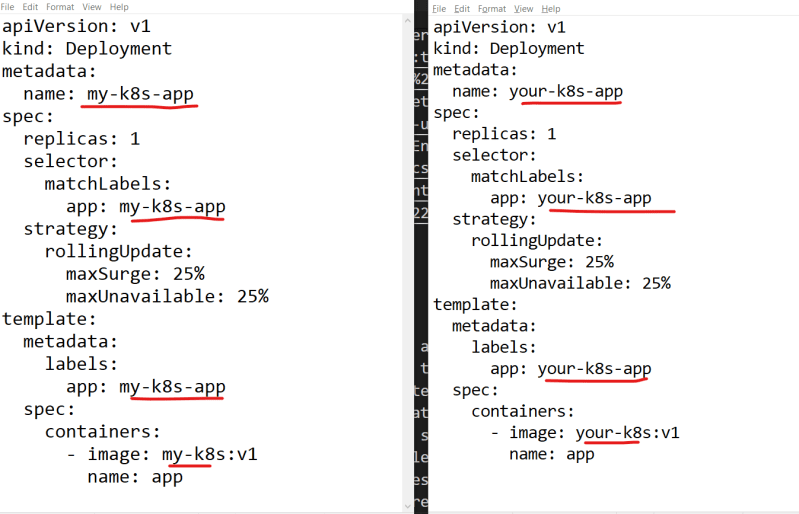
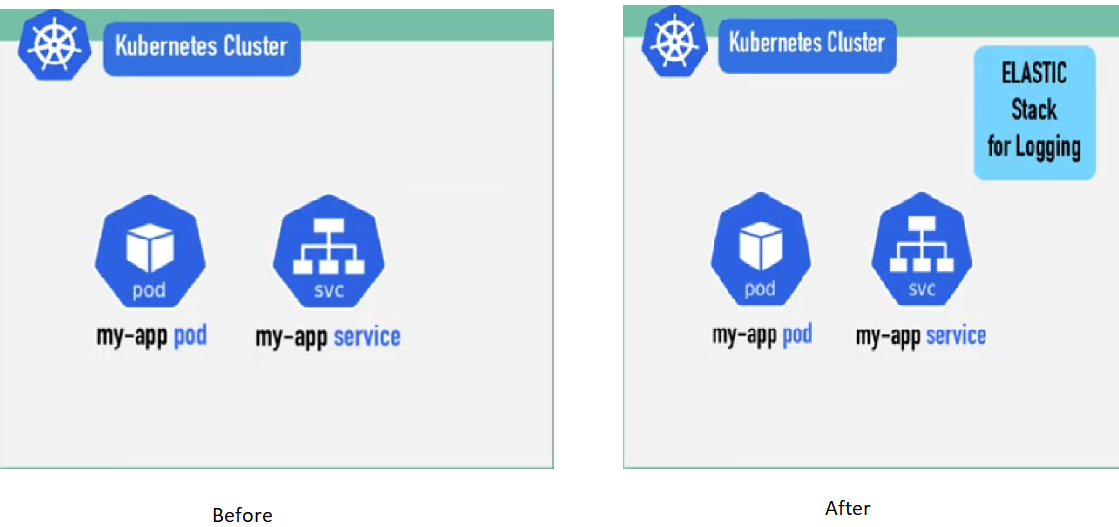
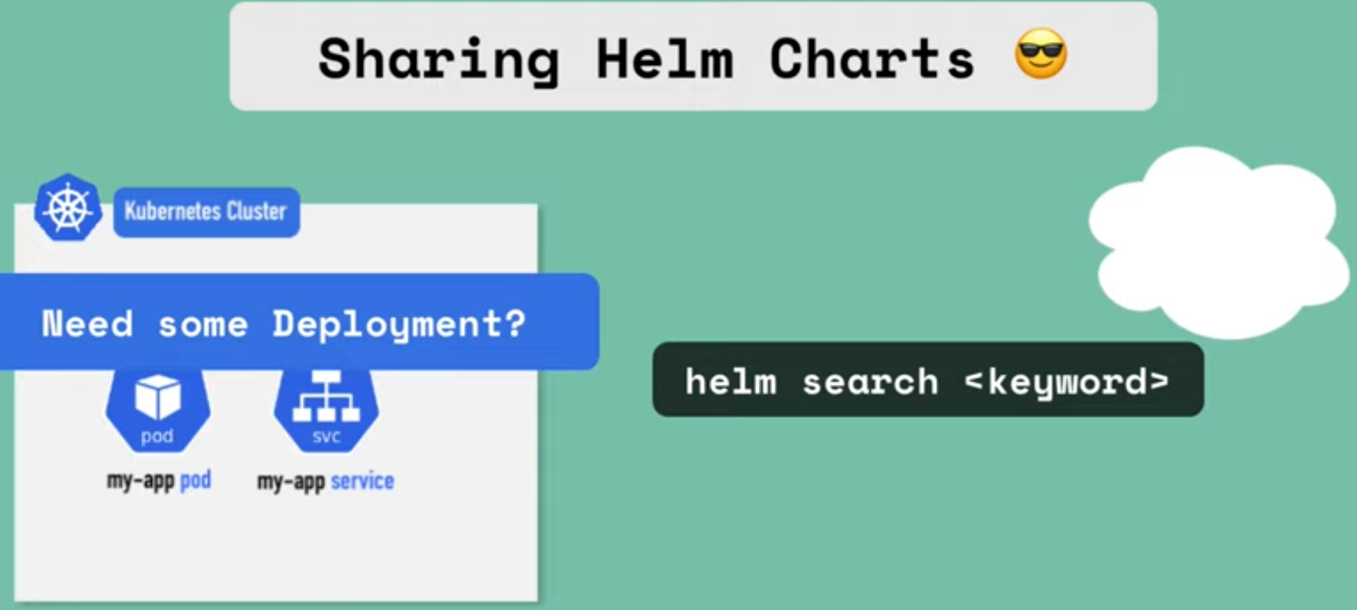
**HELM**

