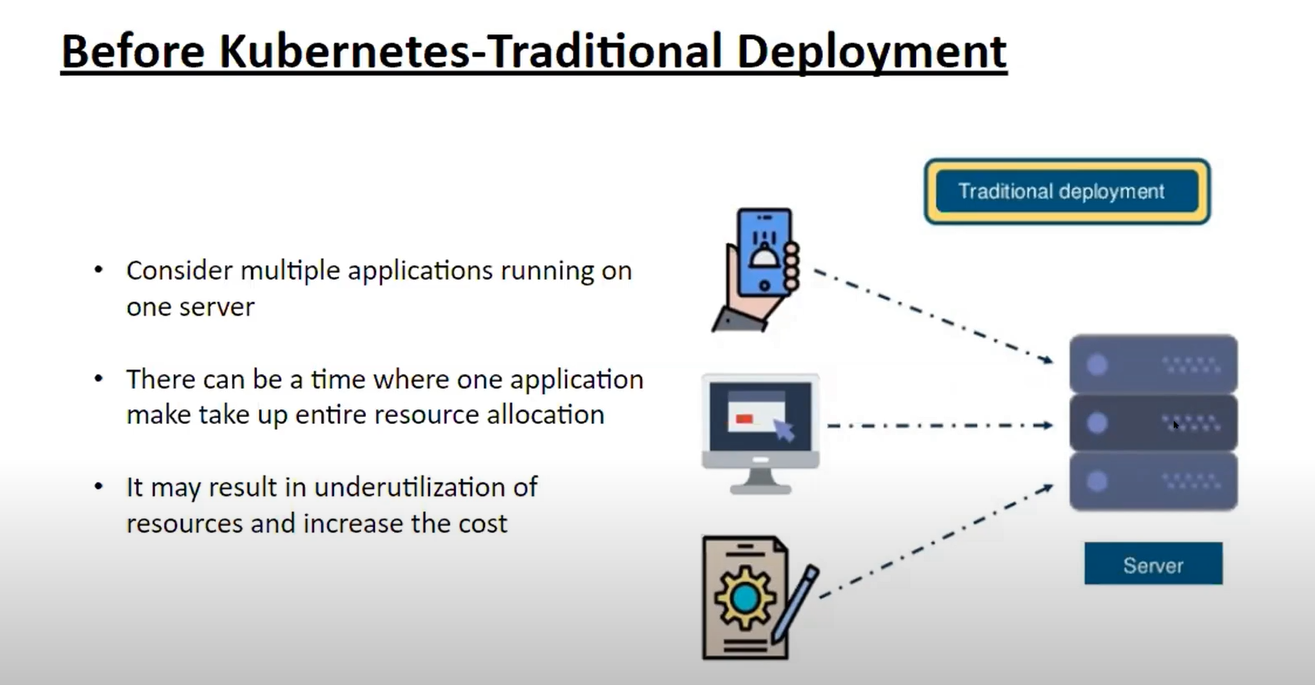
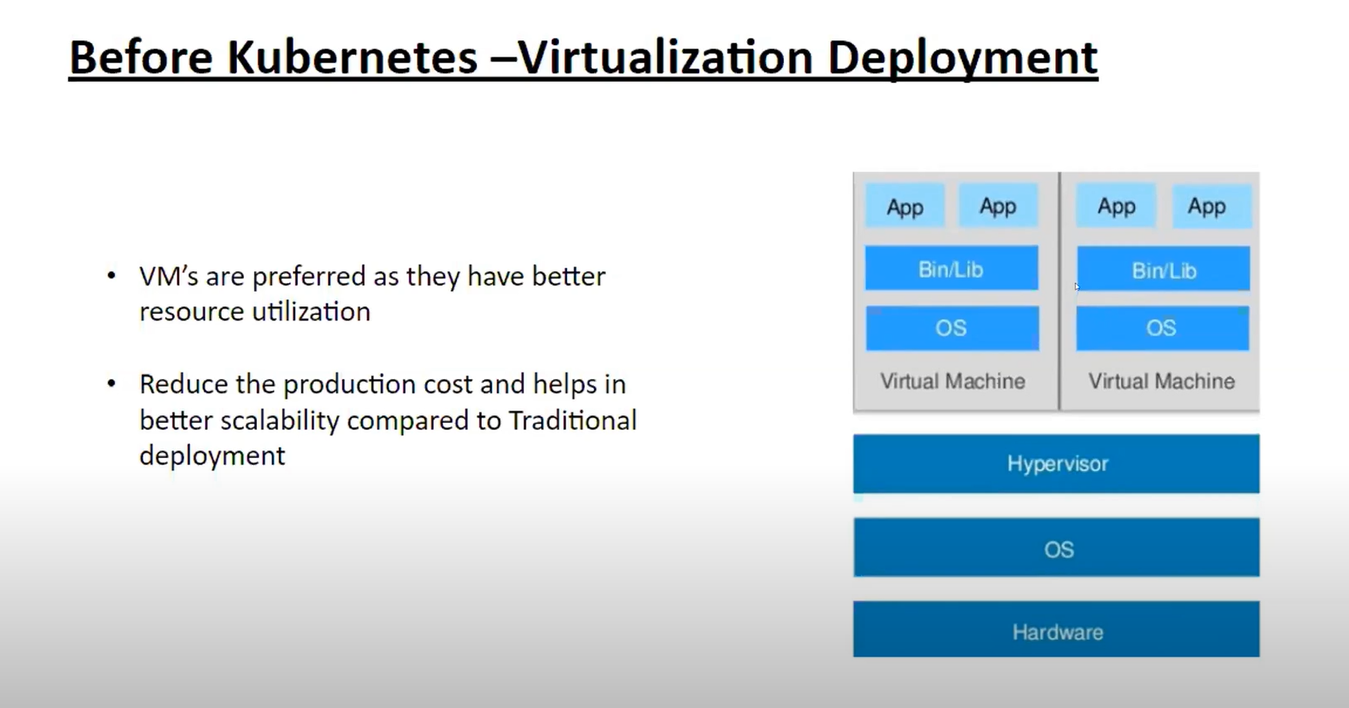
**Kubernetes**

