstall mysql devopsx

$ kubectl get pods

$ helm install ourchart ./ourchart

$ kubectl get all

$ curl http://10.104.136.0

$ helm list

$ kubectl get all

$ helm delete ourchart

$ rm ourchart/values.yaml

$ echo 'containerImage: nginx:1.17' > ourchart/values.yaml

$ more ourchart/values.yaml

$ vi ourchart/templates/deployment.yaml

$ helm install ourchart ./ourchart

$ kubectl get all

$ kubectl get deployment -o jsonpath='{ .items[\*].spec.template.spec.containers[\*].image }'

$ helm upgrade ourchart ./ourchart --set containerImage=nginx:1.18

$ kubectl get all

$ kubectl get deployment -o jsonpath='{ .items[\*].spec.template.spec.containers[\*].image }'

$ helm list

$ helm show ourchart -h

$ helm list -h

$ helm get -h

$ helm upgrade -h

$ helm package ./ourchart --destination package

# Helm Lab – 5: Creating Local & Remote Helm Repository

[**RAJESH KUMAR**](https://www.devopsschool.com/blog/author/rajeshkumar/)**JUNE 27, 2023**[**LEAVE A COMMENT**](https://www.devopsschool.com/blog/helm-lab-5-creating-local-remote-helm-repository/#respond)

Creating Local helm Repository

|  |  |
| --- | --- |
|  |  |
|  |  |
|  | # test deploying Chart Museum from the custom repository |
|  | # offical repository for chartmuseum is |
|  | # https://artifacthub.io/packages/helm/chartmuseum/chartmuseum |
|  |  |
|  | helm repo add chartmuseum https://chartmuseum.github.io/charts |
|  | helm install chartmuseum chartmuseum/chartmuseum --dry-run --debug |
|  |  |
|  |  |
|  |  |
|  | # deploy Chart Museum |
|  | helm install chartmuseum chartmuseum/chartmuseum --set env.open.DISABLE\_API=false |
|  | # Ref - helm install my-chartmuseum chartmuseum/chartmuseum --version 3.10.1 |
|  |  |
|  |  |
|  |  |
|  | # view release |
|  | helm list |
|  |  |
|  |  |
|  |  |
|  | # view Chart Museum pod ans its labels |
|  | kubectl get pods --show-labels |
|  |  |
|  |  |
|  |  |
|  | # grab the Chart Museum pod name using the label retrieved from the previous command |
|  | # Powershell |
|  | $POD\_NAME=$(kubectl get pods -l "app.kubernetes.io/name=chartmuseum" -o jsonpath="{.items[0].metadata.name}") |
|  |  |
|  | #Linux |
|  | POD\_NAME=$(kubectl get pods -l "app.kubernetes.io/name=chartmuseum" -o jsonpath="{.items[0].metadata.name}") |
|  | echo $POD\_NAME |
|  |  |
|  |  |
|  |  |
|  | # set up port forwarding |
|  | # this would be accessible only by 127.0.0.1 |
|  | kubectl port-forward $POD\_NAME 8080:8080 --namespace default |
|  |  |
|  | # this would be accessible by NodeIP of the host where u ran a command |
|  | kubectl port-forward --address 0.0.0.0 $POD\_NAME 8080:8080 --namespace default |
|  |  |
|  |  |
|  |  |
|  | # confirm Chart Museum is up and running |
|  | # http://127.0.0.1:8080 |
|  |  |
|  |  |
|  |  |
|  | # view Helm help to get location of Helm Configuration Path |
|  | helm --help |
|  |  |
|  |  |
|  |  |
|  | # navigate to folder containing repository names and URLs, e.g. - |
|  | # C:\Users\apruski\AppData\Roaming\helm\ |
|  |  |
|  |  |
|  |  |
|  | # add Chart Museum as a Helm repository |
|  | helm repo add chartmuseum http://127.0.0.1:8080 |
|  |  |
|  |  |
|  |  |
|  | # list repositories |
|  | helm repo list |
|  |  |
|  |  |
|  |  |
|  | # view updated repositories.yaml file |
|  | # C:\Users\apruski\AppData\Roaming\helm\ |
|  |  |
|  |  |
|  |  |
|  | # naviage to packaged Chart |
|  | cd C:\Charts |
|  |  |
|  |  |
|  |  |
|  | # confirm curl |
|  | curl --help |
|  |  |
|  |  |
|  |  |
|  | # if using powershell v5.1 you'll need to remove |
|  | # the alias that maps curl to Invoke-WebRequest |
|  | Remove-Item alias:\curl |
|  |  |
|  |  |
|  |  |
|  | # push chart to Chart Museum |
|  | # a place where u have ourchart-0.1.0.tgz |
|  | cd /home/ubuntu/charts |
|  | curl --data-binary "@ourchart-0.1.0.tgz" http://localhost:8080/api/charts |
|  |  |
|  |  |
|  |  |
|  | # search for the chart |
|  | helm search repo chartmuseum/ourchart |
|  |  |
|  |  |
|  |  |
|  | # view helm help again to get Helm Cache Path |
|  | helm --help |
|  |  |
|  |  |
|  |  |
|  | # view cached chartmuseum-index.yaml file for repository |
|  | # C:\Users\apruski\AppData\Local\Temp\helm\repository |
|  |  |
|  |  |
|  |  |
|  | # update Helm repositories |
|  | helm repo update |
|  |  |
|  |  |
|  |  |
|  | # view updated chartmuseum-index.yaml file for repository |
|  | # C:\Users\apruski\AppData\Local\Temp\helm\repository |
|  |  |
|  |  |
|  |  |
|  | # search Chart Museum for our chart |
|  | helm search repo chartmuseum/ourchart |
|  |  |
|  |  |
|  |  |
|  | # we can also see the chart in the Helm section of |
|  | # of the Kubernetes extension for vs code |
|  |  |
|  |  |
|  |  |
|  | # deploy the chart from Chart Museum |
|  | helm install ourchart chartmuseum/ourchart |
|  |  |
|  |  |
|  |  |
|  | # view release |
|  | helm list |

