TASK 5 : KUBERNETES

Submitted by:-
Aditya Kumar Verma

Guided by:
Ashish Mittal Sir

What is Kubernetes?

Kubernetes, also known as K8s, is an open source system for managing containerized applications across multiple hosts. It provides basic mechanisms for the deployment, maintenance, and scaling of applications.

Kubernetes builds upon a decade and a half of experience at Google running production workloads at scale using a system called Borg, combined with best-of-breed ideas and practices from the community.

Kubernetes is hosted by the Cloud Native Computing Foundation (CNCF). If your company wants to help shape the evolution of technologies that are container-packaged, dynamically scheduled, and microservices-oriented, consider joining the CNCF. For details about who's involved and how Kubernetes plays a role, read the CNCF announcement.

Kubernetes Features

Kubernetes has many features that help orchestrate containers across multiple hosts, automate the management of K8s clusters, and maximize resource usage through better utilization of infrastructure. Important features include:

- **Auto-scaling**

  Automatically scale containerized applications and their resources up or down based on usage.

- **Lifecycle management**

  Automate deployments and updates with the ability to:

  Rollback to previous versions

  Pause and continue a deployment

- **Declarative model**

  Declare the desired state, and K8s works in the background to maintain that state and recover from any failures.

- **Resilience and self-healing**

  Auto placement, auto restart, auto replication and auto scaling provide application self-healing.

- **Persistent storage**

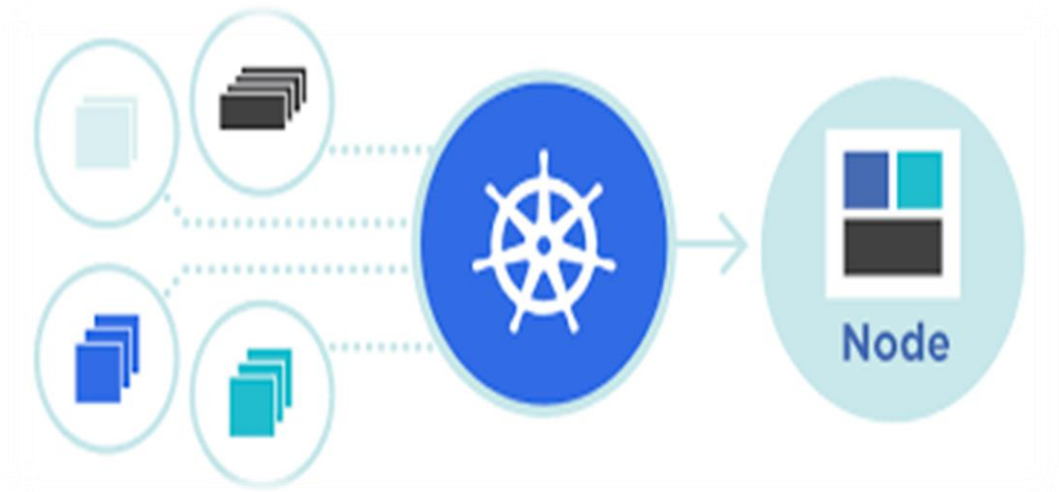
  Ability to mount and add storage dynamically.

