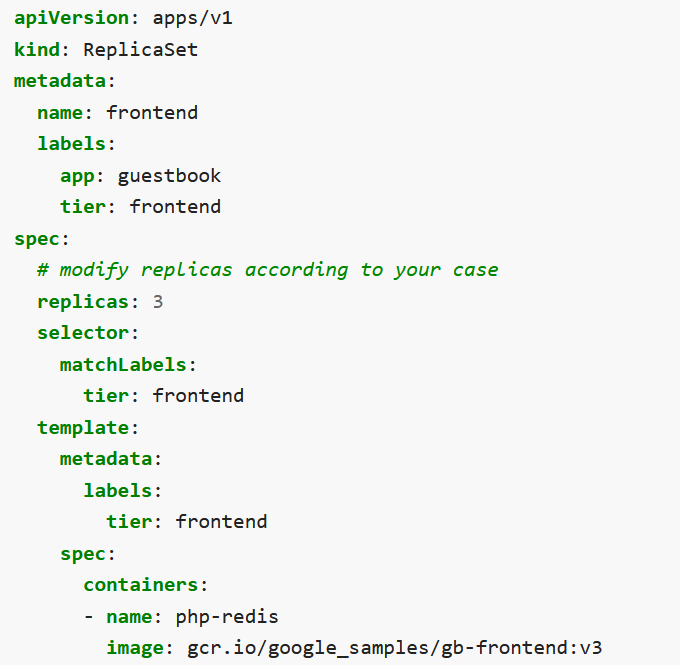
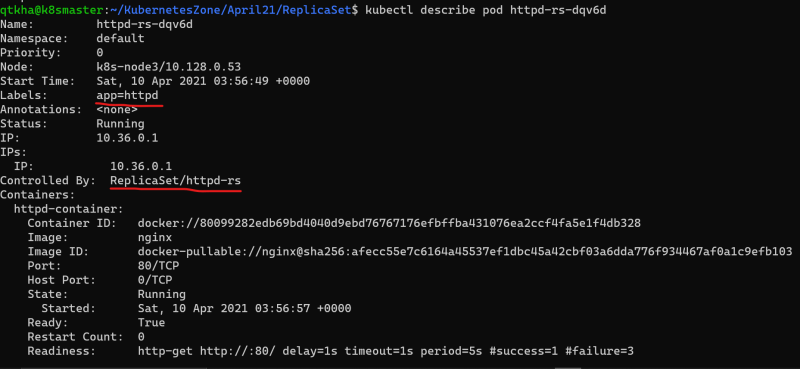
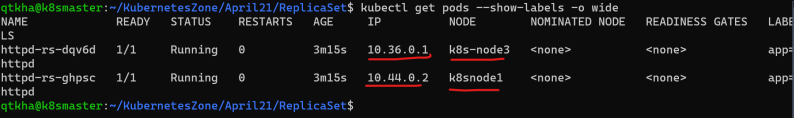
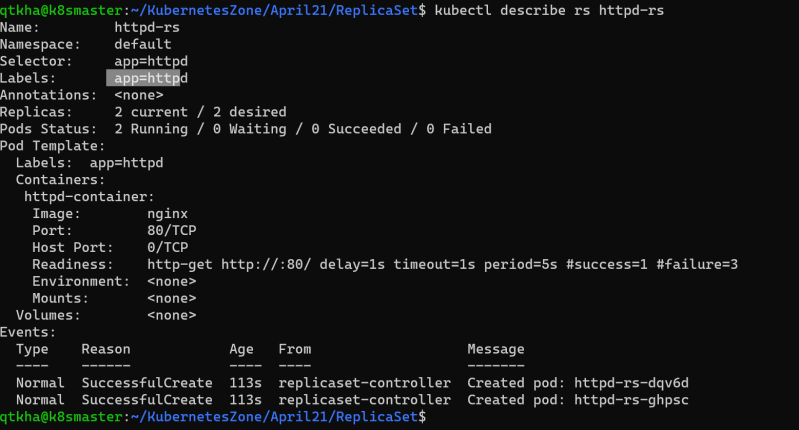
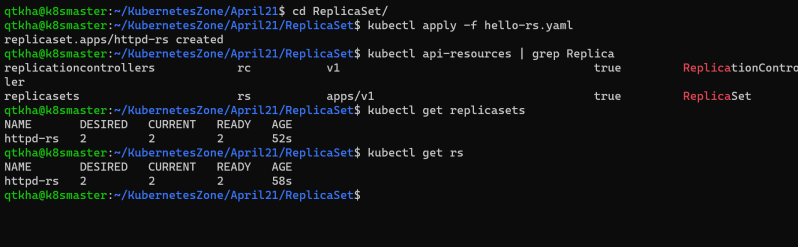
**Kubernetes Controllers**

