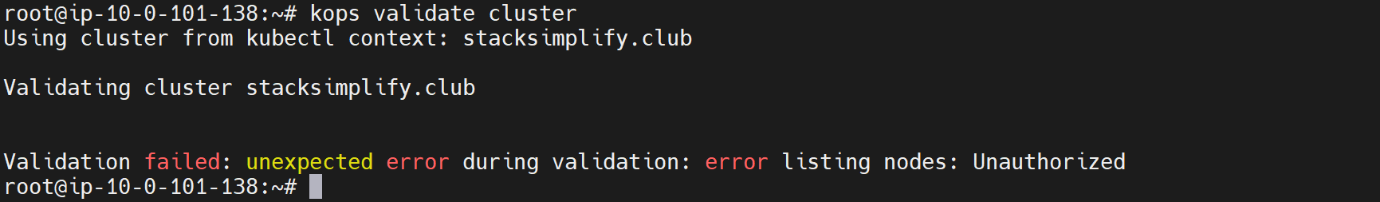
**0. important commands**

--- Reference - <https://bitnami.com/stacks>

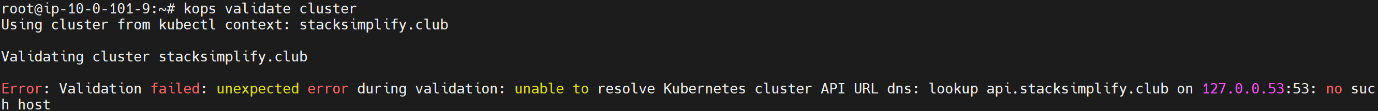
**Error1**

--- kops export kubecfg (cluster name) --admin

--- **kops validate cluster**



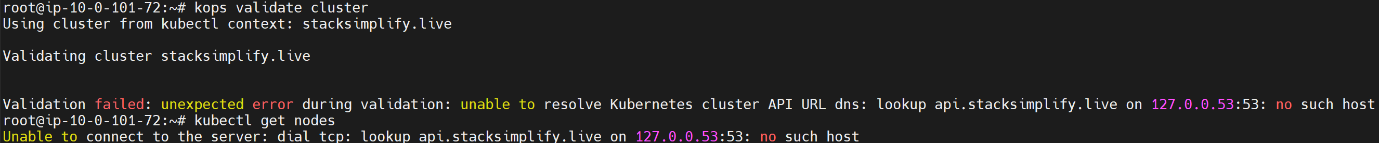
--- **kops export kubecfg stacksimplify.live --admin**



--- **note** – if you facing unexpected error during validation then execute command.

**Error2**

--- kops validate cluster



--- kubectl config use-context stacksimplify.live

**Helm workflow**

Stage1 - it will load charts

Stage2 - substitute the values through values.yml

Stage3 - render the required kubernetes templates and formats those templates into yml

Stage4 – submit the yaml files to the kubernetes.

**Install Helm**

--- Reference - <https://helm.sh/docs/intro/install/>

--- Helm is written in Google Go programming language, which is a compiled language so you can download the binaries for your operating system and start using it.

--- the recommended and easy way is to use the packaging manager for any operating system.

**Install on MacOS**

--- **Reference** - <https://helm.sh/docs/intro/install/>

**# Install helm on MacOS**

--- brew install helm

**Install on windows**

--- **Reference** - <https://helm.sh/docs/intro/install/>

**# Install helm on windows**

--- choco install kubernetes-helm

**Install on Debian/ubuntu (apt)**

--- **Reference** - <https://helm.sh/docs/intro/install/>

**# Install on 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

--- You should have homebrew installed.