- **Load balancing**

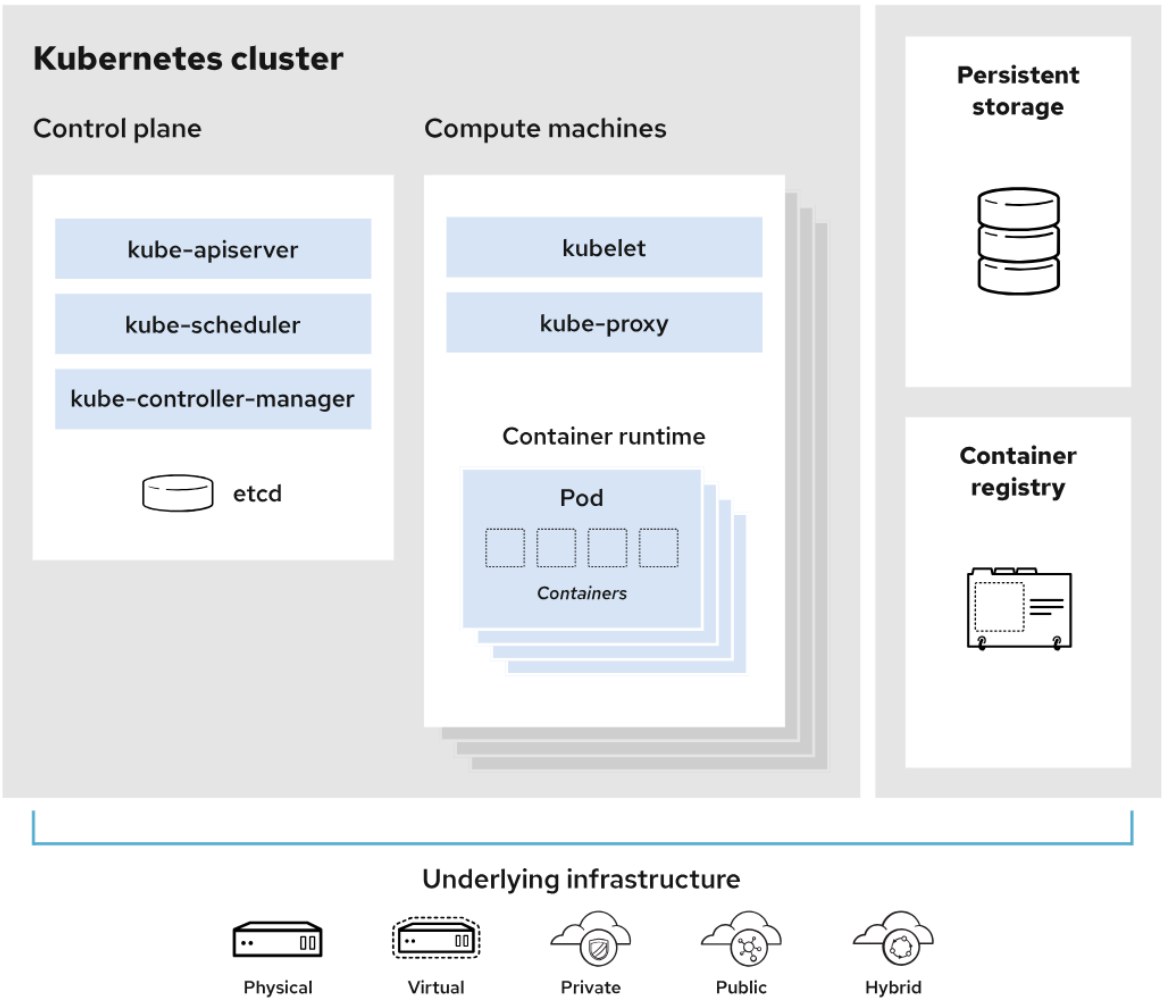
  Kubernetes supports a variety of internal and external load balancing options to address diverse needs.

- **DevSecOps support**

  DevSecOps is an advanced approach to security that simplifies and automates container operations across clouds, integrates security throughout the container lifecycle, and enables teams to deliver secure, high-quality software more quickly. Combining DevSecOps practices and Kubernetes improves developer productivity.



Working of Kubernetes



Task 5 :

- kubernetes:-

- killercoda:-playground
- https://killercoda.com/playgrounds/scenario/kubernetes

- 1 master and one node
  for 60 miutes

- pods 10:-httpd,caddy,nginx
- 5 frpm CLI and 5 from Defination-file.

