Some of the Commands Used in The MiniKube

**docker build -t test-image . :-** This will create the image on the docker if my application contain the docker file

**docker run -p 5000:5000 test-image: -** it will run the image

**docker tag test-image saif7755/test-image:latest :- add the tag**

**docker push saif7755/test-image:latest :- push the image to docker hub**

**Start: -** starts a local Kubernetes cluster

**Status: -** Gets a status of a local Kubernetes cluster

**Stop:** - Stops a running Kubernetes cluster

**Delete: -** Deletes a local Kubernetes cluster

**Dashboard: -** Access the Kubernetes dashboard running on minikube cluster

**Pause: -** Pause Kubernetes

**Unpause:-** Unpause Kubernetes

**Ip: -** minikube ip gives the ip of the docker ( 192.168.59.101 )

**Minikube update-status: -** to update the context of the minikube cluster if there are two

If there is no docker in the window there is always a docker inside the minikube node.  
Each node in the Kubernetes is the server whether it is virtual and physical we just need to connect with it with ip.

**SSH : -**it is a standard protocol for management of any service ( including remote servers )

But Minikube also provides command to SSH into the local minikube node

**Minikube SSH**

If you set –driver = docker you should use minikube ssh because ssh@<minikube ip> will not work.

ssh [docker@192.168.59.101](mailto:docker@192.168.59.101) after that I will ask for password and

you can check where is the key then you used this path

> minikube ssh-key

C:\Users\SaifUllahKhan\.minikube\machines\minikube\id\_rsa

PS C:\Users\SaifUllahKhan> ssh -i ~/.minikube/machines/minikube/id\_rsa [docker@192.168.59.101](mailto:docker@192.168.59.101)

After running into the container

**Docker ps :-** Tell us about the container in the cluster

\*\*\*\*\* we can not used the kubectl command as we are in the container \*\*\*\*\*\*\*\*\*

But

\*\*\*\*\* Minikube also povide kubectl tool and we can run commands for example like this

^^^^^^^^^^^^^ **minikube kubectl – cluster-info** ^^^^^^^^^^^^^^^^^

**Kubectl cluster-info: -** give the information about the cluster

**Kubectl get nodes : -** give us the information about the nodes

**kubectl get pods :-**  to get the information of the pods in the default namespace

**kubectl get pods --namespace=kube-system :-**  to get the pods of specific namespace

**kubectl get namespaces :-**  to get the information about the namespaces and these are used to group the resources and configuration objects.

Docker run command create single container

Nginx is the docker container image available in the docker hub

**kubectl run nginx --image=nginx: -** Create the pods and in the pod create the nginx container

**kubectl describe pod nginx:-**  give the information about the

Using ssh we are inside the node or cluster so we used docker commands

In order to connect to the container that is running the nginx service .Go inside the cluster run command **docker ps | grep nginx**

It will return the information about the nginx pod or service

**9fc72b6dbd74 using the id we can connect with the container**

**docker exec -it 9fc72b6dbd74 sh** = after the -it is the key of the container and it is used to connect with the container

**hostname** = give the name off the pod which is same for the container  
**hostname i** = it will return the for example ( 10.244.0.6)

**curl 10.244.0.6** = we connect with the server using the curl which is running inside the container

**10.244.0.6 =**  return the pods with the ip addresses

To connect to the container from the computer to the container

Curl 10.244.0.6 will not help us to connect to the container

**kubectl delete pod nginx : -** delete the pod and remove the namespaces , volumes and resources.

**What are the Deployments?**

Ans:- A Deployment in Kubernetes is a powerful tool for managing and scaling containerized applications. It automates tasks like updating applications, scaling them up or down, and maintaining a desired state, ensuring that your applications run reliably and consistently in a Kubernetes cluster.

Instead of creating separate pods we used deployment.

We create services to connect with the deployments from the external world.

**kubectl create deployment <name-of-deployment> --image=nginx :-**  Create the deployments

**kubectl describe deployment nginx-deployment:-**  Get the details of the deployments

Selectors :- connect the pods with the deployments

Replicas : - Tell about the quantity of the pods

ReplicaSet :- Manages the all the pods of the deployments and it is set of replicas of our applications like we can create 1, 10, 100 different pods that can be included in the replicaset.

K get pods return the pod name starting with the replica set name.

Replica set control the pods

**kubectl scale deployments nginx=deployment --replicas= <number of replicas>:-**  it will help us to scale the deployments like to increase and decrease the number of the pods inside the deployment.

**We did not create these pods manually but k8s scale this deployment for us**

All have same prefix it is the name of the replica set to which all pods are belong to

Each pods have different ip addresses although all the pods were created on the single node.

**Connection with the Nodes with local computer**

**Curl 10.244.0.17** using one of the ip of the pod we try to connect but failed

Now try to connect them by going inside the node and check whether it works or not.

