

## Slide 1

After the installation, scroll down in the same previous document, we will now get access to cluster as shown, use these two commands as displayed to interact with your cluster ■ follow the steps from next slide

## Slide 2

Copy the first command as shown

## Slide 3

Open the command prompt/ Windows PowerShell always in administrative mode and paste the command as shown

## Slide 4

minikube installed successfully

## Slide 5

## Slide 6

Exercise - 1

## Slide 7

Open Windows PowerShell in Administrator Mode and login into Docker

## Slide 8

We can start minikube using Hyper-V, Docker or VM minikube start --vm-driver=virtualbox if I use only minikube start by default Linux: docker (if installed) or kvm2, Windows: hyperv, macOS: hyperkit will be used.

## Slide 9

The minikube status command is used to check and display the current status of the Minikube cluster running on your local machine. In simple terms, it provides information about whether the Minikube cluster is up and running or if it's stopped.

## Slide 10

Lets deploy an application in Kubernetes `kubectl create deployment mynginx --image=nginx` if already created then `kubectl set image deployment/mynginx nginx=nginx:latest` `kubectl` is a command-line tool used in Kubernetes to interact with and manage Kubernetes clusters. This command is like telling Kubernetes to create a group named "mynginx" that will run an application (Nginx) inside it. This is a common way to start running and managing applications on a Kubernetes cluster.

## Slide 11

Nginx (pronounced "engine-x") is a web server software that helps deliver and manage websites on the internet. It can handle tasks like serving web pages, processing HTTP requests, and load balancing, making it a crucial component in hosting and delivering web content efficiently. You've just told Kubernetes to create a team (deployment) called "mynginx" that will manage an application (Nginx) for you. Seeing "created" means Kubernetes has successfully done that for you. Now, your Nginx application is ready to run and be managed by Kubernetes.

## Slide 12

`kubectl get deployment` when you execute this command, Kubernetes responds by showing you a list that includes the names of your deployment groups, how many instances of your applications are running, and other useful details. `kubectl get pods` running `kubectl get pods` is a quick way to check which of your applications are currently active and doing their jobs inside the Kubernetes system. \*Here the status shows that the containercreating\*

## Slide 13

This command opens up each pod's file and tells you everything about it. It's like reading a detailed report that includes the current state, recent events, and all the configurations of each of your applications.

So, running `kubectl describe pods` is a way to get a thorough understanding of what's happening inside each pod in your Kubernetes cluster. `>kubectl describe pods`

## Slide 14

Continuation of previous

## Slide 15

Type the following commands once again `kubectl get deployment` `kubectl get pods` Now the status of a pod is "Running," it means that the container within that pod is currently up and actively running.

## Slide 16

`kubectl scale deployment mynginx --replicas=4` this command is used to increase the number of replicas (copies) of the "mynginx" deployment in your Kubernetes cluster to 4. `scale deployment mynginx`: Specifies that you want to scale the deployment named "mynginx." `--replicas=4`: Sets the desired number of replicas to 4. This means that Kubernetes will ensure there are four instances of your "mynginx" application running, distributing the workload and potentially improving performance or resilience. This command helps you adjust the number of running instances of your application to meet the demands of your workload or improve fault tolerance.

## Slide 17

Now if we check the following command,  
`kubectl get deployment`  
We can see that 4 instances of mynginx are available  
>`kubectl get depolyment`

## Slide 18

Now if we check the following command,  
`kubectl get pods`  
We can see that 4 instances of mynginx are running  
`kubectl get pods`

## Slide 19

`kubectl describe pod mynginx-6b78685d4d-hrc9h`  
This command provides detailed information about a specific pod  
`describe pod`: Instructs Kubernetes to give detailed information about a specific pod.  
`mynginx-6b78685d4d-hrc9h`: The unique identifier for the specific pod you want information about.  
When you run this command, you'll receive information such as the pod's current status, events, labels, containers, and other details. It's a helpful command for troubleshooting or gaining insights into the configuration and state of a particular pod in your cluster. (shown in next slide)

## Slide 20

## Slide 21

Use the existing service  
`kubectrl get service myngnix`

Here 30088 is the NodePort on your host.

Port-forward to localhost (most reliable on Windows)

## Slide 22

Expose deployment on the correct container port (80)  
`kubectrl expose deployment myngnix --type=NodePort --port=80 --target-port=80 --name=myngnix`

This command is used to make a service publicly accessible from outside the Kubernetes cluster.  
expose deployment myngnix: Creates a service to expose the "mynginx" deployment.  
--type=NodePort: Specifies the type of service. In this case, it's a NodePort service, which means the service will be accessible on a specific port on each node in the cluster.  
--port=88: Specifies that the service should listen on port 88.  
So, after running this command, you'll have a service that forwards external traffic to the "mynginx" deployment on port 88 through a NodePort. This allows you to access your application externally by connecting to any node's IP address and the specified NodePort.

## Slide 23

Check the NodePort assigned  
`kubectrl get service myngnix`

## Slide 24

Port-forward to localhost (most reliable on Windows):  
>`kubectrl port-forward service/myngnix 30088:80`

## Slide 25

Copy and Paste the URL generated in a browser

## Slide 26

The nginx server is now accessible

## Slide 27

minikube dashboard  
command is used to open the Kubernetes Dashboard, a web-based user interface, when you are

working with Minikube.

When you run this command, Minikube will launch a web browser window or provide a URL that you can visit to access the Kubernetes Dashboard. The Dashboard is a graphical interface that allows you to monitor and manage different aspects of your local Kubernetes cluster, such as viewing running pods, services, deployments, and more. It's a useful tool for visualizing and interacting with your Kubernetes resources during development and testing.

## Slide 28

Minikube dashboard

## Slide 29

you can see deployment of mynginx

## Slide 30

4 replicas of mynginx

## Slide 31

`kubectl delete deployment mynginx1`

this command is telling Kubernetes to delete the deployment named "mynginx1" along with its associated resources (like pods).

The command `minikube stop` is used to stop a running Minikube cluster. Minikube is a tool that allows you to run Kubernetes clusters locally for development and testing purposes.

When you run `minikube stop`, it halts the operation of the Minikube cluster, suspending the virtual machines and freeing up system resources on your local machine. This is useful when you're done working with the cluster and want to conserve resources or when you need to temporarily pause the cluster. You can later use `minikube start` to resume the cluster from where you left off.

## Slide 32

THE END