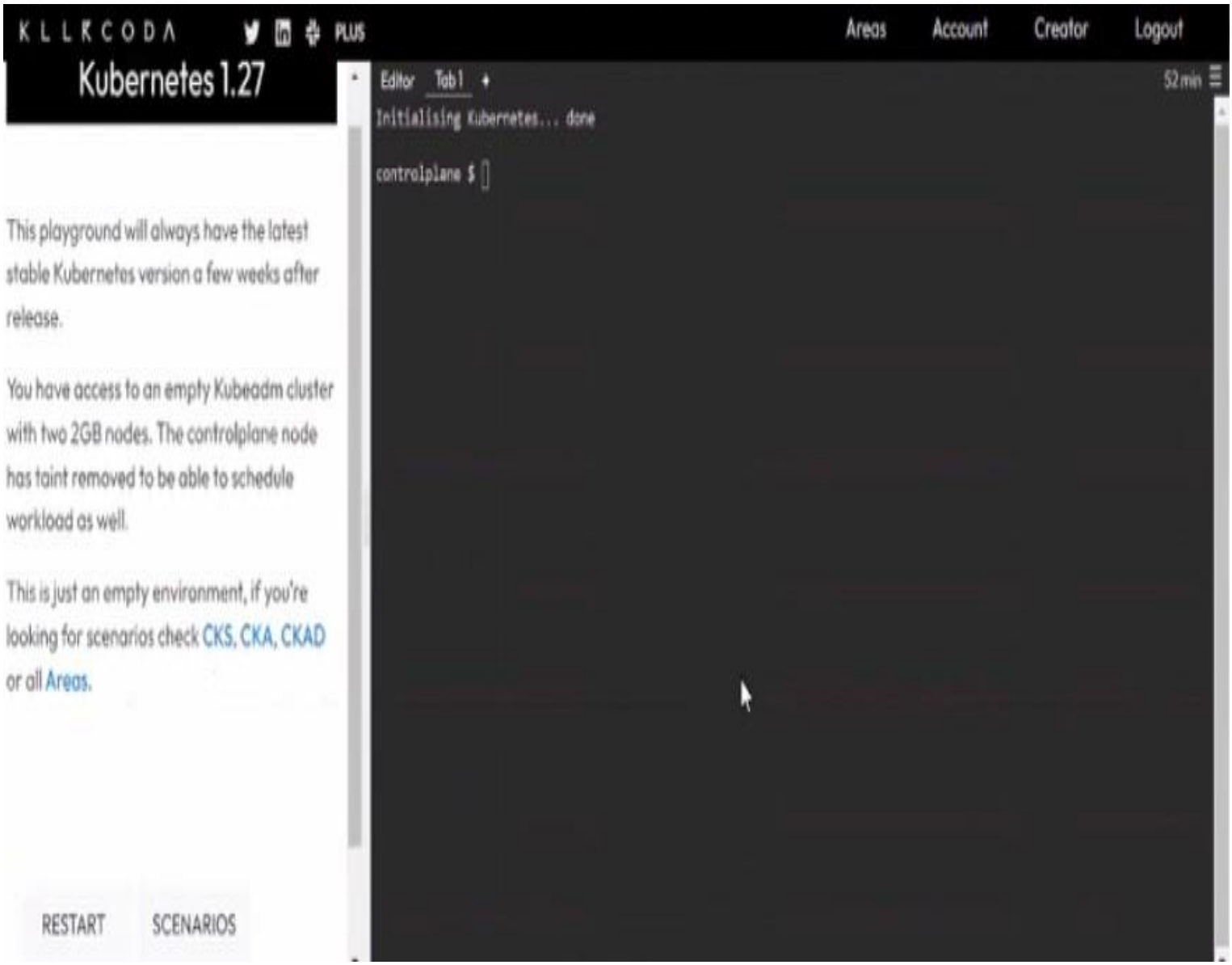
- https://kubernetes.io/docs/concepts/workloads/pods/