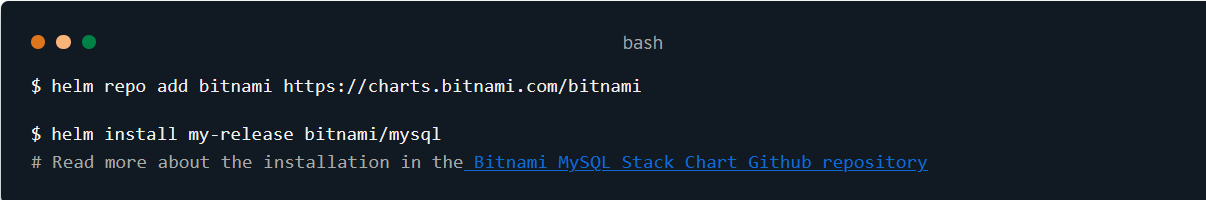


* In order to deploy an application on k8s we need to interact with k8s API to create resources, kubectl is the tool we use to do this
* We express Kubernetes resources in YAML Files
* These files are static in nature.
* Resource files are static:
  + This is the challenge that primarily effects the declarative configuration style of applying YAML resources
  + K8s YAML files are not designed to be parametrized
  + Consider the below two manifests written to deploy two different applications 
  + In the above image each file is almost exactly the same, but we still cannot parametrize
* Helm to the rescue: Helm is an opensource tool used for packaging and deploying applications on k8s. It is often referred as Kubernetes Package Manager.
* Suppose you have a Kubernetes cluster, and you want to install ELK as a side car container, then you need to write stateful set, PV, PVC, services, config map and secrets. 





* Helm was designed to provide an experience similar to that of package manager (apt, yum, dnf etc)
* APT operates on Debian packages and yum/dnf operates on RPM package.
* Helm operates on Charts.
* A Helm Chart contains declarative k8s resource files required to deploy an application
* Helm relies on repositories to provide access to charts
* Chart developers create declarative YAML files, package them into charts and publish them to chart repository
* End users then Helm to search for existing chart to deploy some app on to k8s
* Refer below to view a sample usage of helm chart which installs mysql.
* Link: <https://bitnami.com/stack/mysql/helm>



Let’s try to understand Helm’s subcommands

| **DNF Subcommands** | **Helm Subcommands** | **Purpose** |
| --- | --- | --- |
| install | install | Install an application and its dependencies |
| upgrade | upgrade | Upgrades an application to newer version |
| downgrade | rollback | Reverts the application to previous version |
| remove | uninstall | Delete an application |

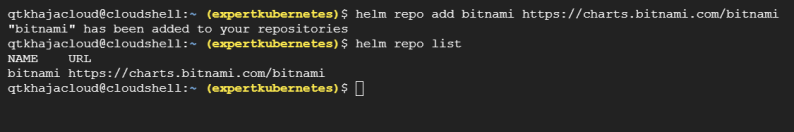
* The abstracted complexity of k8s resources:
  + Let’s assume a developer has been given a task of deploying a MySQL database onto k8s.
  + Developer needs to create resources required to create containers, network and storage
  + With Helm, developer tasked with deploying a mysql database could simply search for MySQL Chart in chart repositories
* Automated Life cycle Hooks:
  + Helm provides the ability to define the life cycle hooks. Lifecycle hooks are actions that take place automatically at different stages of an application’s life cycle.
  + Examples:
    - Perform a data backup on an upgrade
    - Restore data on rollback
    - Validate k8s environment prior to installation