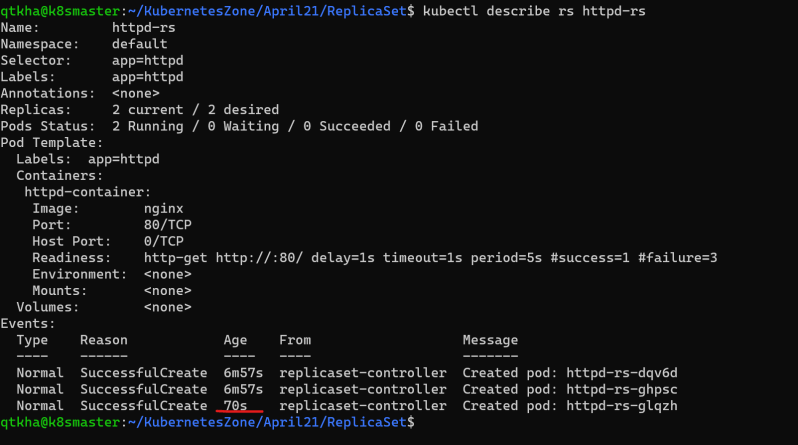
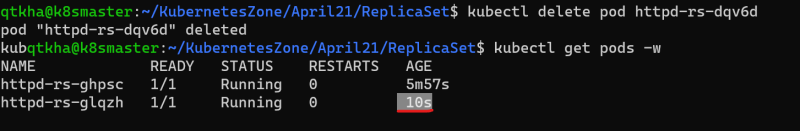
* Pod is the workload in the k8s cluster
* When we deploy our application in production,
  + we might need more than one replica of pod
  + We can balance load across multiple pods so that one pod is not overloaded
* k8s supports different controllers that you can use for replications and maintaining state. The controllers we have are
  + ReplicaSets
  + Replication controller
  + Deployments
  + DaemonSets
  + StatefulSets
  + Jobs
* A controller is an object in k8s that ensures application runs in the desired state for its entire runtime

**ReplicaSet**

* A ReplicaSet is a k8s controller that keeps a certain number of Pods running at any given time
* ReplicaSet acts a supervisor for multiple Pods across different nodes in a k8s cluster. A ReplicaSet will terminate or start new Pods to match the configuration specified in Replicaset Template.



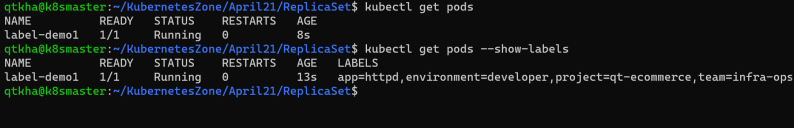


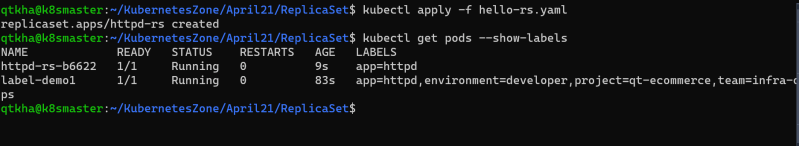
let’s delete one pod manually and see what happens 