- name:-every pod different name
  label(key=value):-5 types

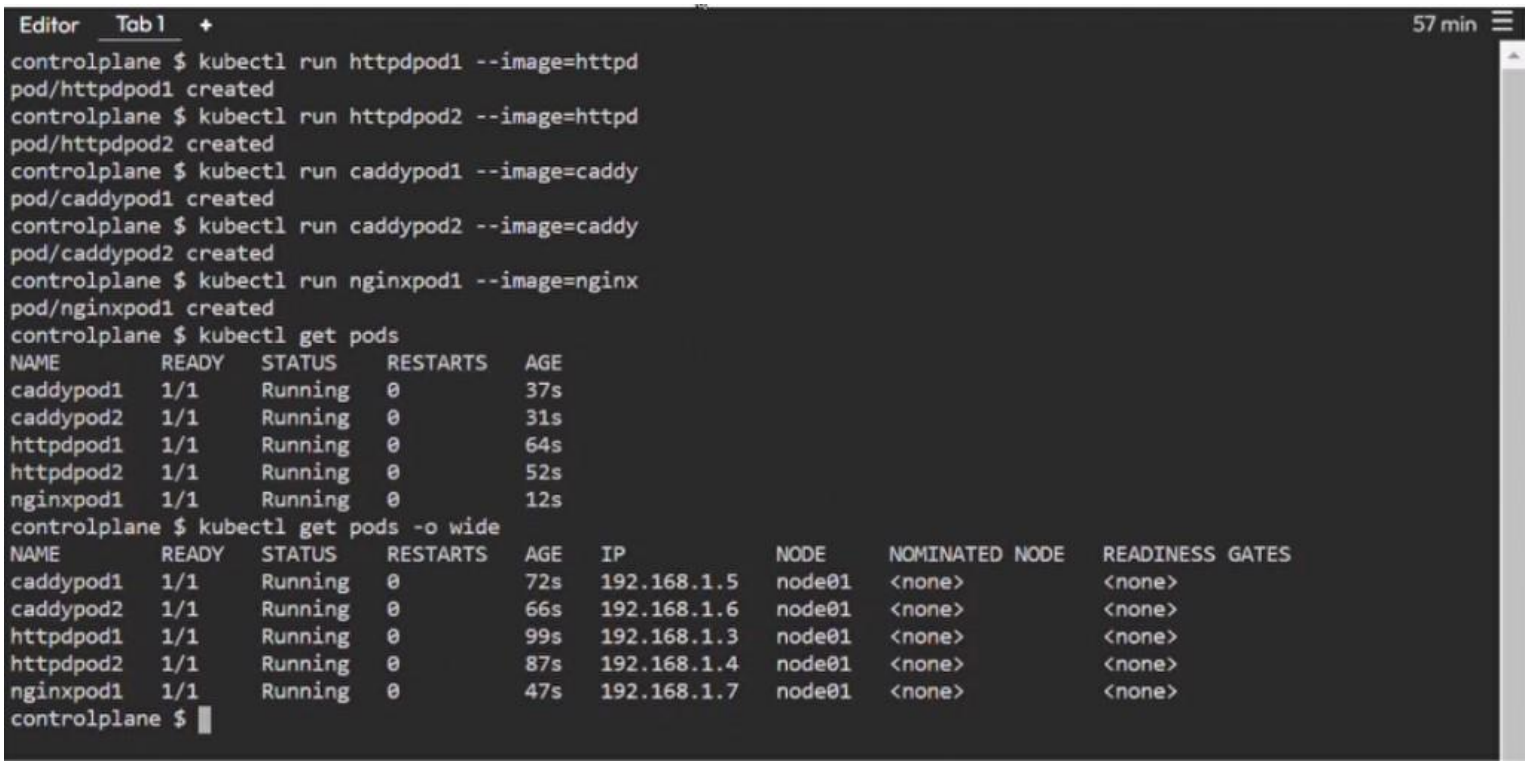
- 5 rc:-5 label(pod-label)
- each rc should have 4 replicas.

