Kubernetes (K8s)

Container Orchestration Tool

Introduction

- Kubernetes is an **open-source container orchestration** system for automating software deployment, scaling, and management.
- Kubernetes orchestration **allows you to build application services that span multiple containers, schedule containers across a cluster, scale those containers, and manage their health over time**.
- **Google** originally designed Kubernetes

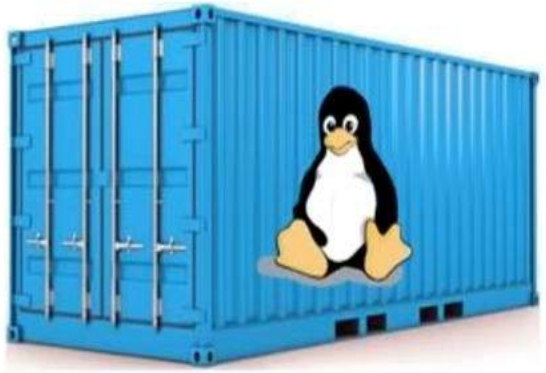
Introduction

- Kubernetes allows us to host the applications in the form of containers in an automated way.
- Dockers will only run containers, if in any case the container fails/stopped/killed, the docker will not help us, here is where Kubernetes plays an important role, Kubernetes cluster will be responsible in creating a new container and managing various containers.

Containers Are Good…

Both ***Linux Containers*** & ***Docker Containers***
isolate the application from the host.

FASTER, RELIABLE, EFFICIENT, LIGHT-WEIGHT & SCALABLE.

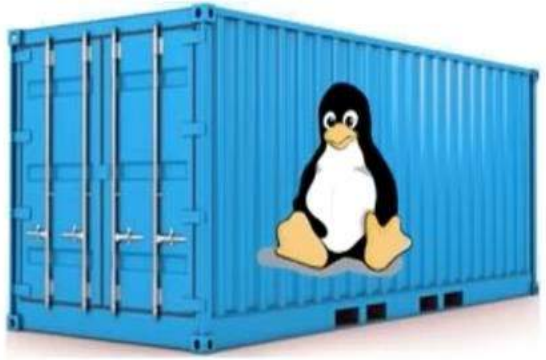


Damn! Container Problems...

Both ***Linux Containers*** & ***Docker Containers*** isolate the application from the host.

But.....Not easily Scalable...

FASTER, RELIABLE, EFFICIENT, LIGHT-WEIGHT & SCALABLE.



Problems With Scaling Up The Containers

1. Containers could not **communicate** with each other
2. Containers had to be **deployed appropriately**
3. Containers had to be **managed carefully**
4. **Auto scaling** was not possible
5. **Distributing traffic** was still challenging

So, What Is Needed?

A Container Management Tool !!!

Kubernetes is an open-source **Container Management** tool which automates *container deployment, container (de)scaling & container load balancing*.

Benefit: *Works brilliantly with all cloud vendors:* ***Public, Hybrid & On-Premises***.

A Container Management Tool !!!

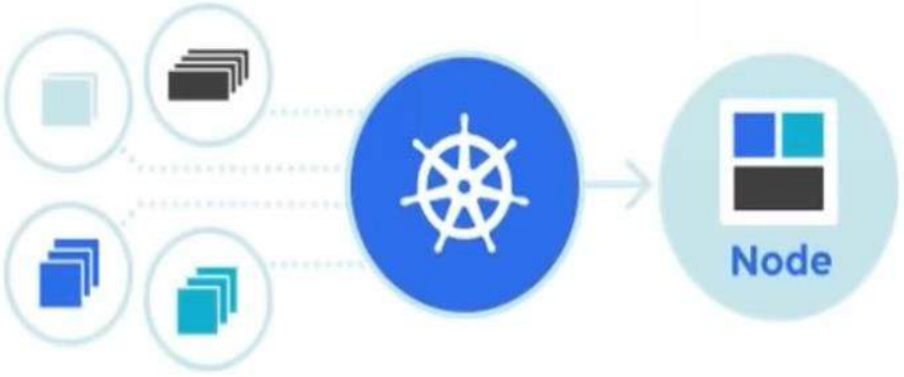
Kubernetes is an open-source **Container Management** tool which automates *container deployment, container (de)scaling & container load balancing*.

***Benefit**: Works brilliantly with all cloud vendors: **Public, Hybrid & On-Premises**.*

More About Kubernetes

- Written on Golang, it has a huge community because it was first developed by Google & later donated to **CNCF**
- Can group 'n' no of containers into one logical unit for managing & deploying them easily

Cloud native computing foundation



Reference: https://kubernetes.io/

Kubernetes

- Kubernetes is an open-source container orchestration platform. It automates the deployment, scaling, and management of containerized applications. It allows you to run your applications on a cluster of machines, abstracting away the underlying infrastructure complexities and providing a unified way to manage and deploy your software

- Container orchestration is like managing a group of containers (small, lightweight, standalone software packages that include everything needed to run a piece of software) to work together smoothly.