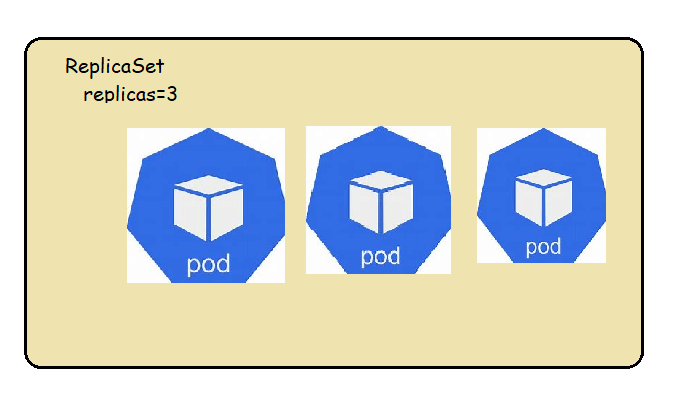
Now lets try to delete the rs 

Now let’s create a pod with app: httpd label and apply this

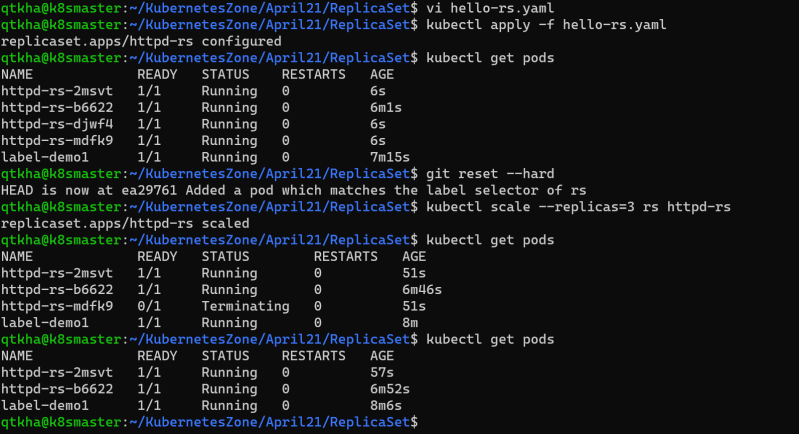




Now lets apply the replica set which needs 2 replicas 

* replica set didn’t create 2 pods rather it created one pod as there was one pod which matched the label selctor mentioned in replica set specification 