- https://kubernetes.io/docs/concepts/workloads/controllers/replicationcontroller/
- deployment:-5

1. Creating a Kubernetes cluster with one master node and one worker node.



2. Deploy 5 pods with different names and using different container images (httpd, caddy, nginx). You can use the “kebuctl run” command to create the pods directly from the command-line interface (CLI):

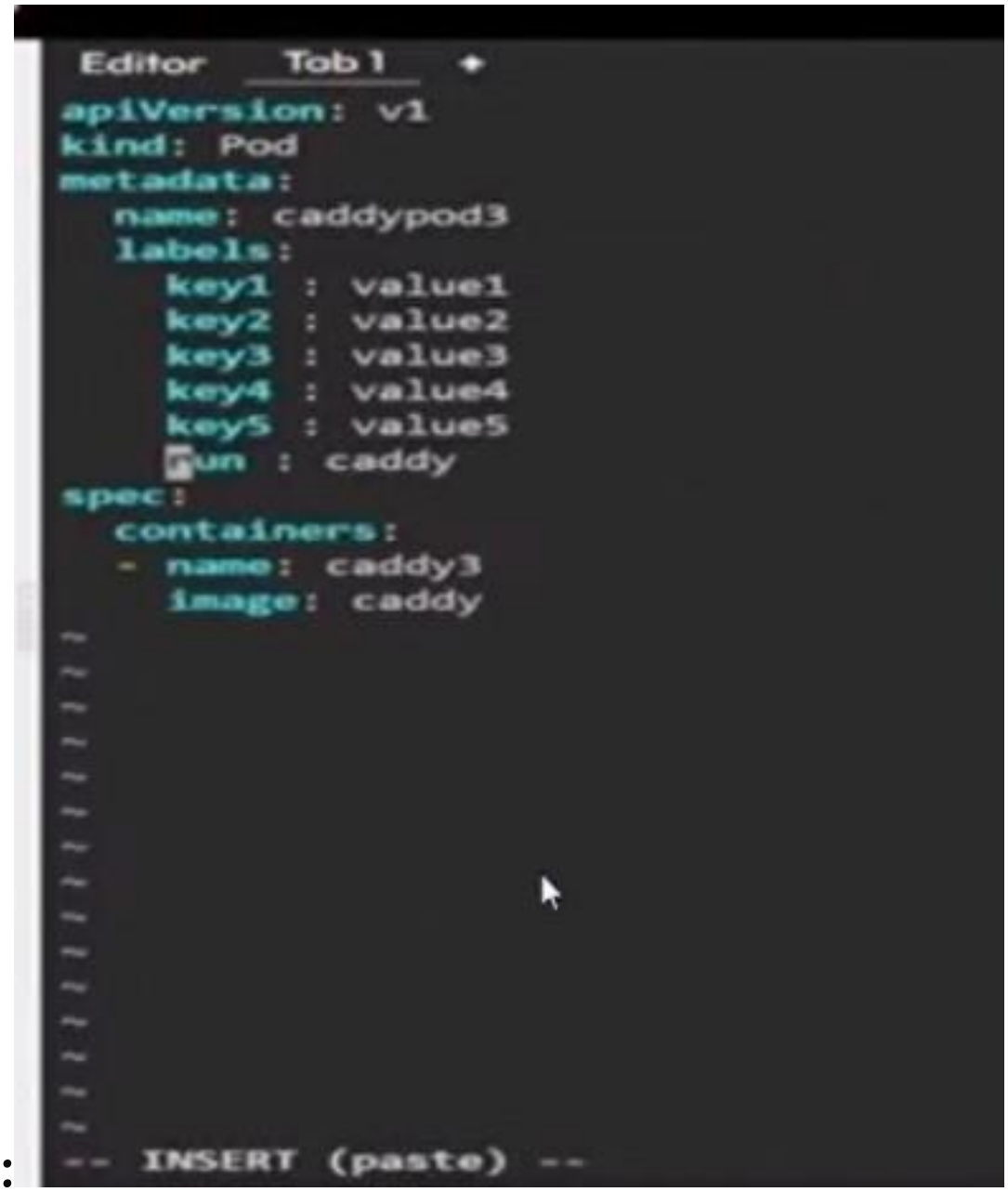


3. For the remaining 5 pots, create a YAML definition file for each pod and use the “kubectl create” command to create the pods for the destination files.