Features Of Kubernetes

1. Automatic Binpacking
2. Service Discovery & Load Balancing
3. Storage Orchestration
4. Self Healing
5. Secret & Configuration Management
6. Batch Execution
7. Horizontal Scaling
8. Automatic Rollbacks & Rollouts

Features of Kubernetes

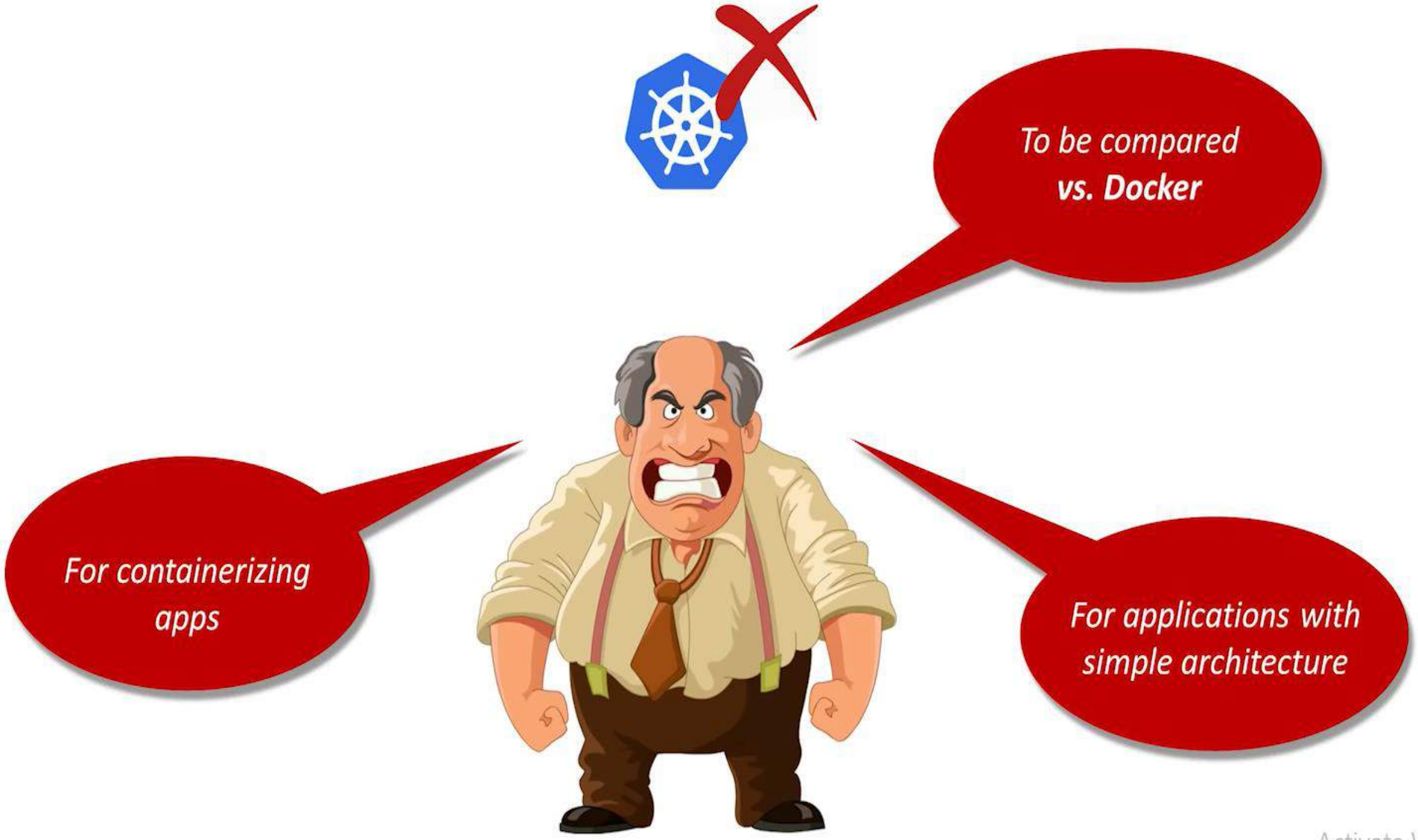
- a) Container Orchestration: Manages the deployment, scaling, and operation of application making it easy to run and scale applications in various environments.
- b) Automated Load Balancing: Distributes network traffic across multiple containers to ensure even utilization and prevent any one container from being overloaded.
- c) Scaling: Allows automatic scaling of the number of containers based on demand, ensuring applications have enough resources during periods of high traffic.
- d) Self-healing: Monitors the health of containers and automatically replaces or restarts failed containers, maintaining the desired state of the application.
- e) Rolling Updates: Enables seamless updates of applications with zero downtime, allowing for continuous delivery and integration.