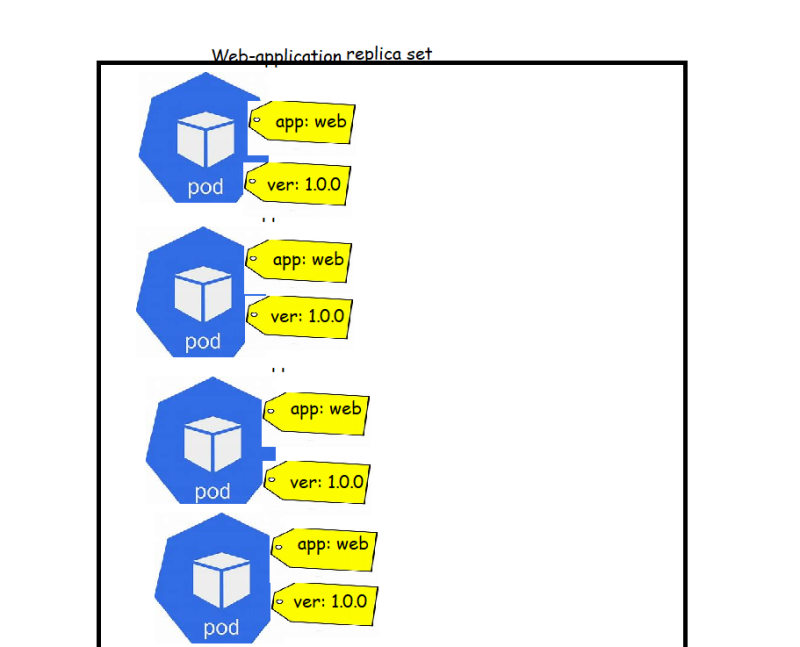
Manually Scaling the replicas

* + change the spec
  + executing the command 

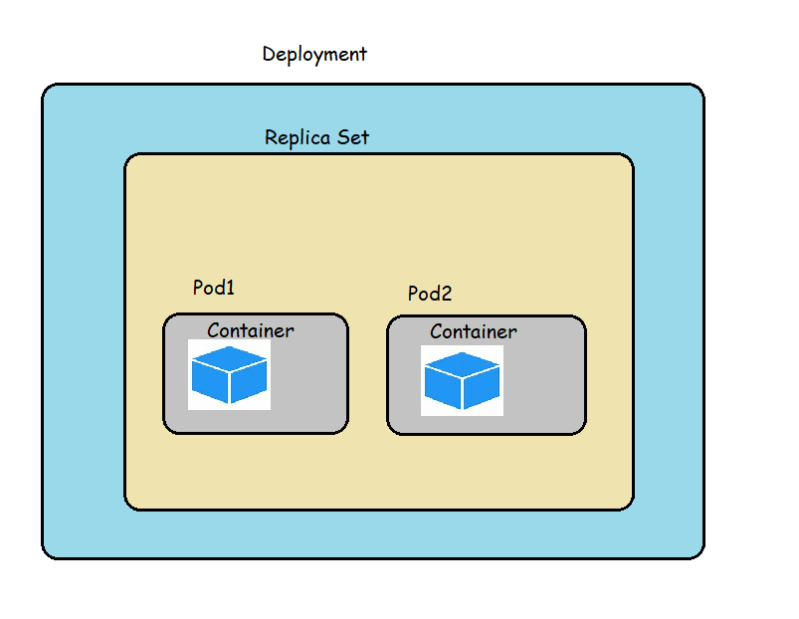
Note: command to create token and display the join command in k8s cluster-

kubeadm token create --print-join-command --ttl=0

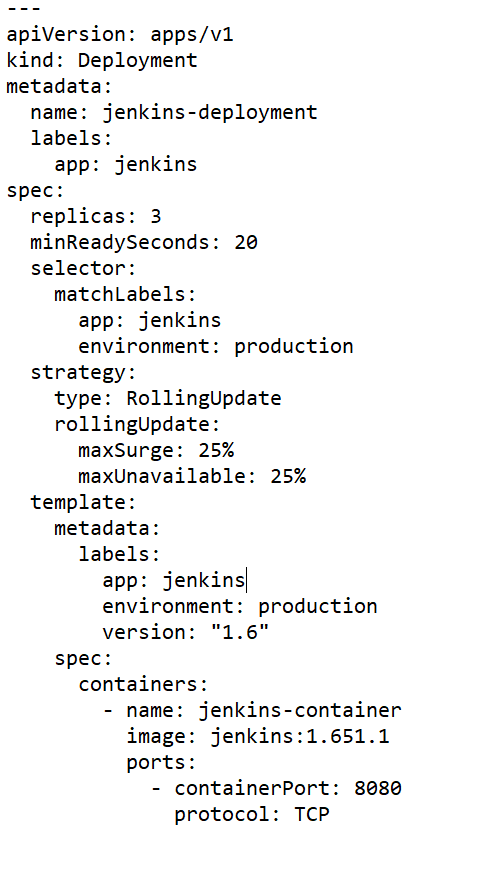
**Introducing the problem**

* Consider we have ReplicaSets where we deployed the application with 4 replicas 
* Now when we get a new release from dev team, we build a new docker image and upload it to the docker hub or any private registry
* How can we apply the new image to ReplicaSets?
  + one approach is to change the ReplicaSets and apply new changes (new docker image) but this will have downtime
* We need a solution where we can apply the new docker images with replications without having downtime
* This can be achieved by Kubernetes Deployment controller object