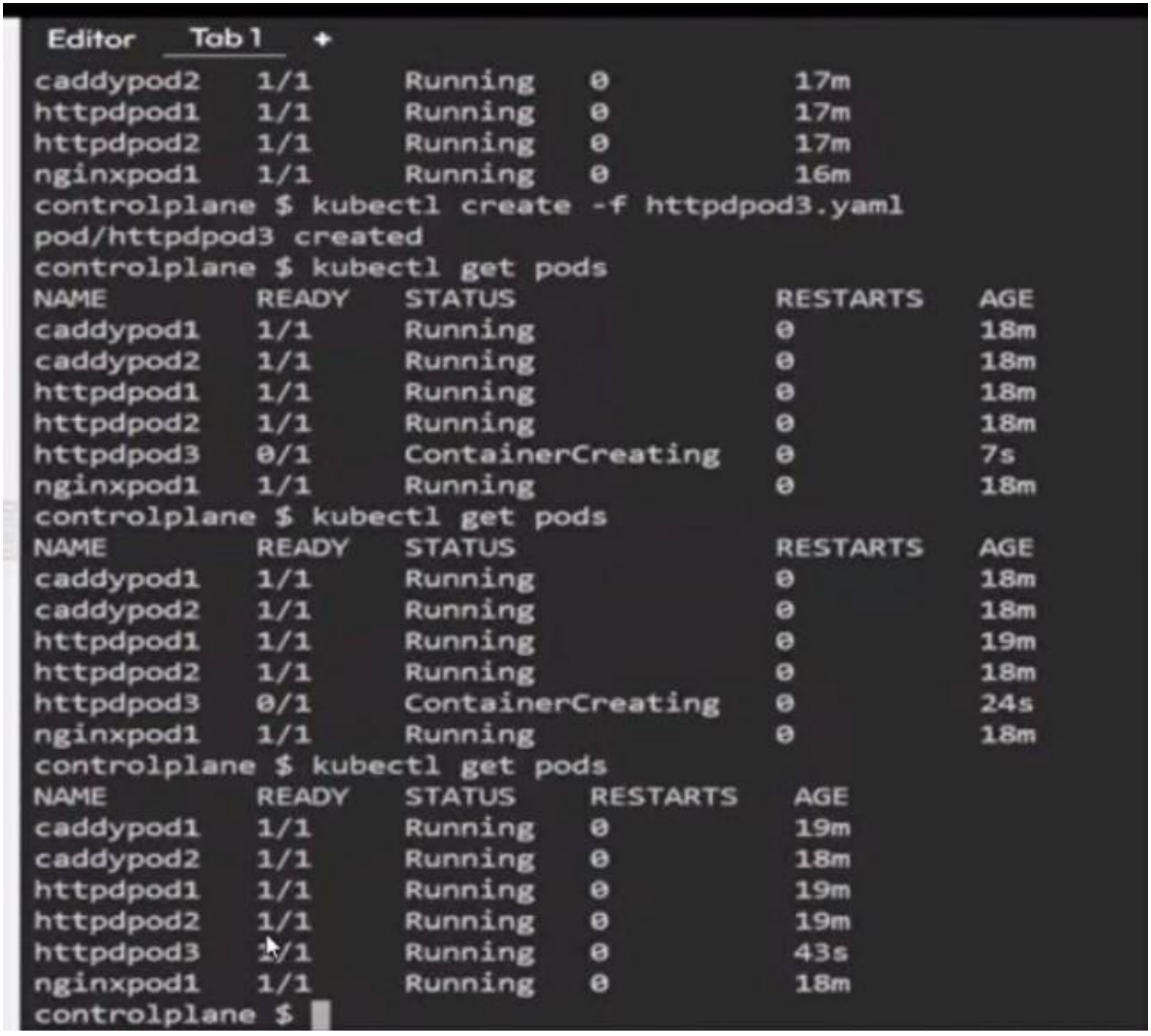
Step 1 :

```
controlplane $ vi httpdpod3.yaml
```