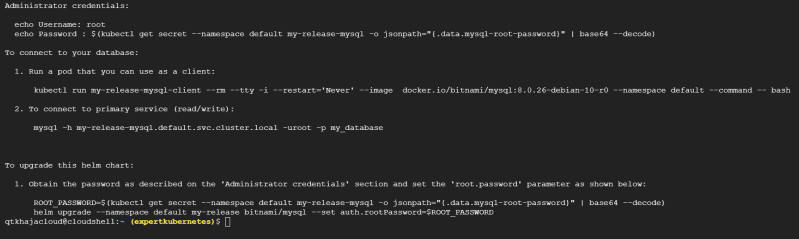
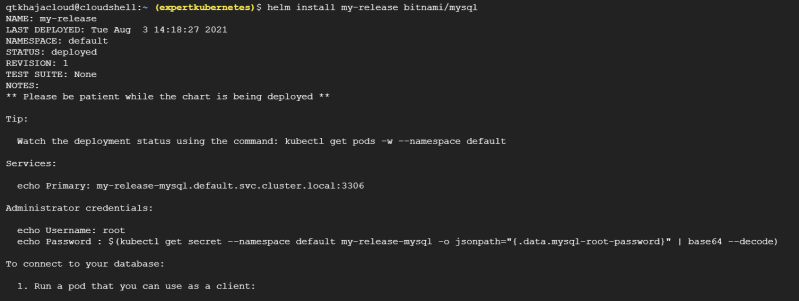
**Installing Helm** Link: <https://helm.sh/docs/intro/install/>

$ 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

**Configuring Helm**

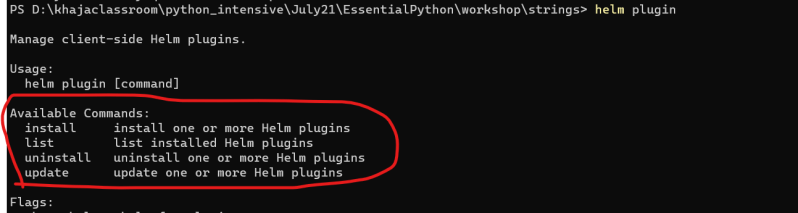
* Helm is a tool with sensible default settings that allow users to be productive without needing to perform a large of post-installation tasks
* With that being said there are several options users can change or enable to modify the Helm’s behaviour.
* Adding upstream repositories:
  + Helm provides the **repo** subcommand to allow users to manage configured chart repositories. This subcommand contains additional subcommands
    - **add**: To add a chart repository
    - **list**: To list chart repositories
    - **remove**: To remove the chart repository
    - **update**: To update information on available charts locally from chart repositories
    - **index**: To generate and index file given a directory containing packaged charts
* Example: Lets install MySQL from bitnami repository

Add bitnami repository as upstream 

Install MySQL 

To uninstall MySQL helm uninstall my-release

Adding plugins:

* + Plugins are add-on capabilities that can be used to provide additional features to helm.
  + For managing plugins, helm has a subcommand **plugin** 
  + Refer <https://helm.sh/docs/community/related/> for some plugins
* ENVIRONMENT Variables: Helm relies on the existence of externalized environmental variables to configure low-level options
  + XDG\_CACHE\_HOME: Sets an alternative location for storing cached files
  + XDG\_CONFIG\_HOME: Sets an alternative location for storing helm configuration
  + XDG\_DATA\_HOME: Sets an alternative location for storing Helm Data
  + HELM\_DRIVER: Sets the backend storage driver
  + HELM\_NO\_PLUGINS: Disables the plugins
  + KUBECONFIG: Sets an alternative Kubernetes configuration file
* Link- <https://helm.sh/docs/helm/helm/>
* Helm has the following paths
  + Windows:
    - Cache Path: %TEMP%\helm
    - Configuration Path: %APPDATA%\helm
    - Data Path: %APPDATA%\helm
  + Linux:
    - Cache Path: $HOME/.cache/helm
    - Configuration Path: $HOME/.config/helm
    - Data Path: $HOME/.local/share/helm