**Install on fedora (dnf/yum)**

--- **Reference** - <https://helm.sh/docs/intro/install/>

**# Install on fedora (dnf/yum)**

--- sudo dnf install helm

**Helm commands**

**# Add bitnami repository to your repositories.**

--- helm repo add bitnami <https://charts.bitnami.com/bitnami>

**# List repos**

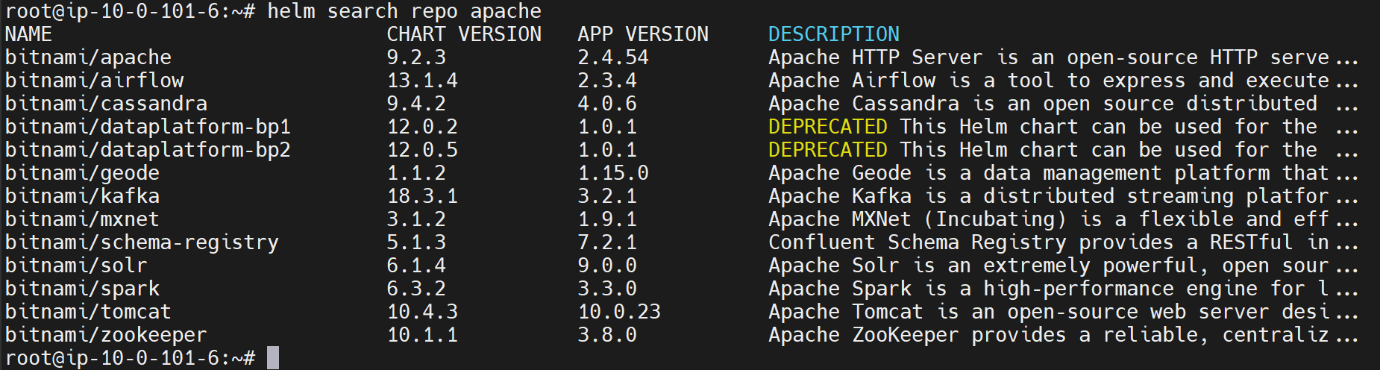
--- helm repo list

**# Lis the installed packaged.**

--- helm list or helm ls

**# Search apache chart in bitnami repo**

--- helm search repo apache



--- **note** – you have chart version and app version.

--- **note** – by default, you will only see the latest version of apache. If you want to see the all the versions of apache then execute the below command.

**# To see the all the version of apache.**

--- helm search repo apache –versions

**Remove repo**

**# Remove the repository**

--- helm repo remove bitnami

**install packages**

**# Install mysql with helm**

--- helm install <dbname> bitnami/mysql

--- helm install mydb bitnami/mysql

**# To see the default configuration information.**

--- helm status mydb

**Uninstall packages**

**# Remove package from namespace.**

--- helm uninstall jenkins

--- helm uninstall mydb

**# Uninstall packages form namespace.**

--- helm uninstall mydb -n <namespace-name>

**Providing mysql with custom values.yml**

**# Create values.yml**

auth:

  rootPassword: "admin123"

**# Install mysql with custom password**

--- helm install mysql bitnami/mysql --values /root/mysql/values.yml

**Upgrade package chart**

**# Upgrade mysql**

--- helm upgrade mysql bitnami/mysql --values /root/mysql/values.yml

**# Reuse the values.yml without giving the file. (Need to use values.yml one time)**

--- helm upgrade mysql bitnami/mysql --reuse-values

**helm --dry-run**

**# Dry run mysql installation using helm**

--- helm install mysql bitnami/mysql --values /root/mysql/values.yml --dry-run

**Helm template**

--- **note** – to overcome the problems we face in the --dry-run command, helm team introduced helm template.

--- **note** – the helm command never communicates with kubernetes api server.

**# Generate template for mysql.**

--- helm template mysql bitnami/mysql --values /root/mysql/values.yml

**Helm get**

**# Get the release notes information of mysql**

--- helm get notes mydb

**# List the custom values you have passed through values.yml**

--- helm get values mysql

root@ip-10-0-101-6:/home/ubuntu# helm get values mysql

USER-SUPPLIED VALUES:

auth:

  rootPassword: admin123

**# List the customized values passed using values.yml in revision**

--- helm get values mysql --revision 1

root@ip-10-0-101-6:/home/ubuntu# helm get values mysql --revision 1

USER-SUPPLIED VALUES:

auth:

  rootPassword: admin123

**# List the manifest for particular revision.**

--- helm get manifest mysql --revision 1

**# List all the values for the chart**

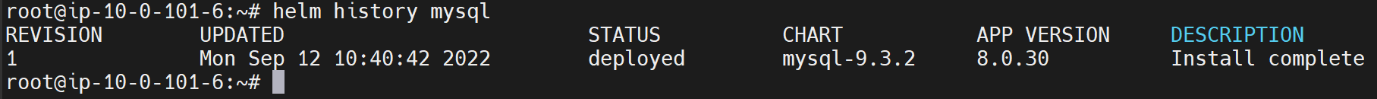
--- helm get values mysql –all

**helm history**

--- **note** - helm history will show us the history of the installations and upgrades.

**# List history of installation**

--- helm history mysql



**Keep history**

**# keep the history of package when you uninstall the package.**

--- helm uninstall mysql --keep-history

--- **important** - if you do not include the --keep-history when you uninstalling the package then the entire history will be gone. You will not be able to roll back or get the installations back.

**create namespace**

--- **note** - To install a package within a namespace, we must create the namespace first and then use the namespace in the installation command.

**# Create name space and install the package in namespace**

--- helm install mywebserver bitnami/apache --namespace mynamespace --create-namespace

**# List the packages in namespace**

--- helm ls -n mynamespace

**install or upgrade**

--- **important** - you will learn how to use the helm upgrade or install command. it will first check if the installation is already there. If it is there it will do the upgrade, otherwise it will do a Install

--- this is very helpful in our CI CD pipeline as the code gets committed to git or any other repository and the CI CD pipeline gets kicked off for the very first time.

**# Install if the package is present otherwise do a upgrade**

--- helm upgrade --install mysql bitnami/mysql

**Wait and Timeout**

--- When we do a helm install, the helm install command considers the installation to be successful as soon as the manifest is received by the kubernetes API server it does not wait for the pods to be up and running if you want that to happen.

--- You can use the **--wait** option when we use --wait. Helm will wait for the services and deployments to be created and the pods should be up and running only then the installation is considered successful.

--- If not, the installation will be considered as a failure by default. It will wait for five minutes or 300 seconds and if the installation does not complete, if the pods are not up and running within that 5m, the installation will be marked as a failure.

**# Install package with wait and timeout parameter**

--- helm install mywebserver bitnami/apache --wait --timeout 5m

**Atomic install**

**# Automatically go back to previous release if the installation is failed.**

--- helm install mywebserver bitnami/apache --atomic --wait --timeout 7m12s

**Forcefully upgrade**

**# Forcefully start the pods when we do upgrade the pods**

--- helm upgrade mywebserver bitnami/apache --force

**Clean Up on failed updates**

**# Clean-up objects, if an upgrade fails**

--- helm upgrade mywebserver bitnami/apache --cleanup-on-failure

**helm package**

--- **note** - you will learn how to package your chart so that it can be distributed or shared through repositories. It can be used to install the software across environments, testing, staging production through our CI CD pipelines as well.

**# Package the chart**

--- helm package firstchart

**# Pull the latest dependencies and package it.**

--- helm package firstchart -u

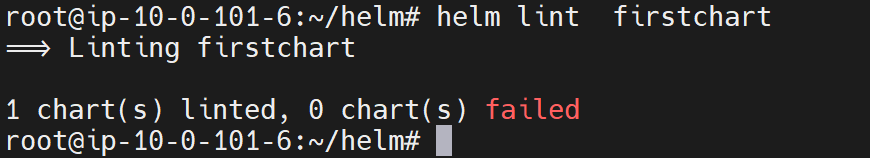
--- **note** - If this chart depends on other charts, it will pull the latest versions of all those charts puts them under this charts folder before the packaging is done.

**helm lint**

--- the Helm Lint Command, the Helm Lint Command will scan through all the template files and the values.yaml files for our chart and it checks if there are any issues, weather there is Syntactical issues are indentation issues in the yaml files to use it,

**# Check for the syntax issues in our chart**

--- helm lint firstchart



--- **note** – it will also check for syntax in zip file as well.

**To see the template of chart**

**# To see the template of firstchart**

--- helm template firstchart