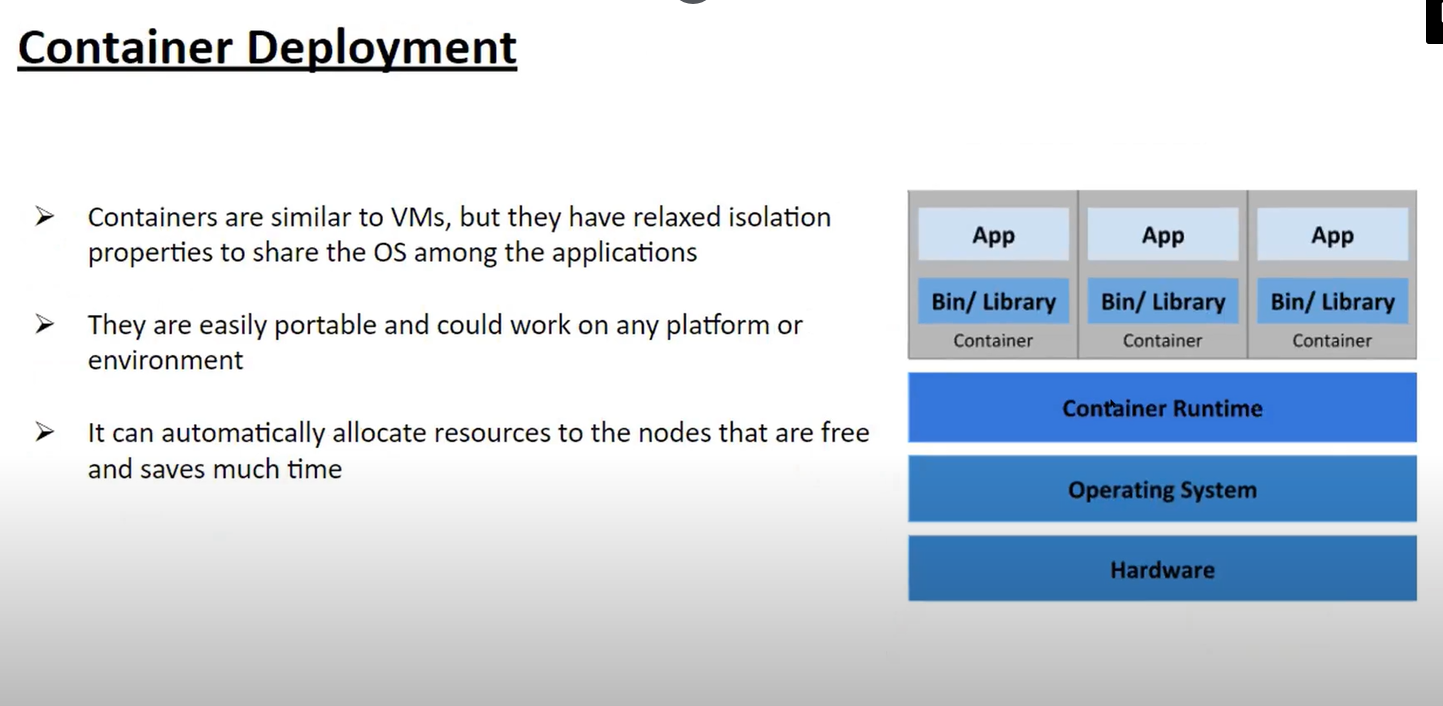
1. Kubernetes helps to scale up and scale down the server or application