Features of Kubernetes

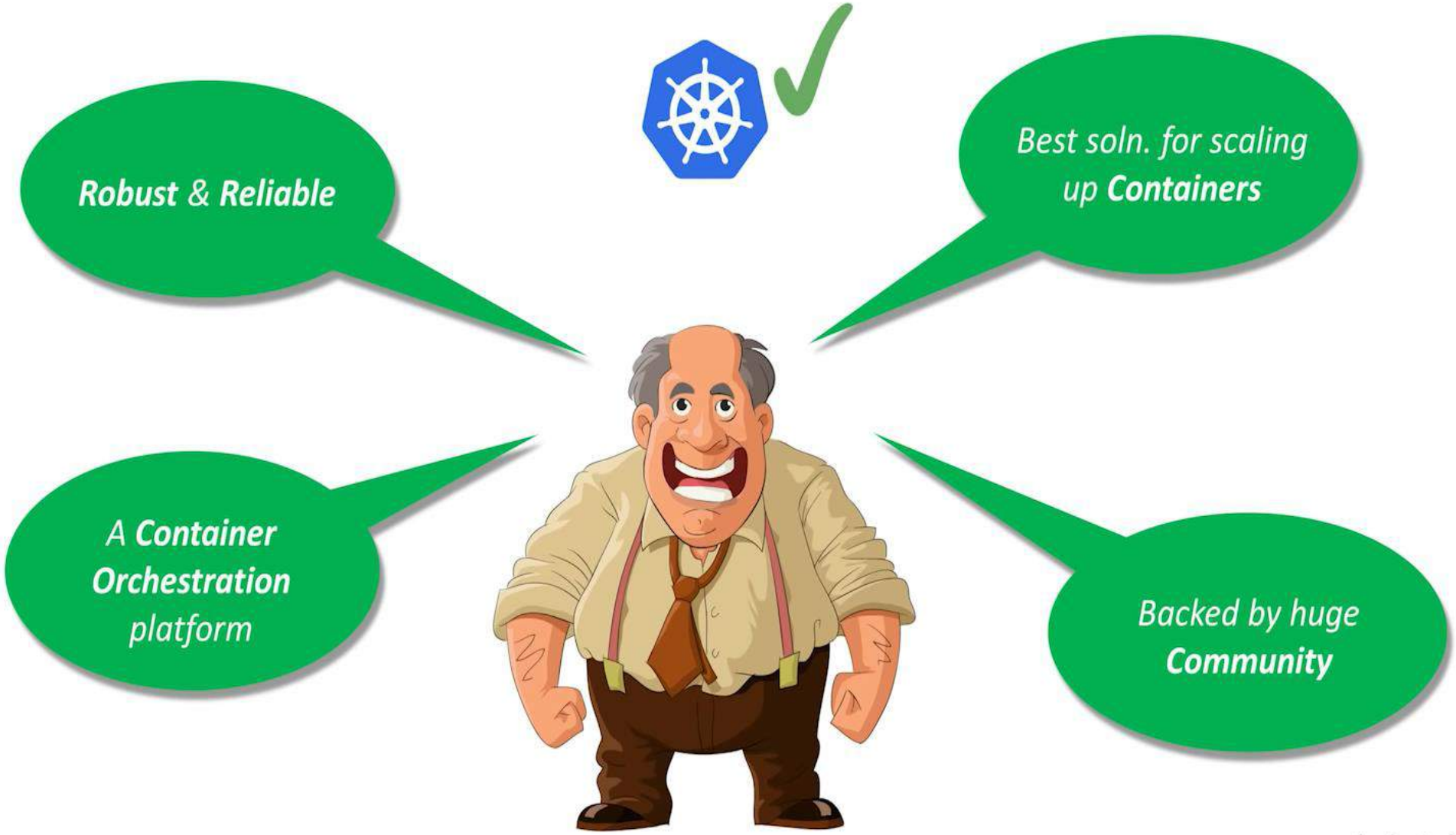
- f) Declarative Configuration: Defines the desired state of an application and Kubernetes works to ensure the actual state matches it, simplifying configuration and reducing manual intervention.
- g) Service Discovery and Load Balancing: Automatically discovers and manages the network endpoints of services, facilitating communication between containers.
- h) Storage Orchestration: Manages storage for containers, allowing them to persist data and be dynamically provisioned based on application needs.
- i) Secrets and Configuration Management: Safely manages sensitive information and

  configuration parameters, preventing exposure of sensitive data.
- j) Multi-Cloud and Hybrid Environments: Provides flexibility by

Kubernetes 'IS NOT'



Kubernetes 'ACTUALLY IS'