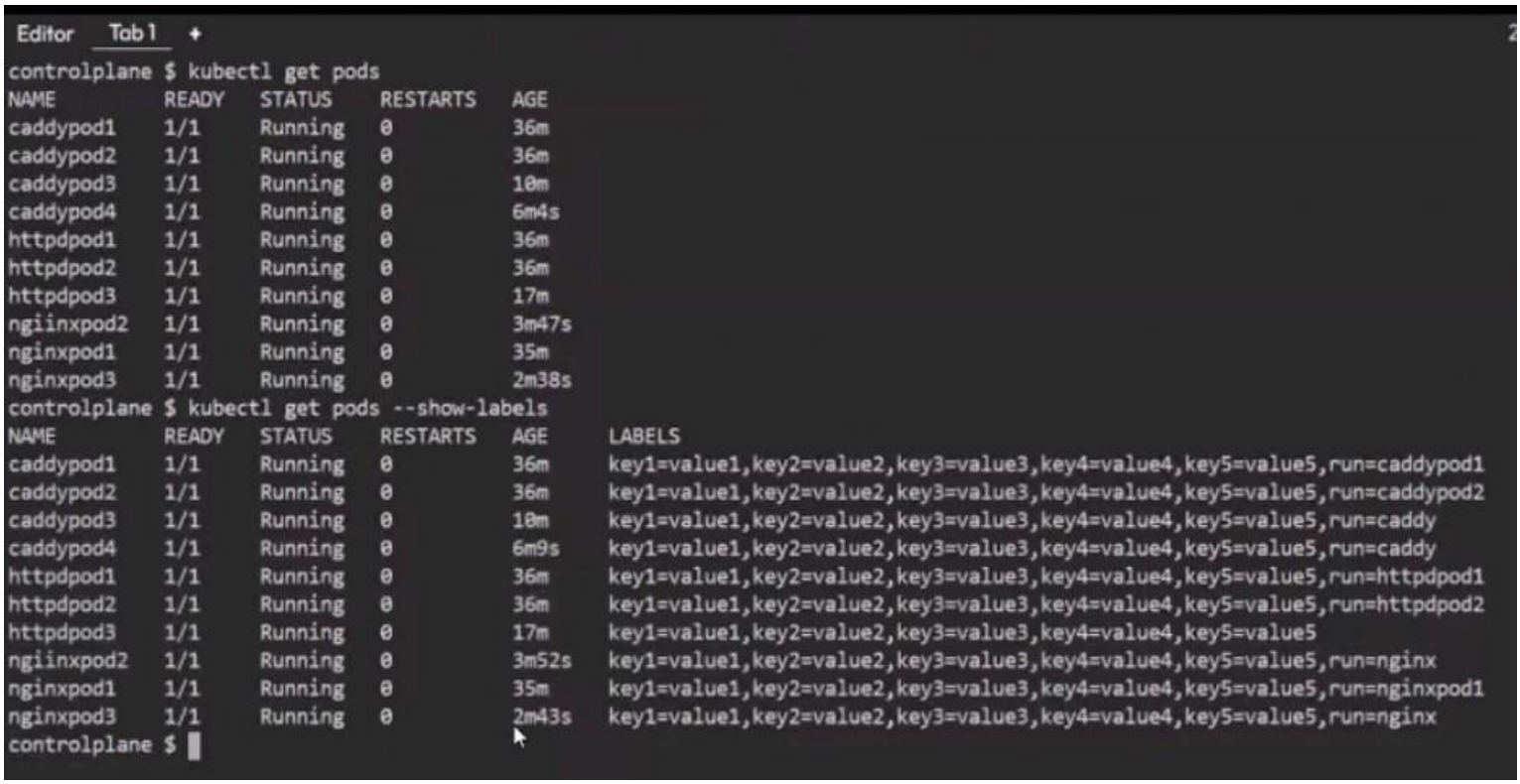
Step 2 :



