Project 5: Kubernetes

Submitted by-
K. Abhinay

Guided by -
Sir Ashish

What is Kubernetes?

- Kubernetes also known as K8s, is an open-source system for automating deployment, scaling, and management of containerized applications.
- It groups containers that make up an application into logical units for easy management and discovery. Kubernetes builds upon 15 years of experience of running production workloads at Google, combined with best-of-breed ideas and practices from the community.

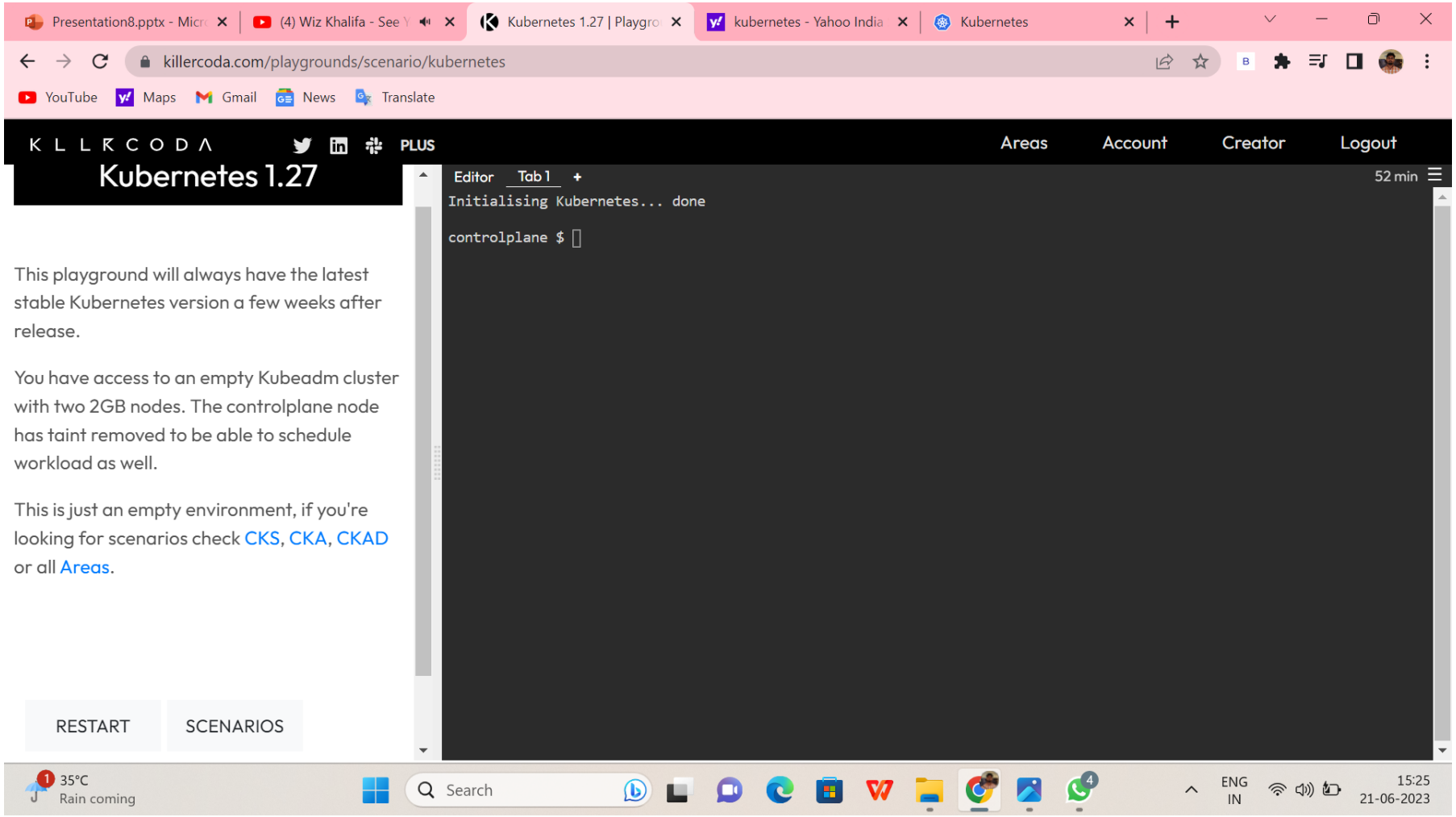
Kubernetes Features

- Automated rollouts and rollbacks
- Service discovery and load balancing
- Storage orchestration
- Self-healing
- Secret and configuration management
- Automatic bin packing
- Batch execution
- Horizontal scaling
- IPv4/IPv6 dual-stack
- Designed for extensibility

Project 5:

- Kubernetes :-
- Killercoda :-playground
- https://killercoda.com/playgrounds/scenario/kubernetes
- 1 master and one node
- for 60 minutes
- pods 10:-httpd, caddy, nginx
- 5 from 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. Create 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 `kubectl run` command to create the pods directly from the command-line interface (CLI):

Editor Tab 1 + 57 min

```
controlplane $ kubectl run httpdpod1 --image=httpd
pod/httpdpod1 created
controlplane $ kubectl run httpdpod2 --image=httpd
pod/httpdpod2 created
controlplane $ kubectl run caddypod1 --image=caddy
pod/caddypod1 created
controlplane $ kubectl run caddypod2 --image=caddy
pod/caddypod2 created
controlplane $ kubectl run nginxpod1 --image=nginx
pod/nginxpod1 created
controlplane $ kubectl get pods
NAME        READY   STATUS    RESTARTS   AGE
caddypod1   1/1     Running   0          37s
caddypod2   1/1     Running   0          31s
httpdpod1   1/1     Running   0          64s
httpdpod2   1/1     Running   0          52s
nginxpod1   1/1     Running   0          12s
controlplane $ kubectl get pods -o wide
NAME        READY   STATUS    RESTARTS   AGE   IP            NODE     NOMINATED NODE   READINESS GATES
caddypod1   1/1     Running   0          72s   192.168.1.5   node01   <none>           <none>
caddypod2   1/1     Running   0          66s   192.168.1.6   node01   <none>           <none>
httpdpod1   1/1     Running   0          99s   192.168.1.3   node01   <none>           <none>
httpdpod2   1/1     Running   0          87s   192.168.1.4   node01   <none>           <none>
nginxpod1   1/1     Running   0          47s   192.168.1.7   node01   <none>           <none>
controlplane $
```

3. For the remaining five pods, create a YAML definition file for each pod and use the `kubectl create` command to create the pods from the definition files.

Step1:

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

Step2:

```
Editor    Tab 1    +
apiVersion: v1
kind: Pod
metadata:
  name: caddypod3
  labels:
    key1 : value1
    key2 : value2
    key3 : value3
    key4 : value4
    key5 : value5
    fun : caddy
spec:
  containers:
  - name: caddy3
    image: caddy
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
-- INSERT (paste) --
```

Step3:

```
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

```
Editor   Tab 1   +                                                                                    22 min
controlplane $ kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
caddypod1     1/1     Running   0          36m
caddypod2     1/1     Running   0          36m
caddypod3     1/1     Running   0          10m
caddypod4     1/1     Running   0          6m4s
httpdpod1     1/1     Running   0          36m
httpdpod2     1/1     Running   0          36m
httpdpod3     1/1     Running   0          17m
ngiinxpod2    1/1     Running   0          3m47s
nginxpod1     1/1     Running   0          35m
nginxpod3     1/1     Running   0          2m38s
controlplane $ kubectl get pods --show-labels
NAME          READY   STATUS    RESTARTS   AGE     LABELS
caddypod1     1/1     Running   0          36m     key1=value1,key2=value2,key3=value3,key4=value4,key5=value5,run=caddypod1
caddypod2     1/1     Running   0          36m     key1=value1,key2=value2,key3=value3,key4=value4,key5=value5,run=caddypod2
caddypod3     1/1     Running   0          10m     key1=value1,key2=value2,key3=value3,key4=value4,key5=value5,run=caddy
caddypod4     1/1     Running   0          6m9s    key1=value1,key2=value2,key3=value3,key4=value4,key5=value5,run=caddy
httpdpod1     1/1     Running   0          36m     key1=value1,key2=value2,key3=value3,key4=value4,key5=value5,run=httpdpod1
httpdpod2     1/1     Running   0          36m     key1=value1,key2=value2,key3=value3,key4=value4,key5=value5,run=httpdpod2
httpdpod3     1/1     Running   0          17m     key1=value1,key2=value2,key3=value3,key4=value4,key5=value5
ngiinxpod2    1/1     Running   0          3m52s   key1=value1,key2=value2,key3=value3,key4=value4,key5=value5,run=nginx
nginxpod1     1/1     Running   0          35m     key1=value1,key2=value2,key3=value3,key4=value4,key5=value5,run=nginxpod1
nginxpod3     1/1     Running   0          2m43s   key1=value1,key2=value2,key3=value3,key4=value4,key5=value5,run=nginx
controlplane $
```

5. Create five ReplicationControllers (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.

Step1:

```
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
```

Step2:

```
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
~
~
~
```

Step3:

```
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

Step1:

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

Step2:

```
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
~
~
~
```

Step3:

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

Once cross check the Deployments :

```
deployment.apps/httpddeploy1 created
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
```