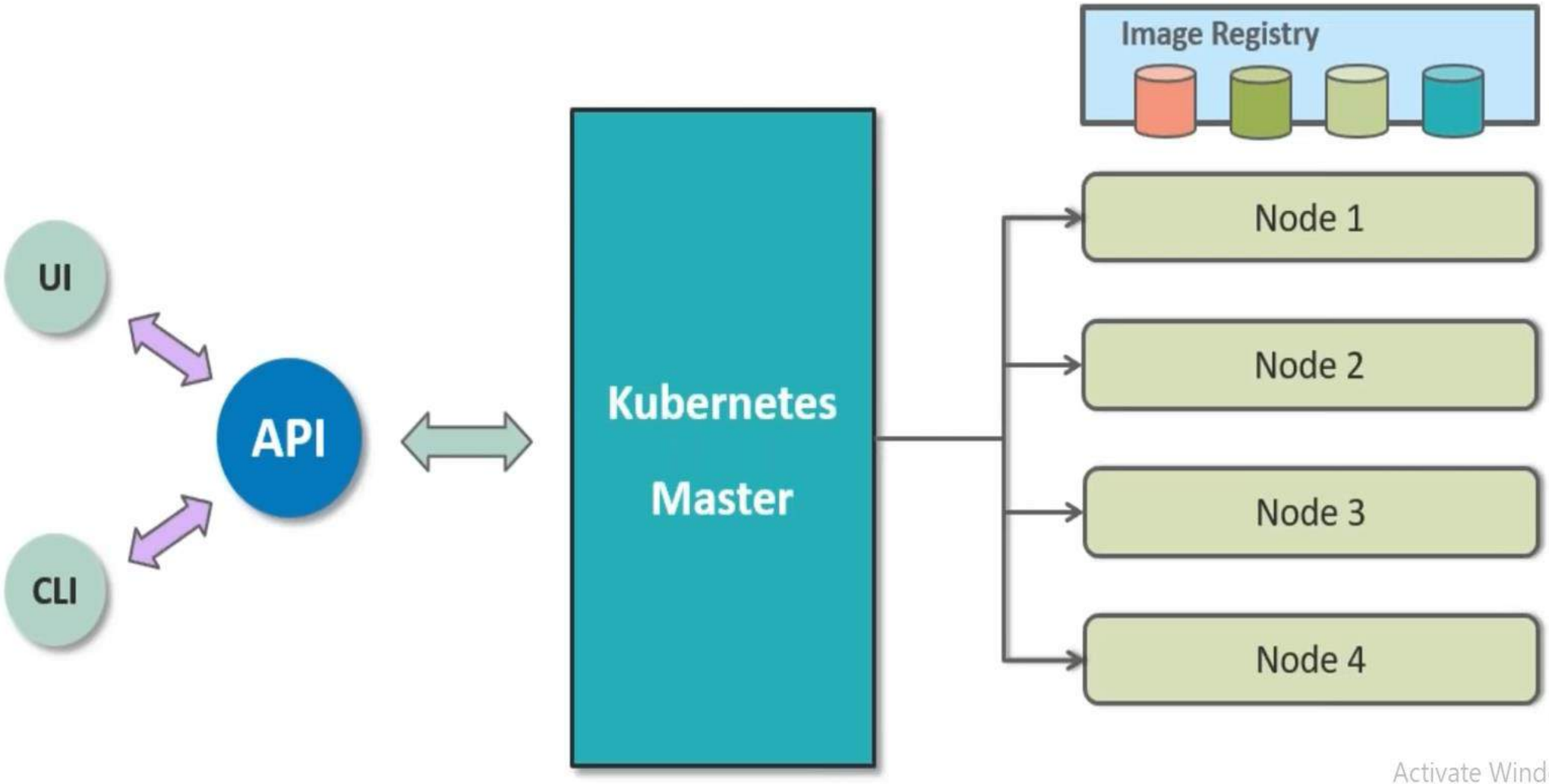
Kubernetes vs. Docker Swarm

FEATURES	Kubernetes	Docker Swarm
Installation & Cluster configuration	Complicated & time consuming	Easy & fast
GUI	GUI available	GUI not available
Scalability	Scaling up is slow compared to Swarm; but guarantees stronger cluster state	Scaling up is faster than K8S; but cluster strength not as robust
Load Balancing	Load balancing requires manual service configuration	Provides built in load balancing technique
Updates & Rollbacks	Process scheduling to maintain services while updating	Progressive updates and service health monitoring throughout the update
Data Volumes	Only shared with containers in same Pod	Can be shared with any other container
Logging & Monitoring	Inbuilt logging & monitoring tools	Only 3rd party logging & monitoring tools