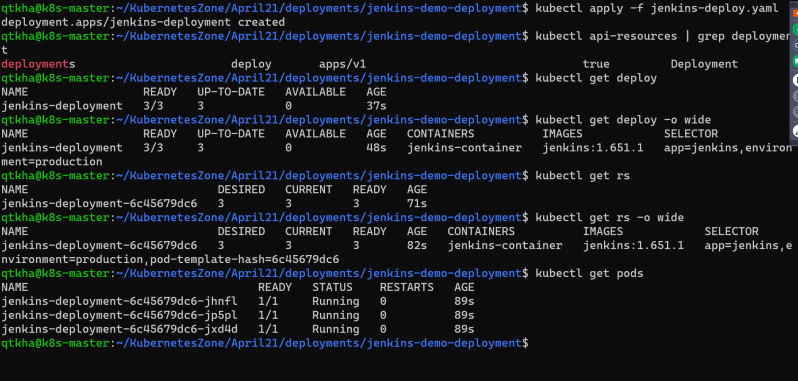
**Deployment**

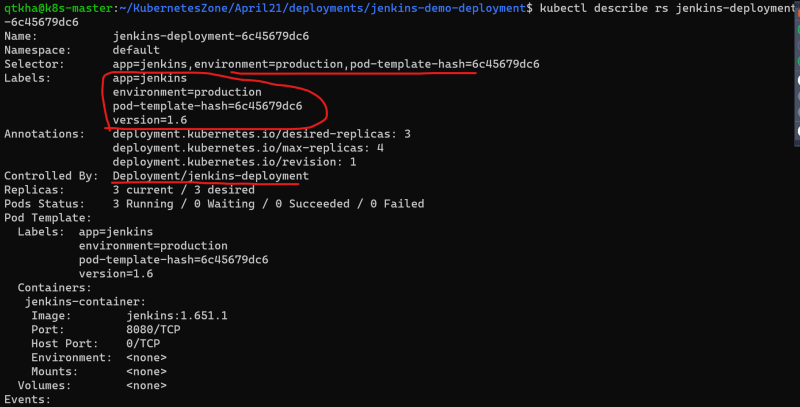
* A deployment is a k8s object that acts a wrapper around Replica Set and makes it easier to use.
* The major motivation for deployments is to maintain the history of revisions.
* Every time a change is made to the replica set or the underlying pods a new version of the ReplicaSets is recorded by Deployment. This way using a Deployment makes it easy to rollback to previous state or version
* Hierarchy of Deployment, ReplicaSets, Pods and container 
* Strategy: In this strategy section we can define strategy to replace old pods with new ones when you are updating. There are two kinds of strategies
  + Rolling Update
  + Recreate
* **RollingUpdate**:
  + We use this strategy to update deployment without having a downtime
  + In Rolling update, we specify two important values
    - maxUnavailable: This is max number of Pods that can be unavailable during the update. default value for maxUnavailable is *25%*
    - maxSurge: this is number of Pods that can be scheduled/created above the desired number of Pods. default value for maxUnavailable is *25%*
* **Recreate**: In this strategy all the existing pods will be killed b4 creating new pods, so there will be some down time