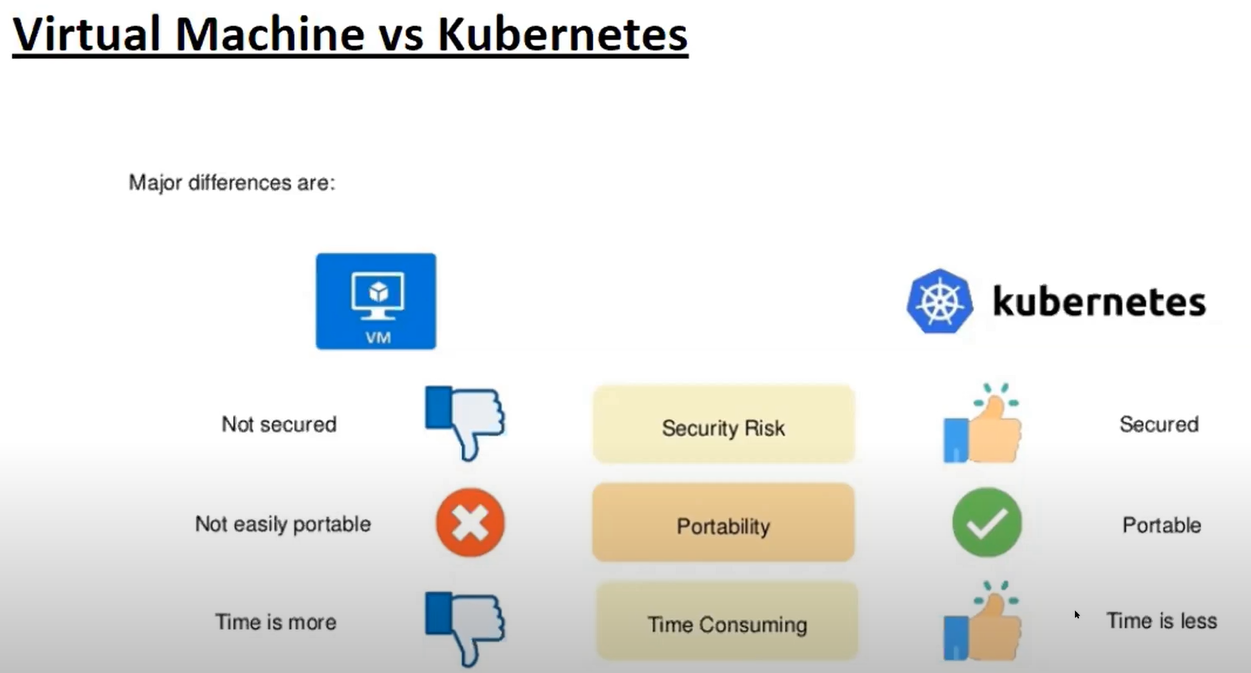


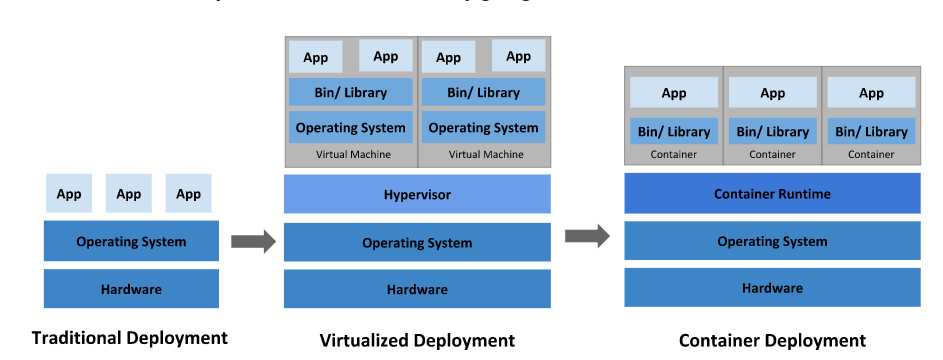
1. So to overcome this problem we are using Virtulization concept



1. In Virtulization, we have the problem of environment and Booting (Restarting)
2. To overcome this problem, we come to Kubernetes