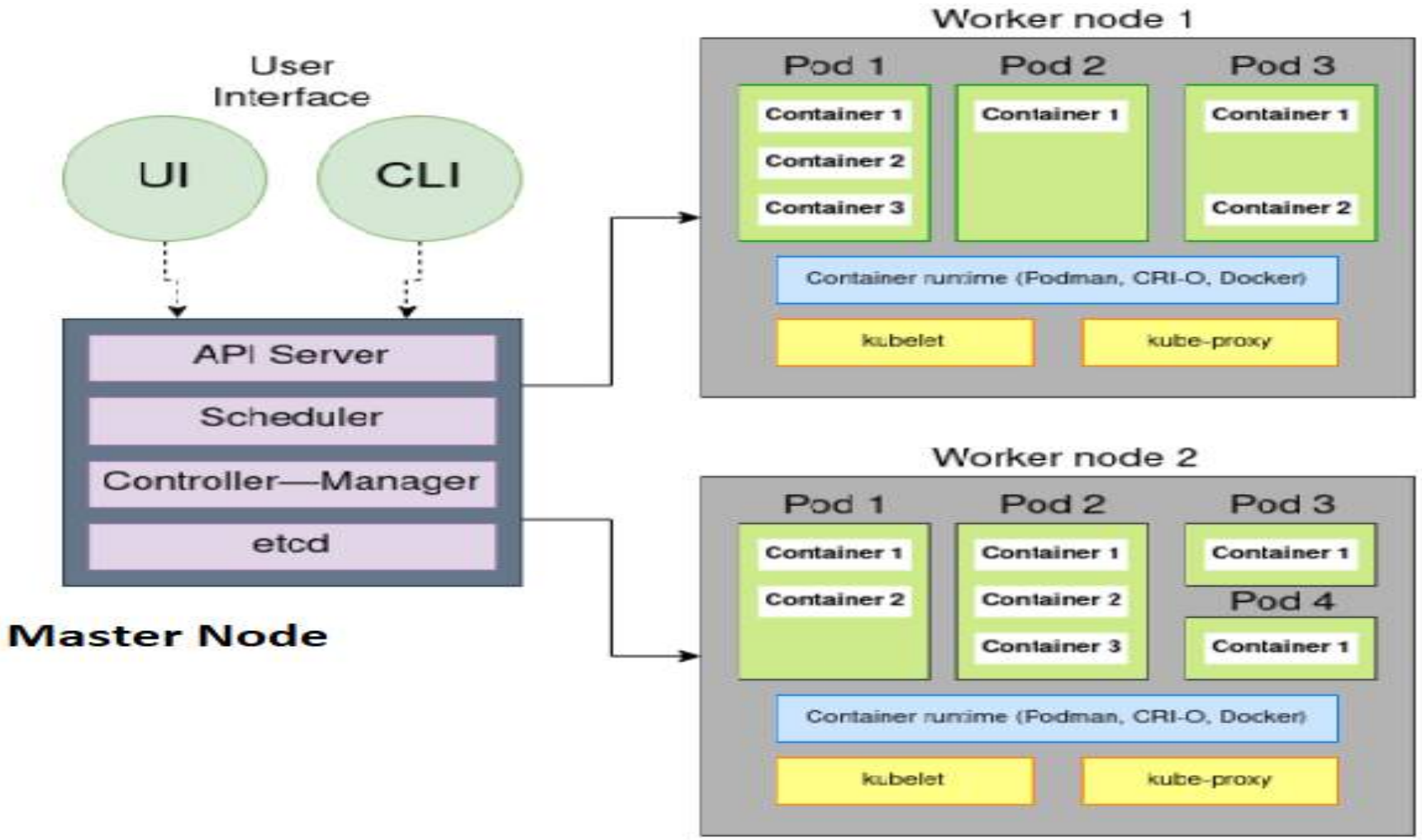
Architecture Of

KUBERNETES

Kubernetes Architecture



Kubernetes architecture



WHAT IS MINIKUBE?