[view raw](https://gist.github.com/devops-school/f63c1d91735ad4b85bd47a610dd8abb4/raw/2a4dc087249fe32948a856f799ea738fc3e184de/local-helm.sh)[local-helm.sh](https://gist.github.com/devops-school/f63c1d91735ad4b85bd47a610dd8abb4#file-local-helm-sh)hosted with  by [GitHub](https://github.com/)

|  |  |
| --- | --- |
|  | ################################################## |
|  | # go to Github and create a repository |
|  | ################################################## |
|  |  |
|  |  |
|  |  |
|  | # clone repository locally |
|  | git clone https://github.com/dbafromthecold/DemoHelmRepo.git |
|  |  |
|  |  |
|  |  |
|  | # navigate to repo |
|  | cd C:\git\DemoHelmRepo |
|  |  |
|  |  |
|  |  |
|  | # copy packaged chart into repo |
|  | cp C:\charts\ourchart-0.1.0.tgz . |
|  |  |
|  |  |
|  |  |
|  | # index repo |
|  | helm repo index . |
|  |  |
|  |  |
|  | # view index.yaml |
|  | cat index.yaml |
|  |  |
|  | # push chart to Github |
|  | git config user.name "Rajesh Kumar" |
|  | git congig user.email "devops@rajeshkumar.xyz" |
|  | git add . |
|  | git commit -m "added ourchart to repo" |
|  | git push |
|  |  |
|  | ################################################## |
|  | # get URL from index.yaml file in GitHub repository |
|  | ################################################## |
|  |  |
|  | # add Githb repo as a Helm repository |
|  | helm repo add dbafromthecold https://raw.githubusercontent.com/dbafromthecold/DemoHelmRepo/master |
|  | helm repo add rajesh https://raw.githubusercontent.com/devopsschool-demo-temporary/demp4helm/main |
|  |  |
|  | # navigate to folder containing repository names and URLs |
|  | # C:\Users\apruski\AppData\Roaming\helm\ |
|  |  |
|  | helm env |
|  | cd /root/.config/helm |
|  | cd /root/.cache/helm |
|  | more repositories.yaml |
|  | cd repository/ |
|  | more rajesh-index.yaml |
|  |  |
|  | # view cached index.yaml file for repository |
|  | # C:\Users\apruski\AppData\Local\Temp\helm\repository |
|  |  |
|  | # search new Helm repository |
|  | helm search repo dbafromthecold/ourchart |
|  |  |
|  | # we can also see the chart in the Helm section of |
|  | # of the Kubernetes extension for vs code |
|  |  |
|  | # deploy chart from repository |
|  | helm install ourchart dbafromthecold/ourchart |
|  |  |
|  | # confirm deployment |
|  | helm list |