Let’s create a simple deployment

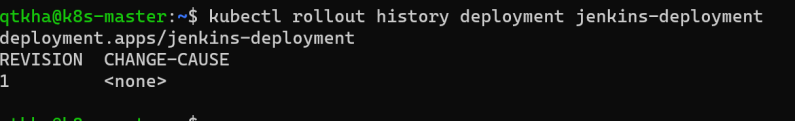


Let’s apply the template





Labels and annotations on the deployment

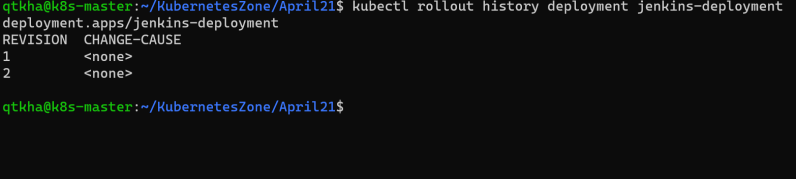
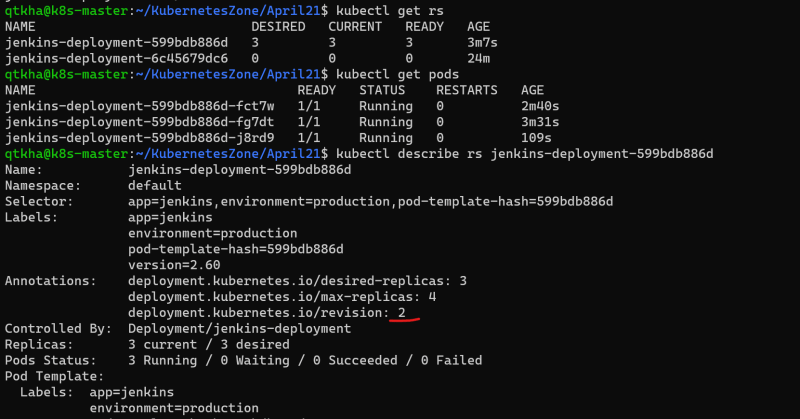
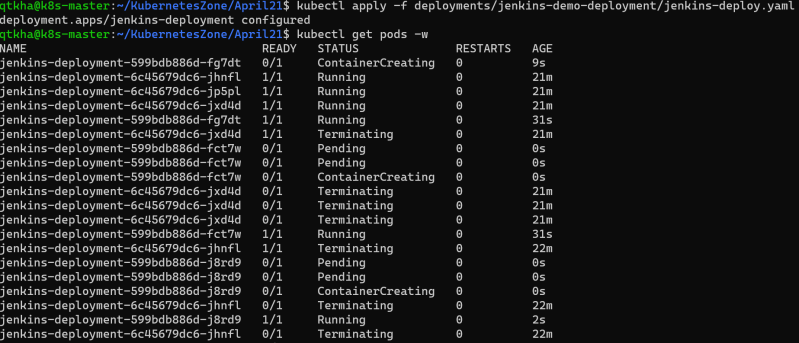
k8s adds an annotation with the deployment.kubernetes.io/revision key which contains information about how many revisions have been there for particular deployment 

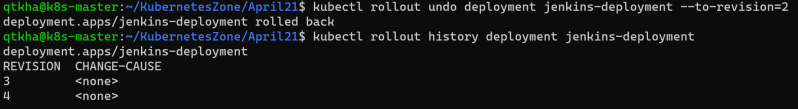
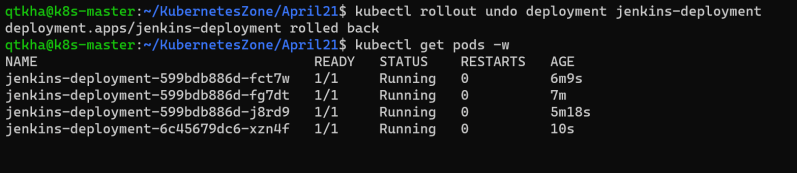
Added a simple service to access Jenkins



Now make some changes to simulate new application version



Now let’s apply the changes using kubectl apply with –record flag. This flag ensures that the update to the deployment is recorded in the history of deployment 