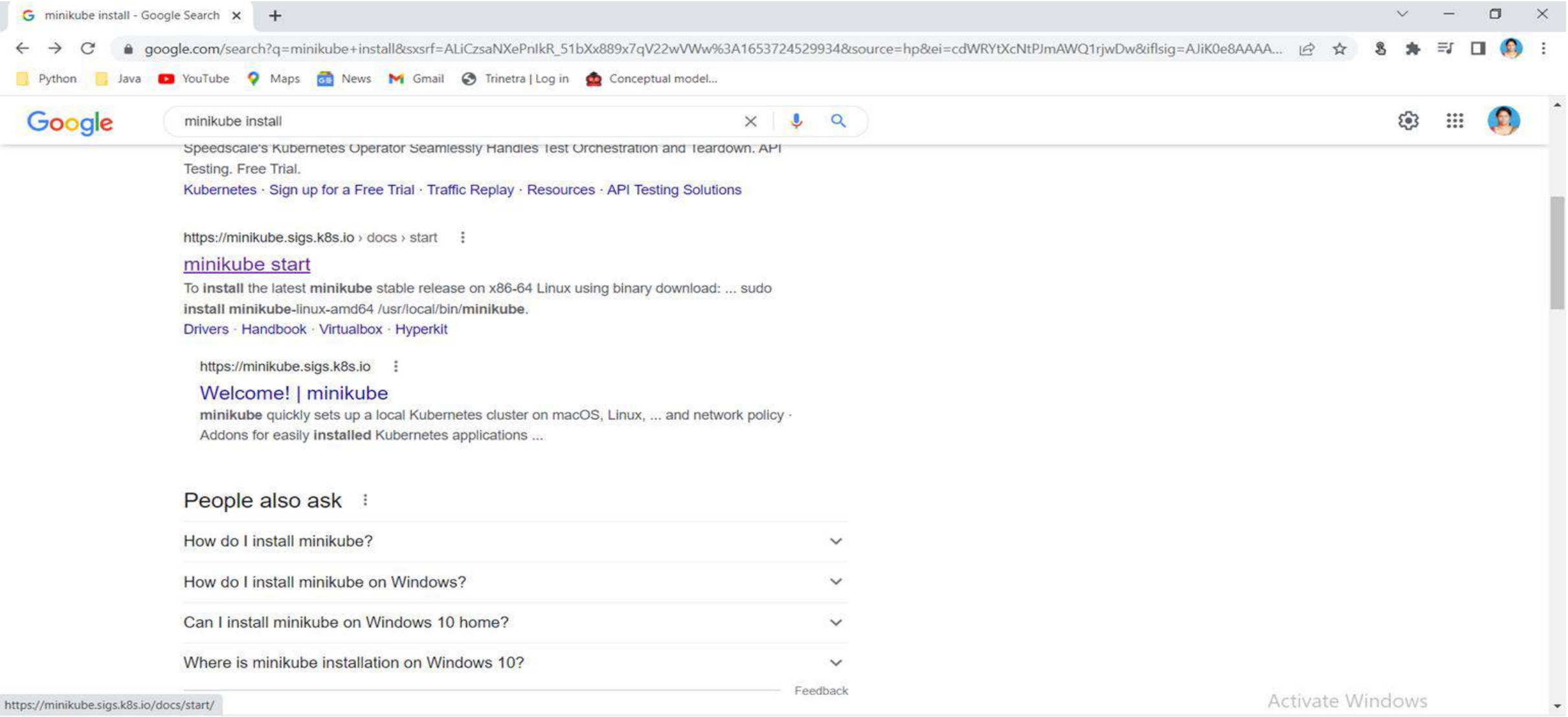
- ✓ Supports the latest Kubernetes release
- ✓ Cross-platform
- ✓ Deploy as a VM, a container, or on bare-metal
- ✓ Multiple container runtimes

minikube is a tool that lets you run Kubernetes locally. It runs a single-node Kubernetes cluster on your personal computer

MiniKube

- Minikube is a tool that allows you to run a single-node Kubernetes cluster on your local machine.
- Minikube is like a small, personal playground for Kubernetes. It is a miniature version of the Kubernetes system
- When you start Minikube, it's like opening that magic box. Minikube sets up a small, virtual city on your computer, complete with its own little streets (nodes) and houses (containers).
- These containers are where your applications live..

Search for Minikube Install in Google, Click on the highlighted link as shown



Make sure the following options are highlighted and click on latest release to download the executable file

minikube start | minikube

minikube.sigs.k8s.io/docs/start/

Python Java YouTube Maps News Gmail Trinetra | Log in Conceptual model...

minikube Community GitHub Search this site...

Search this site...

- Documentation
- Get Started!
- Handbook
  - Basic controls
  - Deploying apps
  - Kubectl
  - Accessing apps
- Addons
  - Configuration
  - Dashboard
  - Pushing images
  - Proxies and VPNs
  - Registries
  - Certificates
  - Offline usage
  - Host access
  - Network Policy

1 Installation

Click on the buttons that describe your target platform. For other architectures, see the release page for a complete list of minikube binaries.

Operating system	Linux	macOS	Windows
Architecture	x86-64		
Release type	Stable	Beta	
Installer type	.exe download	Windows Package Manager	Chocolatey

To install the latest minikube **stable** release on **x86-64 Windows** using **.exe download**:

1. Download and run the installer for the latest release.
   Or if using `PowerShell`, use this command:

```
New-Item -Path 'c:\' -Name 'minikube' -ItemType Directory -Force
Invoke-WebRequest -OutFile 'c:\minikube\minikube.exe' -Uri 'https://github.com/kubernetes/minikube/releases/latest/dc
```