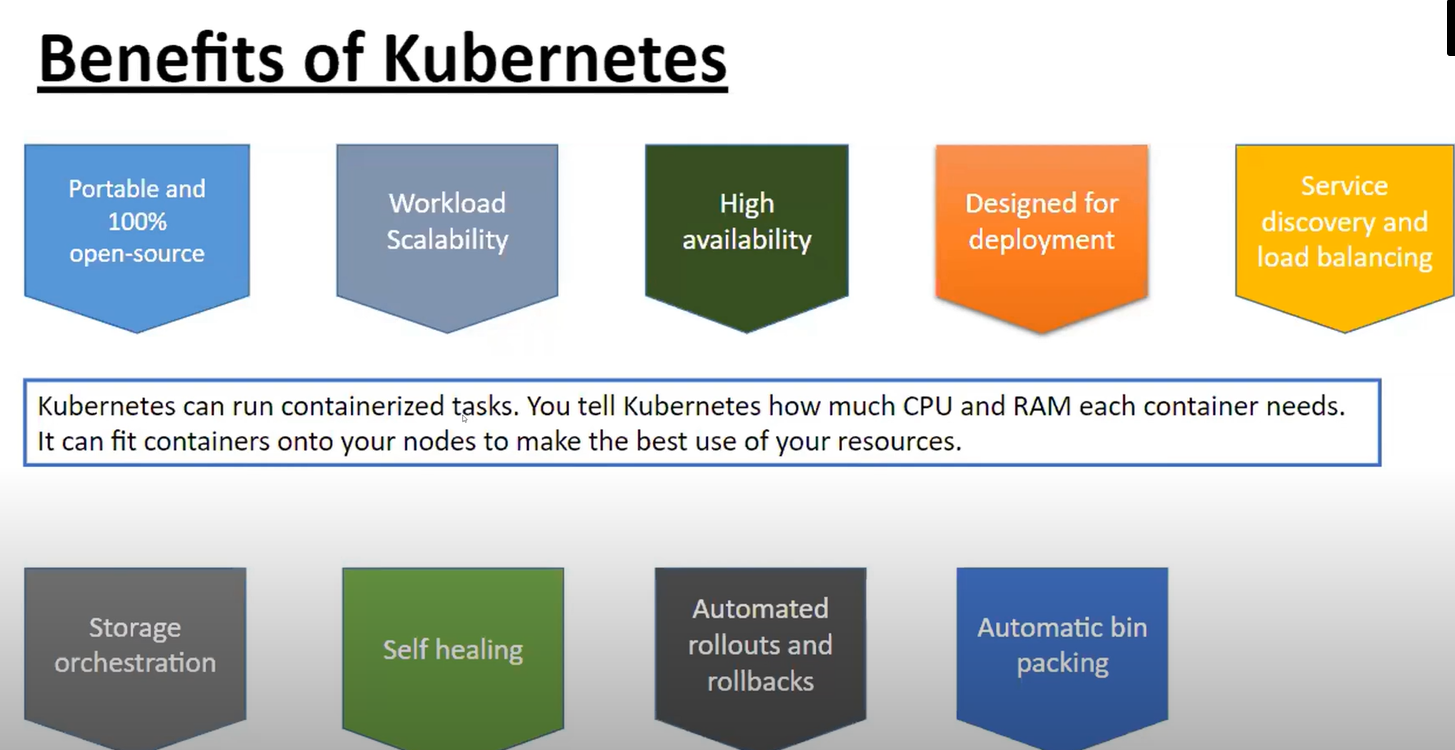


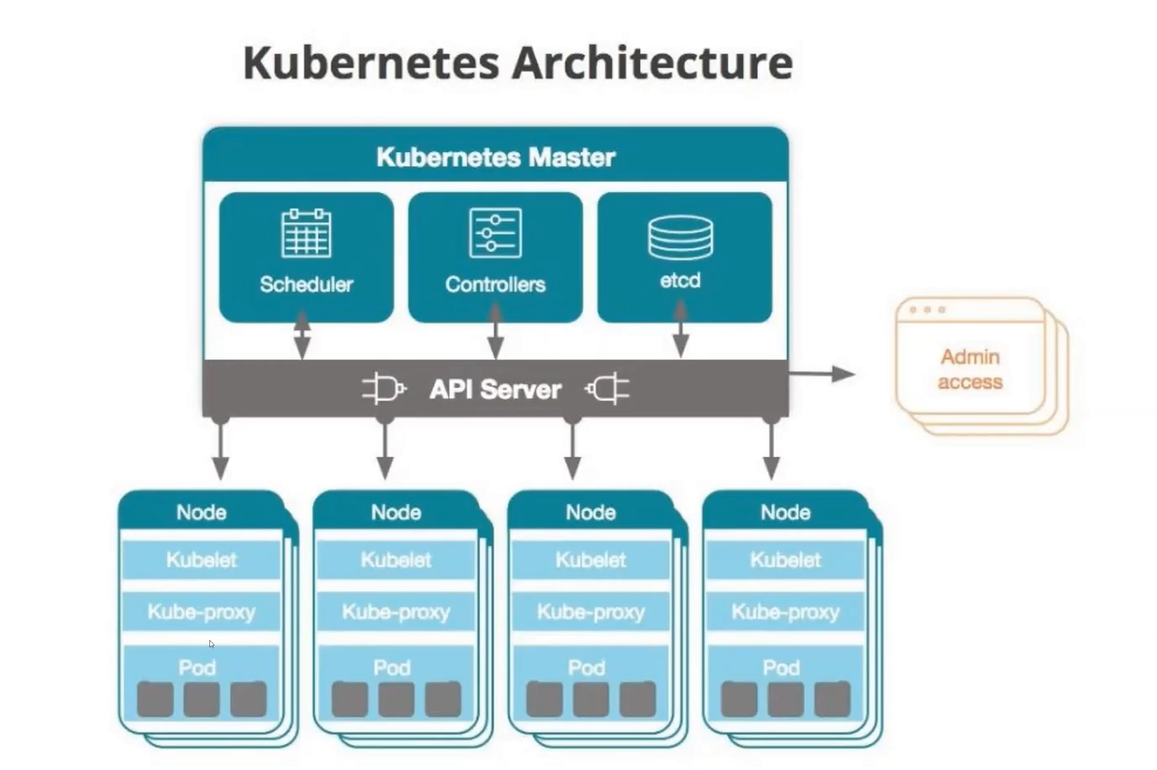


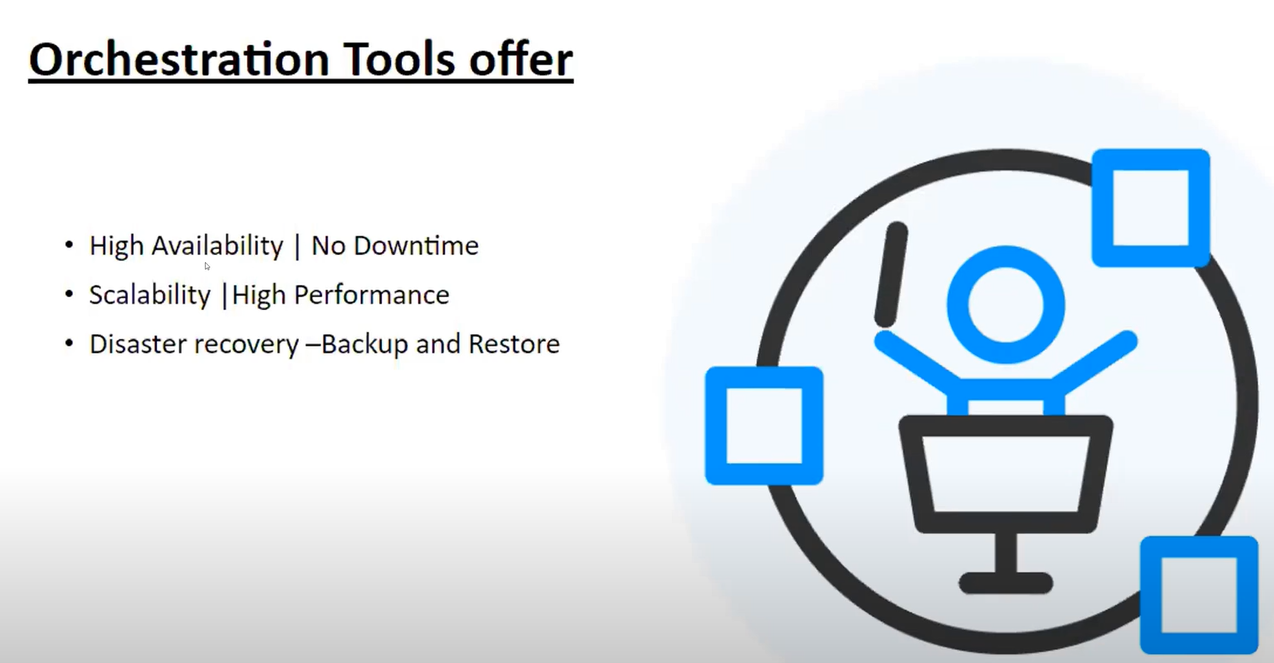


1. Kubernetes is developed by google
2. High availability- No Down Time
3. Scalability- High Performance
4. Backup and Restore

