

**Lab Manual- Download helm chart**

**Prepared for**:

**Date:** 18th Dec 2023

**Prepared by:**

Document Name: Lab Manual **Document Number** AZLabn916

**Contributor:**

Contents

[1. Objective 3](#_Toc156410958)

[2. helm pull ngnix and mysql 3](#_Toc156410959)

[download a chart from a repository and (optionally) unpack it in local directory 3](#_Toc156410960)

[3. Install local down loaded chart 5](#_Toc156410961)

[4. Remove the deployment 6](#_Toc156410962)

# Objective

Retrieve a package from a package repository, and download it locally.

This is useful for fetching packages to inspect, modify, or repackage. It can also be used to perform cryptographic verification of a chart without installing the chart.

There are options for unpacking the chart after download. This will create a directory for the chart and uncompress into that directory.

If the --verify flag is specified, the requested chart MUST have a provenance file, and MUST pass the verification process. Failure in any part of this will result in an error, and the chart will not be saved locally.

# helm pull ngnix and mysql

# download a chart from a repository and (optionally) unpack it in local directory

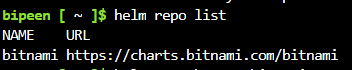
1. Type below command to add bitnami repo

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



1. List the added repo

helm repo list



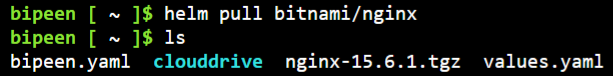
1. Now pull the chart for Ngnix

helm pull bitnami/nginx



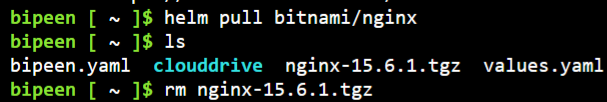
1. Ls will list the Tar file which downloaded , now we can simply untar it

ls



1. Let’s simply remove the Tar file and we will use one of flags –untar=true to download the untar version

rm nginx-15.6.1.tgz



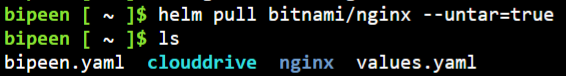
1. Now pull the chart for Ngnix again with flag --untar=true

helm pull bitnami/nginx --untar=true



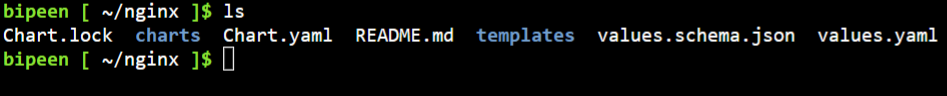
1. Type ls and you will notice **Ngnix Directory**

ls



1. Now go inside the **Ngnix** Directory and list the files

ls



And this is actually the full structure of the source code for a chart. Now when we run **Helm instal** or in fact, any of the helm commands, it uses all of the information in here, it crunches that together, it actually passes it through a template process. And out of that will

come standard Kubernetes yaml, that can be applied to a cluster.

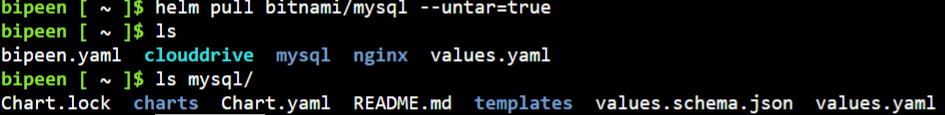
1. Now pull the another char mysql

helm pull bitnami/mysql --untar=true



1. Now list the **Ngnix** Directory

ls mysql/



Note:

Pull a private chart from a repository using credentials:

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

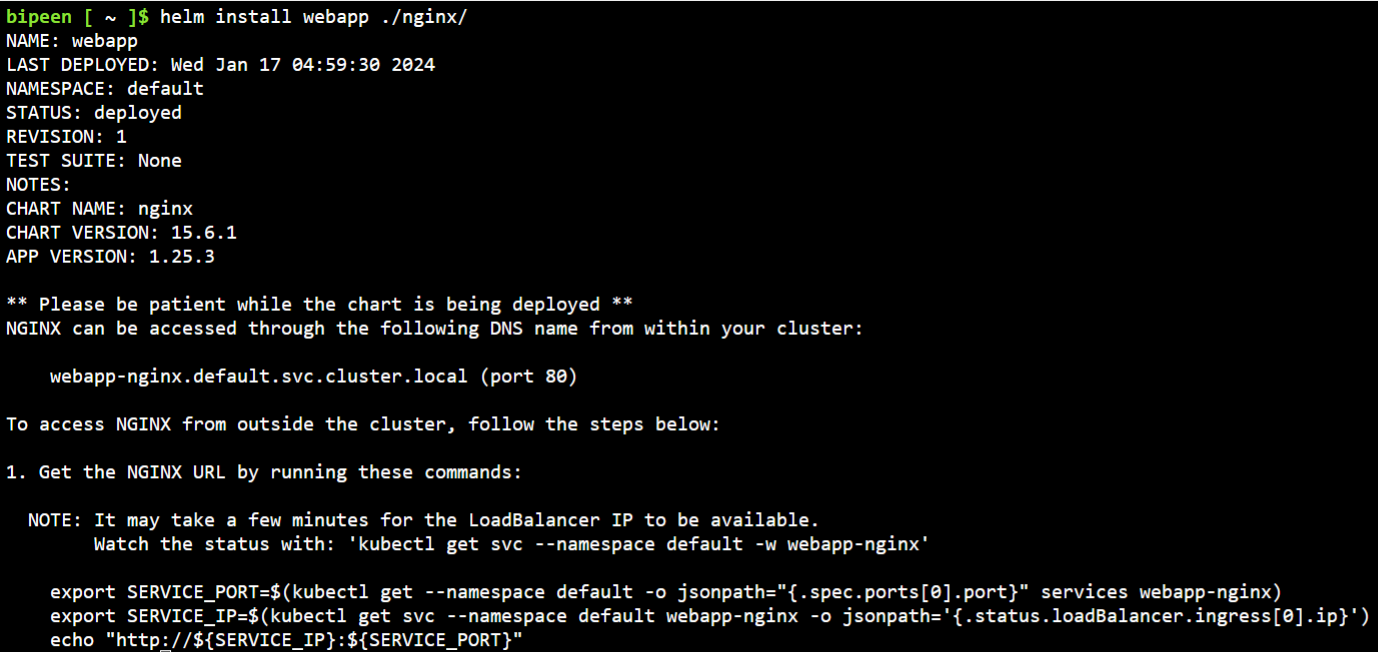
# Install local down loaded chart

Now go ahead and use Helm install, but Helm install on these local downloaded copies instead. How do we do that? Well, it's very simple, I would just do Helm install, as before, I would need to give the installation a name, so I'll call it monitoring. And now instead of using a repository name, I can just use my local folder. and it will use the contents of that folder to run through its processing to generate genuine Kubernetes yaml that it can

apply to my cluster. Let's see if that works.

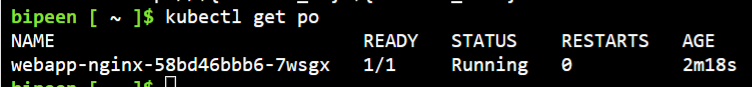
* Run below command to deploy the local downloads chart

helm install webapp ./nginx/



* Check the pod deployed

kubectl get po

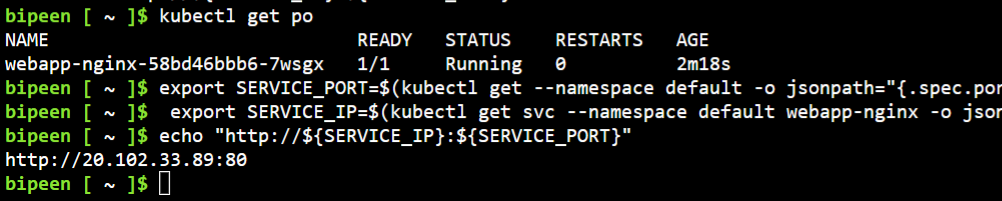


* Run below command to get the public ip on running ninix

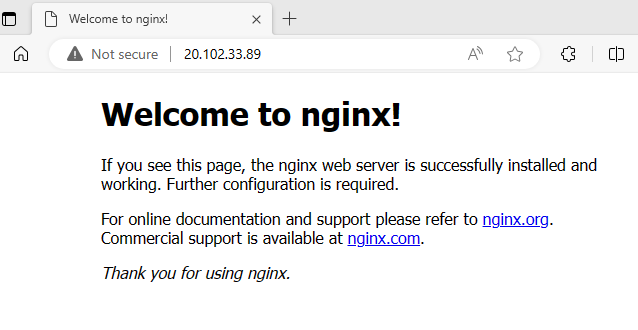
export SERVICE\_PORT=$(kubectl get --namespace default -o jsonpath="{.spec.ports[0].port}" services webapp-nginx)

export SERVICE\_IP=$(kubectl get svc --namespace default webapp-nginx -o jsonpath='{.status.loadBalancer.ingress[0].ip}')

echo "http://${SERVICE\_IP}:${SERVICE\_PORT}"



* Copy the IP and paste in browser



# Remove the deployment

helm list

helm uninstall webapp