Edit this page
Create child page
Create documentation issue

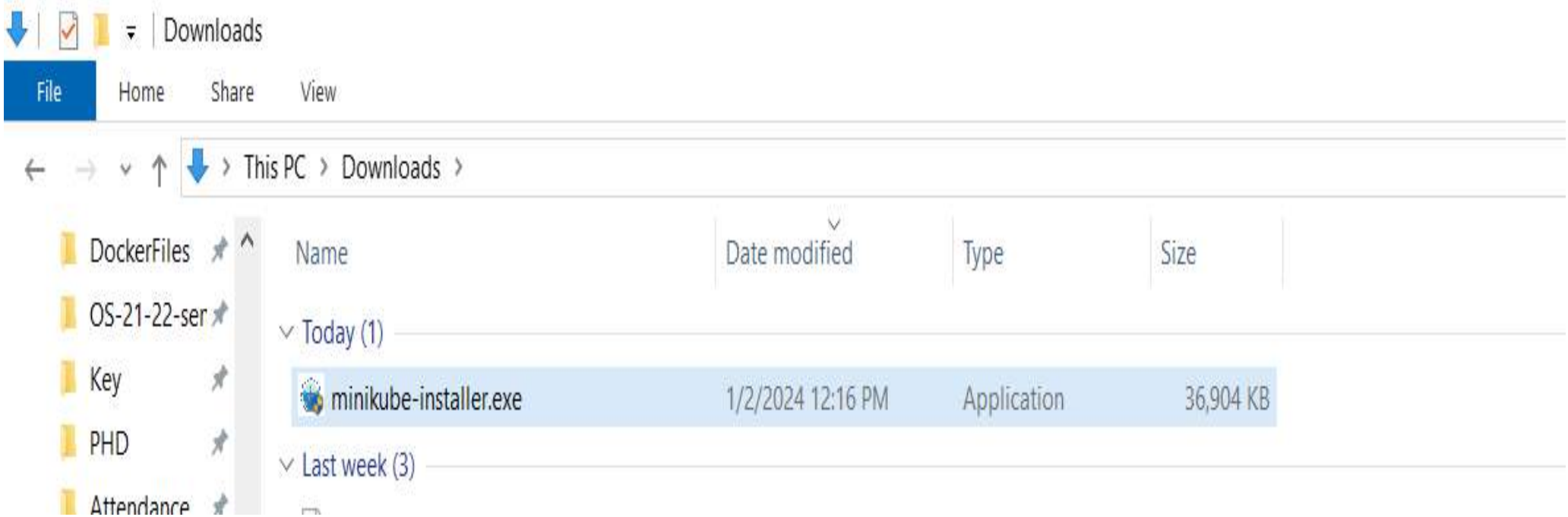
What you'll need
LoadBalancer deployments
Take the next step

https://storage.googleapis.com/minikube/releases/latest/minikube-installer.exe

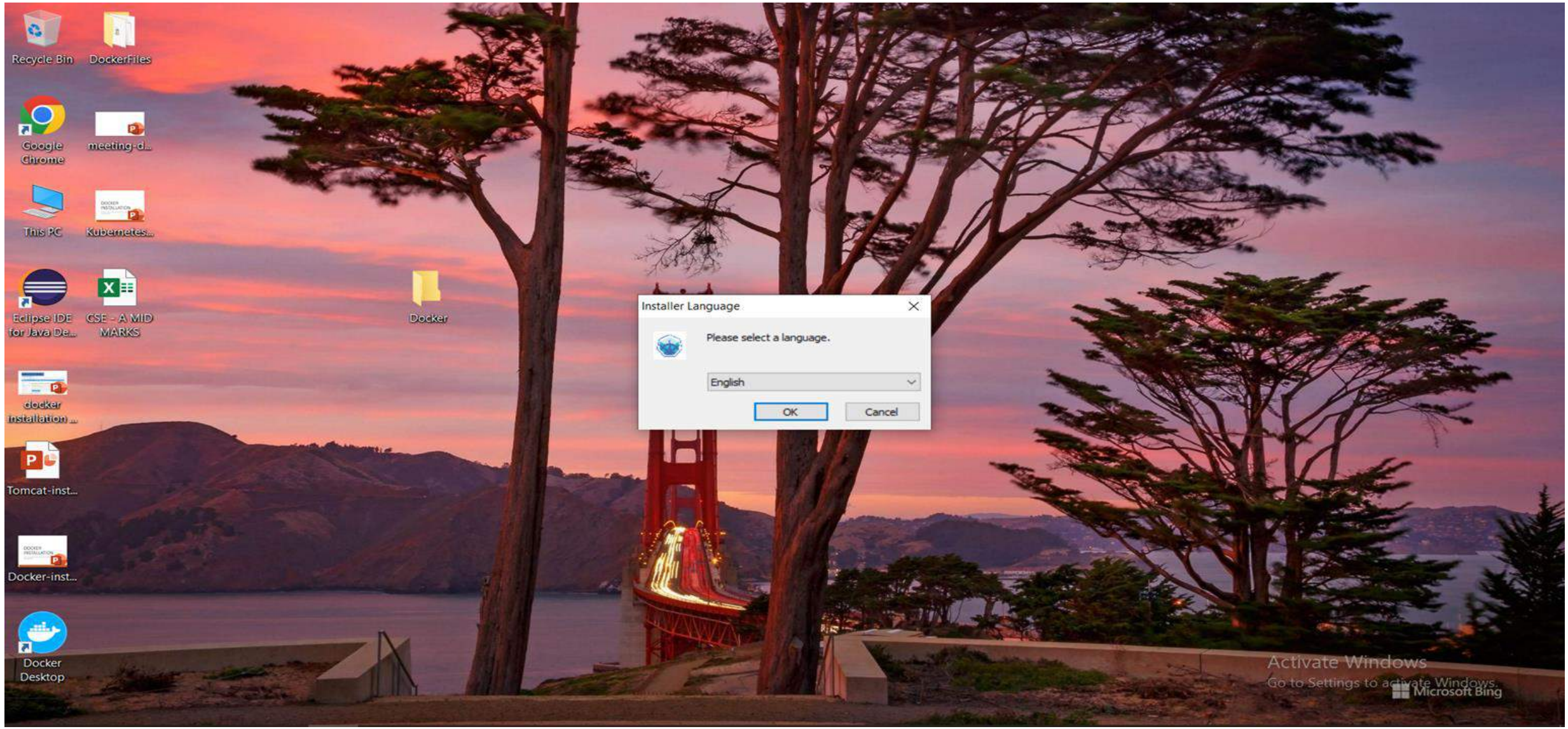
Activate Windows
Go to Settings to activate Windows.