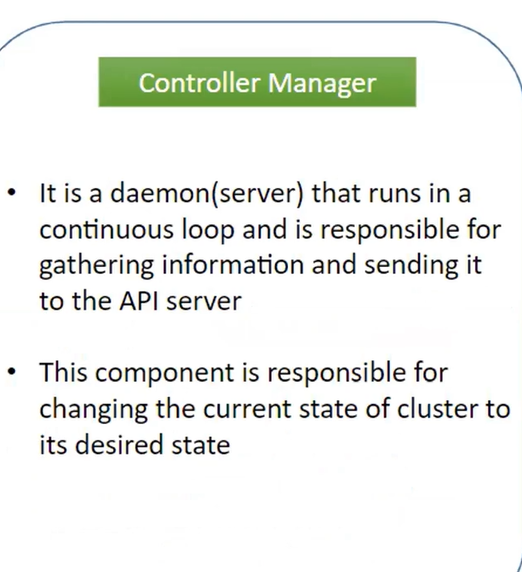
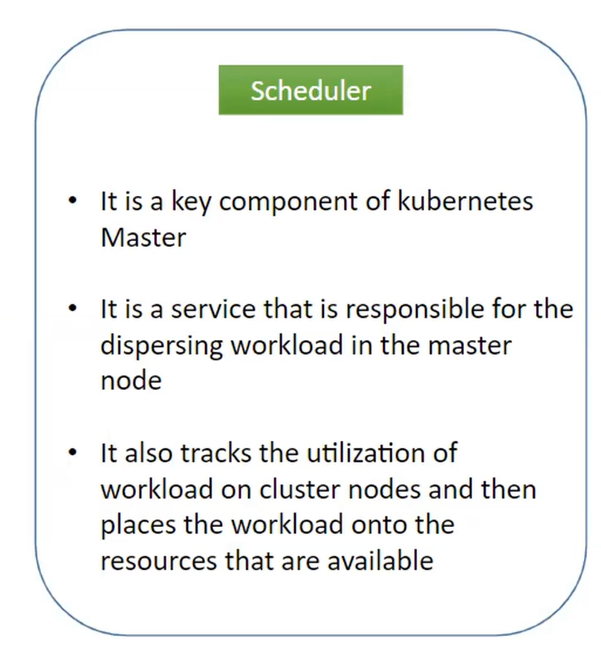
**Benefits of Kubernetes**

****

****

****

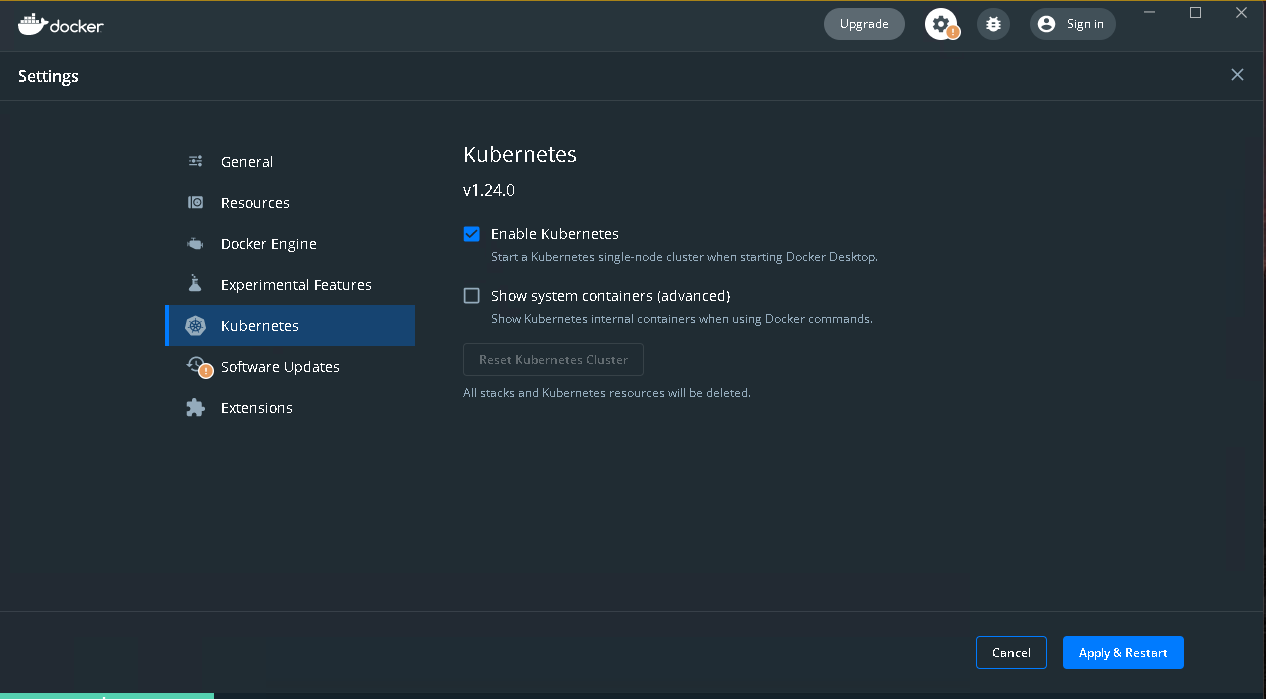
**Master and worker node**

****

**Installing Kubernetes in the Docker**

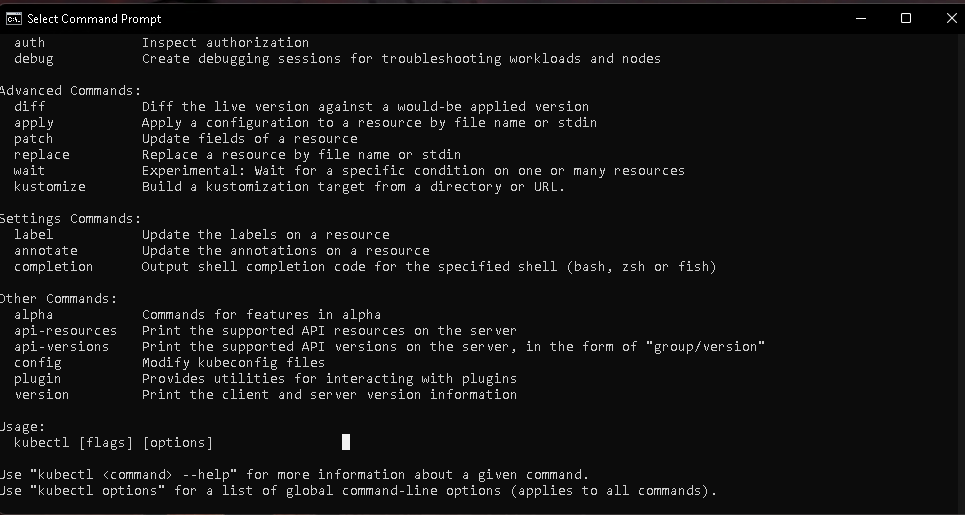
https://kubernetes.io/docs/tasks/access-application-cluster/service-access-application-cluster/