**Steps to SSH into Minikube:**

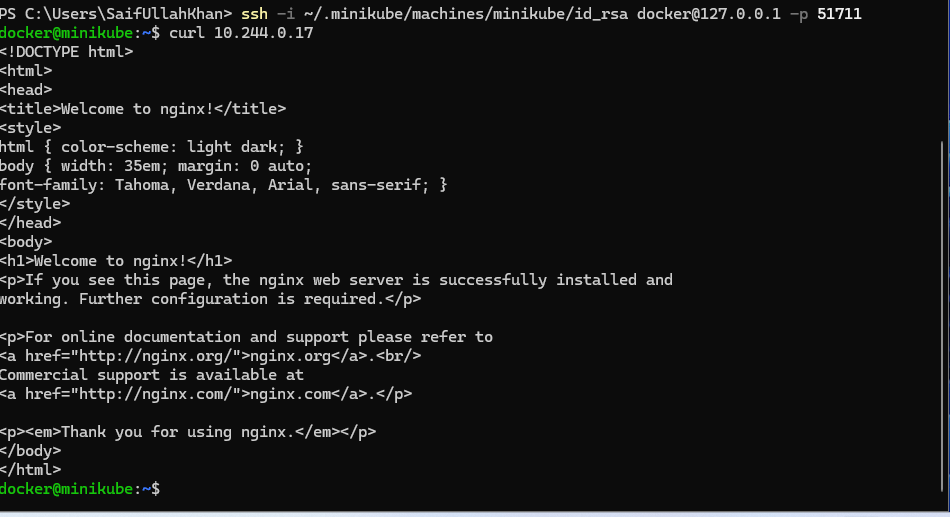
1. **Identify the SSH Port:**
   * **First, identify which port Docker is mapping to the SSH port inside the Minikube container. Usually, Minikube’s Docker container should be configured to expose port 22 (SSH). If Docker has exposed it to a different port on the host machine, you’ll need to identify that port.**

**Run:**

**docker port minikube 22**

**This will show the port mapping, something like 0.0.0.0:2222->22/tcp, which means port 2222 on your host is forwarded to port 22 on the Minikube container.**

Now ion the image we can see that how we can connect with the pods when we are inside the node.



Ip addresses are assigned to the pods dynamically so we cannot much relay on these ips.

So we should utilize some other methods which are managed by Kubernetes and which are allowed any of the pods inside of the deployments.

In Kubernetes we have to create the so called the services if you would like to connect to specific deployment using specific IP addresses. We can create so called cluster IP and such IP addresses will be created and assigned to specific deployment and we will be able to connect to this specific deployment all the inside of the cluster using this virtual IP address. And Kubernetes will distribute a lot across different pods which are assigned to specific deployments.

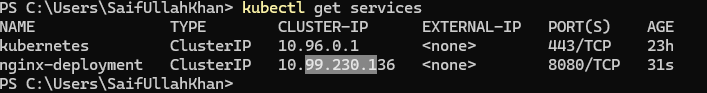
We can use the single ip address of a load balancer which is assigned by the cloud platforms.

By default neginx server run on the port 80 .

**kubectl expose deployemnt nginx-deployment --port=8080 --target=80**

We exposed the internal port (80) of the deployment to the outer port(8080).

Afte that we start a service



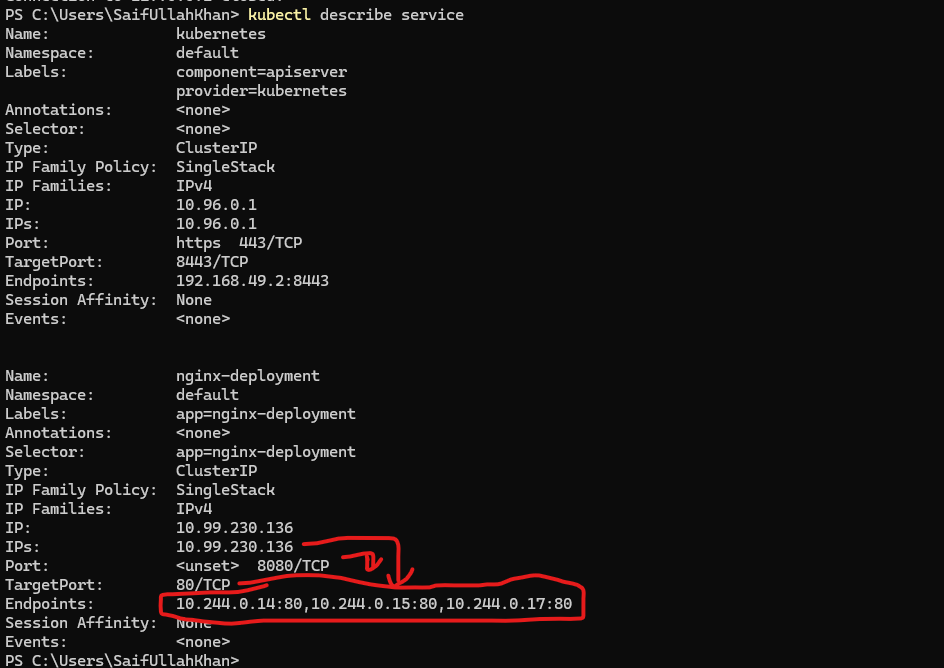
A virtual ip address is created to connect with any of the pod and it is created by the k8s.

Cluster api will be available inside the cluster.

**curl 10.99.230.136:8080** we can not access this ip from the outside but we can get access the from the nodes (any node inside the cluster).

When inside the cluster **curl 10.99.230.136:8080** this will return the answer from the service.

Hence cluster api is the service for the deployments.



**kubectl delete deployment <name of the deployments>**

**Kubectl delete service <name of the service>**