Step 3 :


Editor
Tab 1
caddypod2 1/1 Running 0 17m
httpdpod1 1/1 Running 0 17m
httpdpod2 1/1 Running 0 17m
nginxpod1 1/1 Running 0 16m
controlplane $ kubectl create -f httpdpod3.yaml
pod/httpdpod3 created
controlplane $ kubectl get pods
NAME READY STATUS RESTARTS AGE
caddypod1 1/1 Running 0 18m
caddypod2 1/1 Running 0 18m
httpdpod1 1/1 Running 0 18m
httpdpod2 1/1 Running 0 18m
httpdpod3 0/1 ContainerCreating 0 7s
nginxpod1 1/1 Running 0 18m
controlplane $ kubectl get pods
NAME READY STATUS RESTARTS AGE
caddypod1 1/1 Running 0 18m
caddypod2 1/1 Running 0 18m
httpdpod1 1/1 Running 0 19m
httpdpod2 1/1 Running 0 18m
httpdpod3 0/1 ContainerCreating 0 24s
nginxpod1 1/1 Running 0 18m
controlplane $ kubectl get pods
NAME READY STATUS RESTARTS AGE
caddypod1 1/1 Running 0 19m
caddypod2 1/1 Running 0 18m
httpdpod1 1/1 Running 0 19m
httpdpod2 1/1 Running 0 19m
httpdpod3 1/1 Running 0 43s
nginxpod1 1/1 Running 0 18m
controlplane $

4. Assign labels to each pod using key-value pairs. You can assign five different labels with different values to each pod.



5. Create five replication controllers (RC), each with a different label selector matching the labels assigned to the pods. Ensure that each RC has four replicas. You can create the RCs using YAML definition files, or the “kubectl create” command.

Step 1:

```
controlplane $ kubectl get pod  httpdpod1 --show-labels
NAME        READY   STATUS    RESTARTS   AGE   LABELS
httpdpod1   1/1     Running   0          10m   key1=value1,key2=value2,key3=value3,key4=value4,key5=value5,run=httpdpod1
controlplane $ vi rc1.yaml
```

Step 2:

Editor Tab 1 +

```
apiVersion: v1
kind: ReplicationController
metadata:
  name: rc1
spec:
  replicas: 4
  selector:
    key1: value1
  template:
    metadata:
      labels:
        key1 : value1
    spec:
      containers:
      - name: httpdpod1
        image: httpd
~
~
~
```

Step 3:

```
controlplane $ kubectl create -f rc1.yaml
replicationcontroller/rc1 created
```

6. Finally, create five Deployments, each representing a specific application or scenario. You can use the "Kubectl create" command or create YAML definition files for the deployments.

Step 1:

```
controlplane $ vi httpddeploy1.yaml
```

Step 2:

Editor Tab 1 +

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: httpddeploy1
spec:
  replicas: 4
  selector:
    matchLabels:
      key1: value1
  template:
    metadata:
      labels:
        key1: value1
    spec:
      containers:
      - name: httpd1
        image: httpd
~
~
~
```

Step 3:

```
controlplane $ kubectl create -f httpddeploy1.yaml
deployment.apps/httpddeploy1 created
```

Once cross check the Deployments :

```
controlplane $ kubectl get deployments
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
httpddeploy1  4/4     4            4           33s
controlplane $ kubectl get deployments -owide
NAME          READY   UP-TO-DATE   AVAILABLE   AGE   CONTAINERS    IMAGES   SELECTOR
httpddeploy1  4/4     4            4           43s   httpddeploy1  httpd    key1=value1
```

- - - - - COMPLETED - - - - - - -