1. **Open Docker and click on Kubernetes and install the software**

****

1. **After Installing Kubernetes, open command prompt and type**

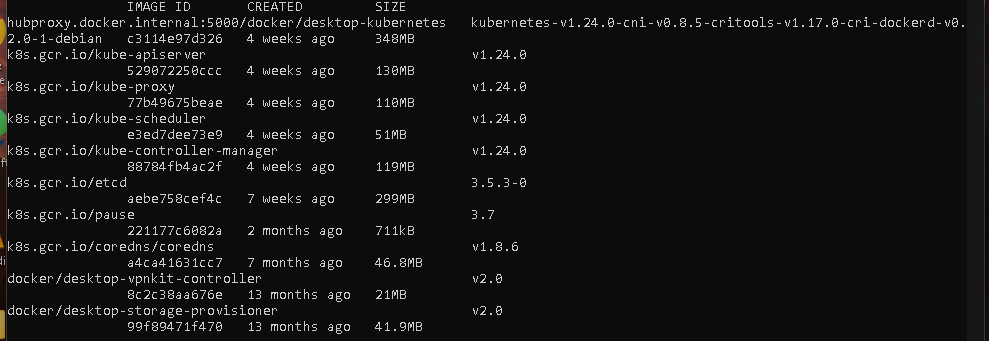
**Kubectl**

****

**Kubectl get pods**

****

1. **After we click docker images, we get many images which are related to Kubernetes**

****

Replica creates another pod when we experienece a server down

**apiVersion**: apps/v1

**kind**: Deployment

**metadata**:

**name**: hello-world

**spec**:

**selector**:

**matchLabels**:

**run**: load-balancer-example

**replicas**: 2

**template**:

**metadata**:

**labels**:

**run**: load-balancer-example

**spec**:

**containers**:

- **name**: hello-world

**image**: gcr.io/google-samples/node-hello:1.0

**ports**:

- **containerPort**: 8080

**protocol**: TCP

-f = file sorce

<https://k8s.io/examples/service/access/hello-application.yaml>

The above link we can access the file dynamically

1. Now we are deploying Hello world, copy the commad and run on command prompt

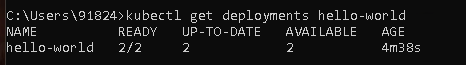
kubectl apply -f https://k8s.io/examples/service/access/hello-application.yaml



1. This is the stage of deploy

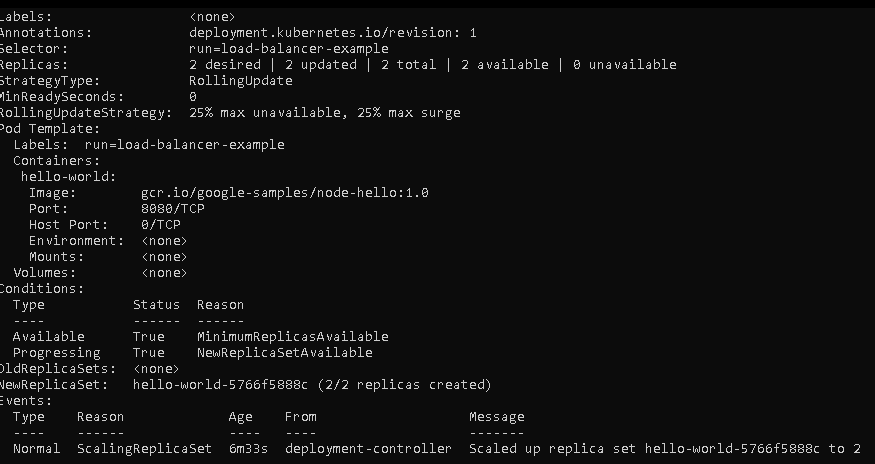
Now run the following command

kubectl get deployments hello-world



Here,2/2 is 2 replicas , and we deploy in 2 different replicas

1. Now run the following command



Name: hello-world

Namespace: default

CreationTimestamp: Sat, 04 Jun 2022 16:27:38 +0530

Labels: <none>

Annotations: deployment.kubernetes.io/revision: 1

Selector: run=load-balancer-example

Replicas: 2 desired | 2 updated | 2 total | 2 available | 0 unavailable

StrategyType: RollingUpdate

MinReadySeconds: 0

RollingUpdateStrategy: 25% max unavailable, 25% max surge

Pod Template:

Labels: run=load-balancer-example

Containers:

hello-world:

Image: gcr.io/google-samples/node-hello:1.0

Port: 8080/TCP

Host Port: 0/TCP

Environment: <none>

Mounts: <none>

Volumes: <none>

Conditions:

Type Status Reason

---- ------ ------

Available True MinimumReplicasAvailable

Progressing True NewReplicaSetAvailable

OldReplicaSets: <none>

NewReplicaSet: hello-world-5766f5888c (2/2 replicas created)

Events:

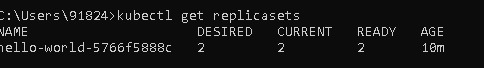
Type Reason Age From Message

---- ------ ---- ---- -------

Normal ScalingReplicaSet 6m33s deployment-controller Scaled up replica set hello-world-5766f5888c to 2