Let’s undo the deployment 

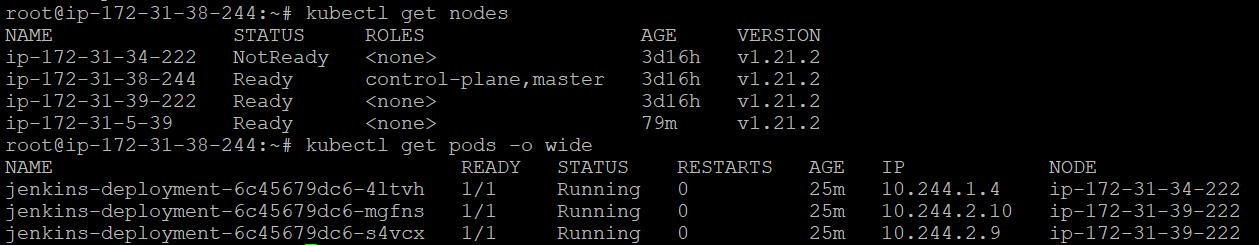
**DaemonSets**

A *DaemonSet* ensures that all (or some) Nodes run a copy of a Pod. As nodes are added to the cluster, Pods are added to them. As nodes are removed from the cluster, those Pods are garbage collected. Deleting a DaemonSet will clean up the Pods it created.

Some typical uses of a DaemonSet are:

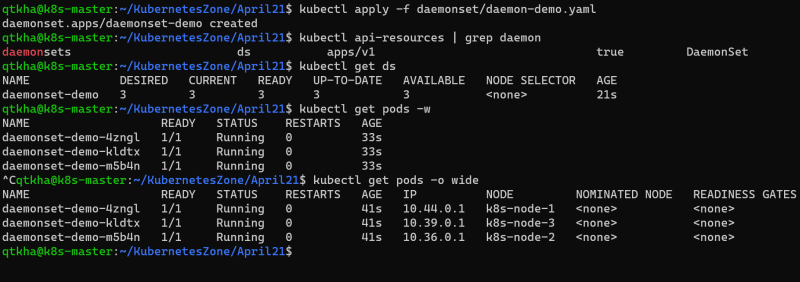
* running a cluster storage daemon on every node
* running a logs collection daemon on every node
* running a node monitoring daemon on every node

Below is the example of creating a pod when no daemon set was present, so all pods are created on a single node.



* DaemonSet are used to manage the creation of particular pod on all or set of selected nodes in a cluster
* Use cases for DaemonSet
  + Logging: To manage running a log collection pod on all nodes and then process the logs in a log processing pipeline (sending logs to central logging server)
  + Local data caching
  + Monitoring
* Let’s write one daemon set

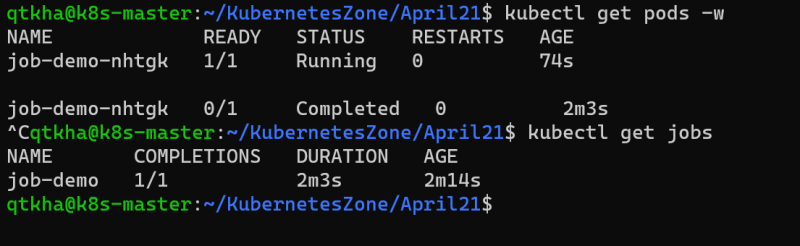
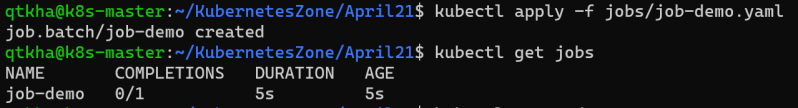


let’s apply the template 

**Jobs**

* A job is supervisor in k8s that can be used to manage pods to run some script/task and exit gracefully
* The pods created by the job are not deleted following the completion of job. The pods run to completion and stay with a Completed status
* Let’s create a k8s job configuration which will run the alpine pod to sleep for 2 minutes (sleep 2m) with restart Policy on Failure



* 

Cron Job: Cronjobs are useful for creating periodic and recurring tasks, like running backups or sending emails. Cronjobs can also schedule individual tasks for a specific time, such as scheduling a Job for when your cluster is likely to be idle.



StatefulSets: There are two type of application (stateful and stateless).

Stateful Application: ELK, MongoDb.

Stateless Application: Calculator

When we write deployment for stateful application, we use stateful sets, and when we write deployment for stateless application we use Deployment.