# Helm Tutorials: Complete tutorials of Helm Hooks

[**RAJESH KUMAR**](https://www.devopsschool.com/blog/author/rajeshkumar/)**JUNE 28, 2023**[**LEAVE A COMMENT**](https://www.devopsschool.com/blog/helm-tutorials-complete-tutorials-of-helm-hooks/#respond)

## Step 1: Understanding Helm Hooks

Helm hooks are a feature introduced in Helm 2.0 that allows you to perform actions during the release lifecycle of a Helm chart. Hooks can be used to execute scripts, run commands, or trigger other actions at specific points in the release process. Helm supports several types of hooks, including pre-install, post-install, pre-upgrade, post-upgrade, pre-delete, and post-delete hooks.

Hooks is a helm feature that allows helm to launch resources into the cluster in response to helm lifecycle events (such as a chart being installed). All it takes to turn a regular Kubernetes object into a helm hook is to apply an annotation that indicates which lifecycle event the object should be tied to.

Helm hooks provide a way to apply certain Kubernetes resources (usually Jobs) at a particular stage of a deployment. Most developers leverage hooks to:

* Load a ConfigMap or Secret during install before any other charts are loaded
* Execute a Job to back up a database before installing a new chart and a second job to restore the data in the database
* Run a job before deleting a release to gracefully take out a service before completely removing it

Hook works like a regular template, but they have special annotations that cause helm to utilize them differently. In this section, we will cover the basic usage pattern for hooks.

We can define Helm hooks that will be executed at these points in the release’s lifecycle:

* **pre-install** – Before the release is installed.
* **post-install** – After the release is installed
* **pre-delete** – Before the release is uninstalled.
* **post-delete**– After the release is uninstalled.
* **pre-upgrade** – Before the release is upgraded.
* **post-upgrade** – After the release is upgraded.
* **pre-rollback** – Before the release is rolled back to a previous version.
* **post-rollback** – After the release is rolled back to a previous version.

# Helm Hook

Hooks is a helm feature that allows helm to launch resources into the cluster in response to helm lifecycle events (such as a chart being installed). All it takes to turn a regular Kubernetes object into a helm hook is to apply an annotation that indicates which lifecycle event the object should be tied to.

Helm hooks provide a way to apply certain Kubernetes resources (usually Jobs) at a particular stage of a deployment. Most developers leverage hooks to:

* Load a ConfigMap or Secret during install before any other charts are loaded
* Execute a Job to back up a database before installing a new chart and a second job to restore the data in the database
* Run a job before deleting a release to gracefully take out a service before completely removing it

Hook works like a regular template, but they have special annotations that cause helm to utilize them differently. In this section, we will cover the basic usage pattern for hooks.

## Step 2: Creating a Helm Chart

Before diving into hooks, let’s assume you have a basic understanding of creating Helm charts. If you’re new to Helm, it’s recommended to familiarize yourself with chart creation first. You can refer to the official Helm documentation for creating a simple chart.

## Step 3: Adding a Pre-Install Hook

To add a pre-install hook to your Helm chart, you need to create a templates directory in your chart’s root directory (if it doesn’t exist already). Inside the templates directory, create a YAML file for your hook. For example, let’s create a file named pre-install-hook.yaml.

In the pre-install-hook.yaml file, define the hook using the following structure:

apiVersion: batch/v1

kind: Job

metadata:

name: {{ include "mychart.fullname" . }}-pre-install-hook

labels:

app.kubernetes.io/managed-by: {{ .Release.Service | quote }}

helm.sh/hook: pre-install

helm.sh/hook-weight: "1"

helm.sh/hook-delete-policy: hook-succeeded

spec:

template:

spec:

containers:

- name: {{ .Chart.Name }}

image: busybox

command: ['sh', '-c', 'echo "Running pre-install hook"']

restartPolicy: Never

In this example, we define a Kubernetes Job that runs a container with the busybox image and executes a simple command.

## Step 4: Adding a Post-Install Hook

Similar to the pre-install hook, you can add a post-install hook to your chart. Create a new YAML file in the templates directory, for example, post-install-hook.yaml.

In the post-install-hook.yaml file, define the hook using the following structure:

apiVersion: batch/v1

kind: Job

metadata:

name: {{ include "mychart.fullname" . }}-post-install-hook

labels:

app.kubernetes.io/managed-by: {{ .Release.Service | quote }}

helm.sh/hook: post-install

helm.sh/hook-weight: "1"

helm.sh/hook-delete-policy: hook-succeeded

spec:

template:

spec:

containers:

- name: {{ .Chart.Name }}

image: busybox

command: ['sh', '-c', 'echo "Running post-install hook"']

restartPolicy: Never

This hook follows a similar structure as the pre-install hook but with a different name and the helm.sh/hook set to post-install.