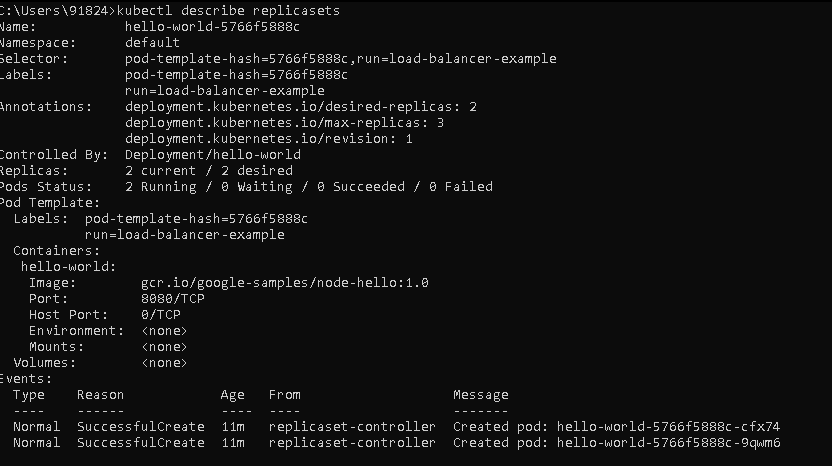
1. We get the replica sets using the following command

kubectl get replicasets



1. We get the status of the replica, any one using it

kubectl describe replicasets



1. Create a Service object that exposes the deployment:

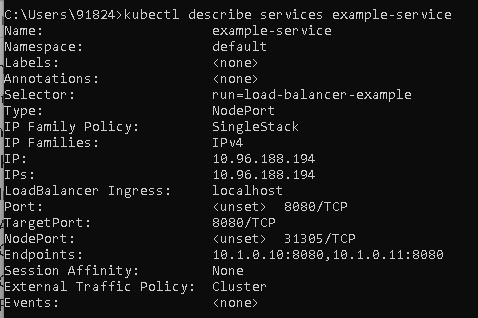
kubectl expose deployment hello-world --type=NodePort --name=example-service



.

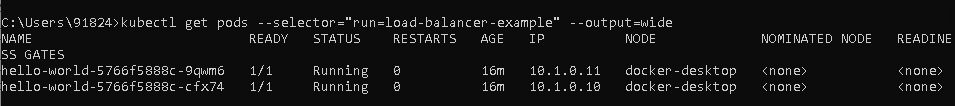
1. Now we need to use the following command

kubectl expose deployment hello-world --type=NodePort --name=example-service



1. List the pods that are running the Hello World application:

kubectl get pods --selector="run=load-balancer-example" --output=wide



1. Now we are running the application

This is the port we need to run the application

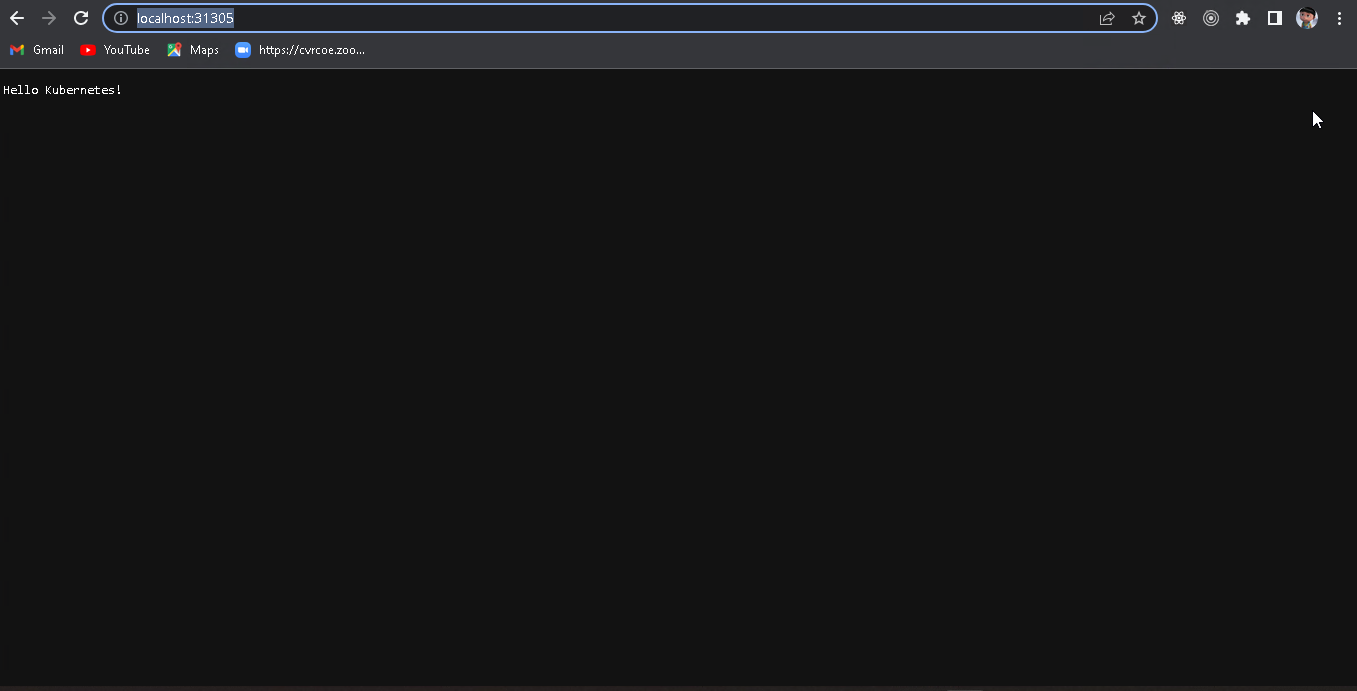
NodePort: <unset> 31305/TCP

We can find the node port using the command:

kubectl describe services example-service

Open browser and give the following ip addrress:

http://localhost:31305/



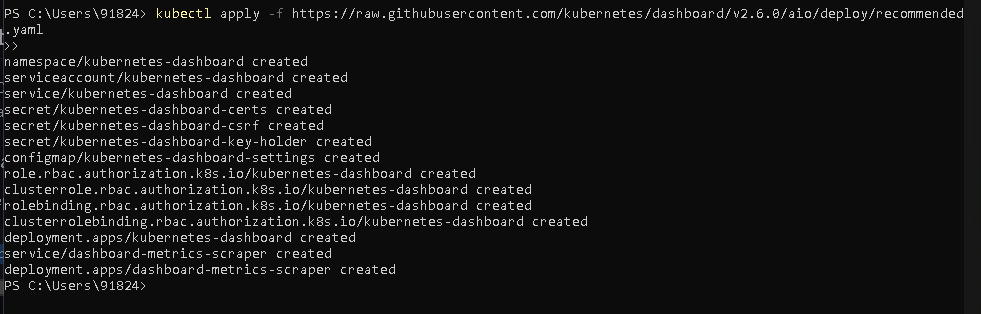
Now we are creating a kubernetes DashBoard

* To have the graphical interface of pods and nodes

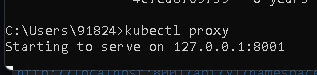
<https://github.com/kubernetes/dashboard>

1. Open Powershell and follow the following commands

kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/v2.6.0/aio/deploy/recommended.yaml



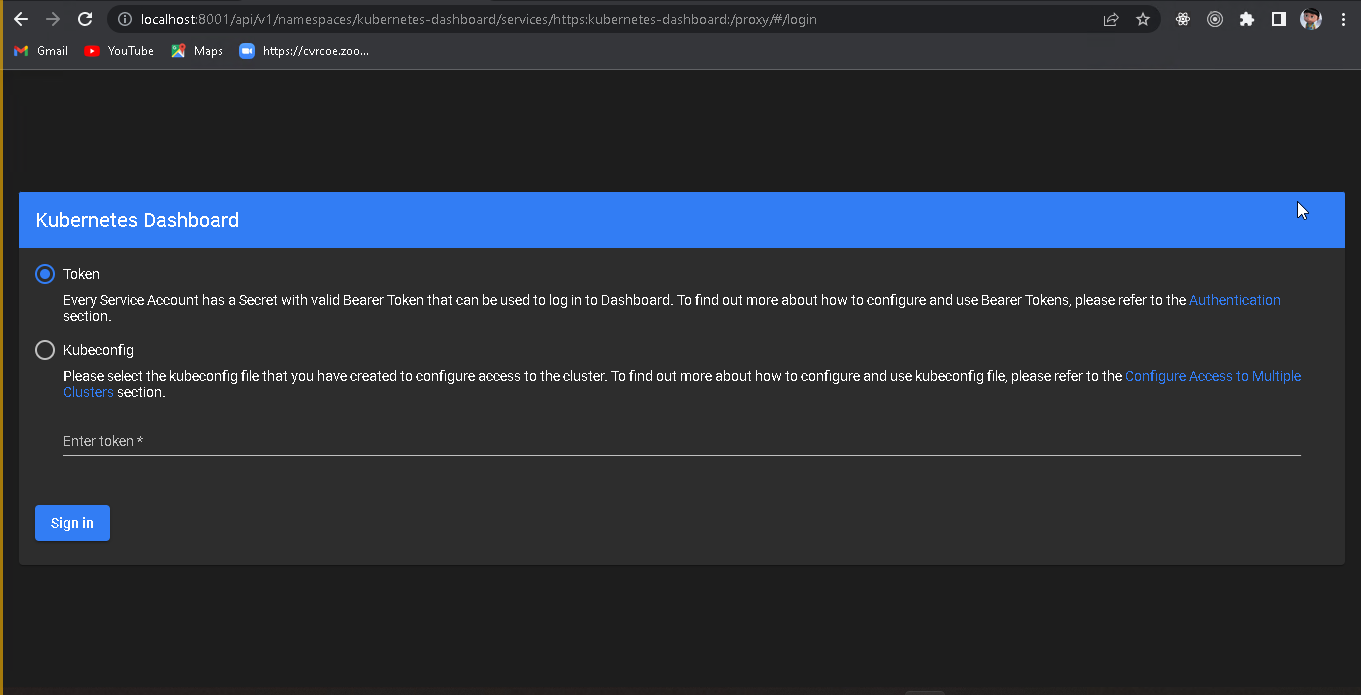
1. kubectl proxy



1. Now after that click on the following link in GITHUB

Open this on browser

<http://localhost:8001/api/v1/namespaces/kubernetes-dashboard/services/https:kubernetes-dashboard:/proxy/>



1. Now inorder to get a new token we need to follow the following steps:

<https://github.com/kubernetes/dashboard/blob/master/docs/user/access-control/creating-sample-user.md>

1. Now we need to create a file with the following names and copy the following code into it

a.service-account.yaml

apiVersion: v1

kind: ServiceAccount

metadata:

name: admin-user

namespace: kubernetes-dashboard

b.clusterbindingrole.yaml

apiVersion: rbac.authorization.k8s.io/v1

kind: ClusterRoleBinding

metadata:

name: admin-user

roleRef:

apiGroup: rbac.authorization.k8s.io

kind: ClusterRole

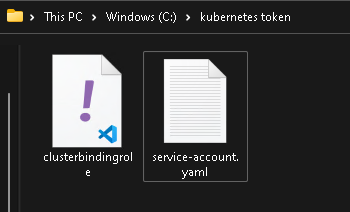
name: cluster-admin

subjects:

- kind: ServiceAccount

name: admin-user

namespace: kubernetes-dashboard



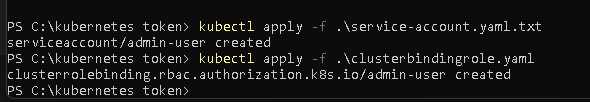
1. Now go to power shell and execute the following codes

Now go to the path of file which we had created the files in power shell

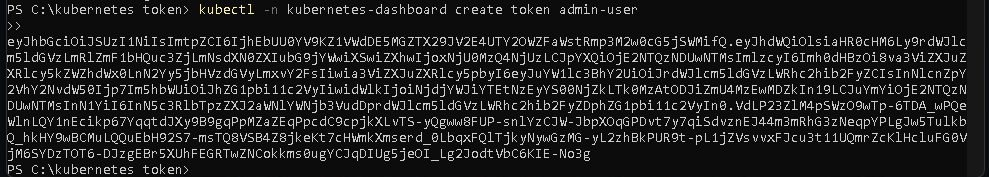
kubectl apply -f .\service-account.yaml

kubectl apply -f .\clusterbindingrole.yaml

we are creating the admin user



kubectl -n kubernetes-dashboard create token admin-user



Now copy the token and paste it in the given field