## Step 5: Updating the Chart’s hooks Section

Now that we have defined our hooks, we need to update the chart’s hooks section in the templates/hooks.yaml file. If the file doesn’t exist, create it.

Add the following code to the hooks.yaml file:

{{- if .Values.hooks.preInstall.enabled -}}

preInstall:

- name: {{ include "mychart.fullname" . }}-pre-install

manifest: {{ include "mychart.pre

# Helm Lab – 7: Managing Dependencies of Helm Charts

[**RAJESH KUMAR**](https://www.devopsschool.com/blog/author/rajeshkumar/)**JUNE 28, 2023**[**LEAVE A COMMENT**](https://www.devopsschool.com/blog/helm-lab-7-managing-dependencies-of-helm-charts/#respond)

Managing dependencies is an essential aspect of creating and maintaining Helm charts, especially when building complex applications composed of multiple services or components. Here’s how you can manage dependencies of Helm charts effectively:  
  
**For managing interdependent charts.**

## How to Define Helm Chart Dependencies?

In your Helm chart, specify the dependencies in either the requirements.yaml file (for Helm 2) or the Chart.yaml file (for Helm 3).

1. Chart.yaml
2. requirements.yaml

$ helm repo update  
$ helm repo add bitnami https://charts.bitnami.com/bitnami  
$ helm repo add stable https://charts.helm.sh/stable

## Example of Chart.yaml

apiVersion: v1

name: mychart

version: 1.0.0

dependencies:

- name: mysql

version: 9.12.1

repository: https://charts.bitnami.com/bitnami

- name: jenkins

version: 2.5.4

repository: https://charts.helm.sh/stable

- name: my-third-chart

version: 0.1.0

repository: "file:///home/ubuntu/charts/chart-dep/ourchart"

## Example of a Chart.yaml file that defines dependencies:

apiVersion: v1

name: mychart

version: 1.0.0

dependencies:

- name: my-first-chart

version: 1.0.0

repository: file://../my-first-chart

- name: my-second-chart

version: 1.0.0

repository: https://artifacthub.io/

- name: my-third-chart

version: 1.0.0

repository: file://../my-third-chart

## Example of requirements.yaml

$ helm repo update

$ helm repo add bitnami https://charts.bitnami.com/bitnami

$ helm repo add stable https://charts.helm.sh/stable

dependencies:

- name: mysql

version: 9.12.1

repository: https://charts.bitnami.com/bitnami

- name: jenkins

version: 2.5.4

repository: https://charts.helm.sh/stable

- name: my-third-chart

version: 0.1.0

repository: "file:///home/ubuntu/charts/chart-dep/ourchart"

$ helm dependency update  
$ helm dependency list

## ****Update Dependencies****:

$ helm dependency update  
$ helm dependency list

Run the helm dependency update command in the directory containing your Helm chart. This command downloads the specified dependencies and stores them in the charts/ directory within your chart’s directory.

helm dependency update:

This command updates the dependencies for a chart. For example, the following command will update the dependencies for the chart mychart:

$ helm dependency update mychart

helm dependency list:

This command lists the dependencies for a chart. For example, the following command will list the dependencies for the chart mychart:

$ helm dependency list mychart

helm dependency build:

This command builds the dependencies for a chart. This is useful if you are developing a chart and want to test it with the latest versions of its dependencies. For example, the following command will build the dependencies for the chart mychart:

$ helm dependency build mychart

helm dependency pull:

This command pulls the dependencies for a chart. This is useful if you want to download the dependencies for a chart without installing them. For example, the following command will pull the dependencies for the chart mychart:

$ helm dependency pull mychart

helm dependency upgrade:

This command upgrades the dependencies for a chart to the latest version. For example, the following command will upgrade the dependencies for the chart mychart to the latest version:

$ helm dependency upgrade mychart

helm dependency list-all:

This command lists all the dependencies for all the charts in a directory. For example, the following command will list all the dependencies for all the charts in the current directory:

$ helm dependency list-all

helm dependency remove:

This command removes a dependency from a chart. For example, the following command will remove the dependency my-chart from the chart my-app:

$ helm dependency remove my-app my-chart