minikube-installer.exe Show all

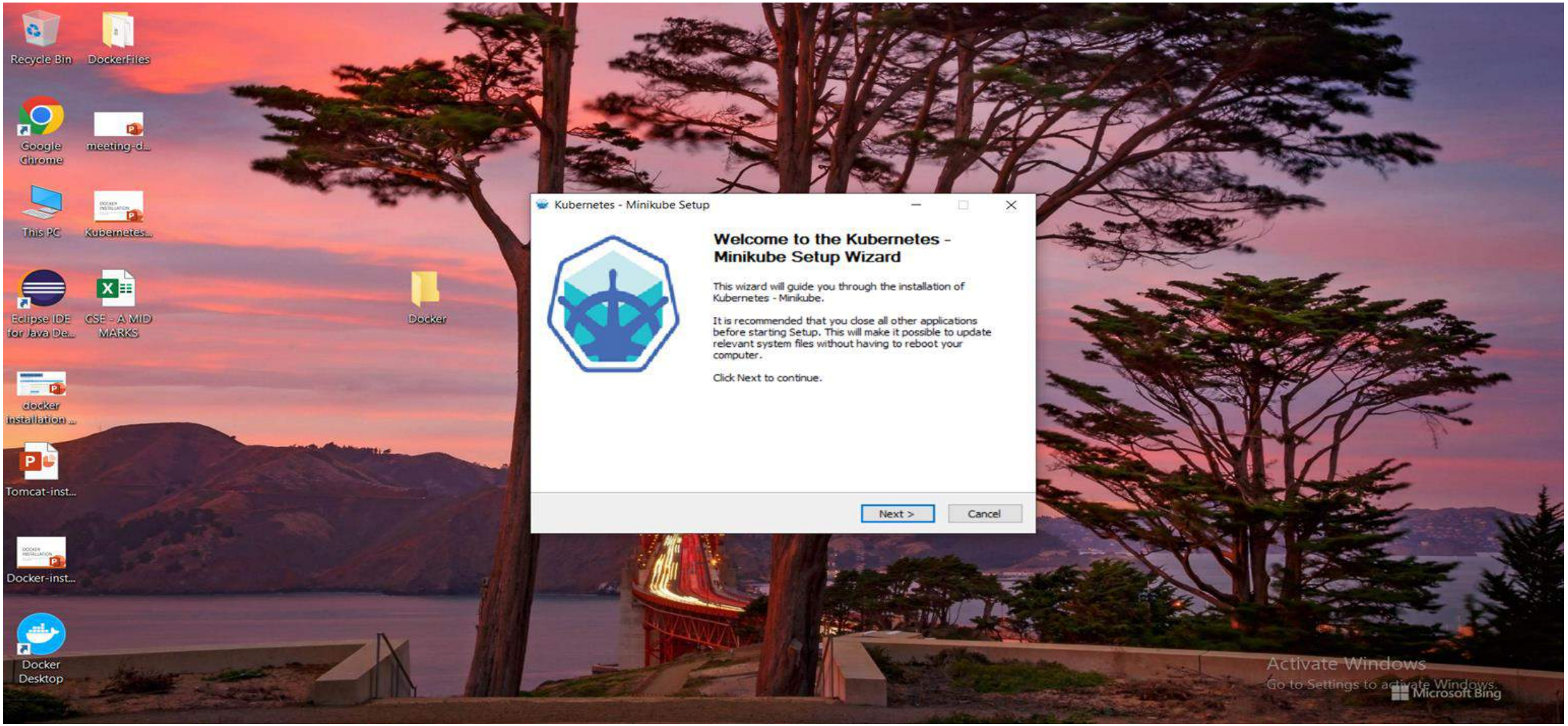
The minikube-installer is downloaded successfully



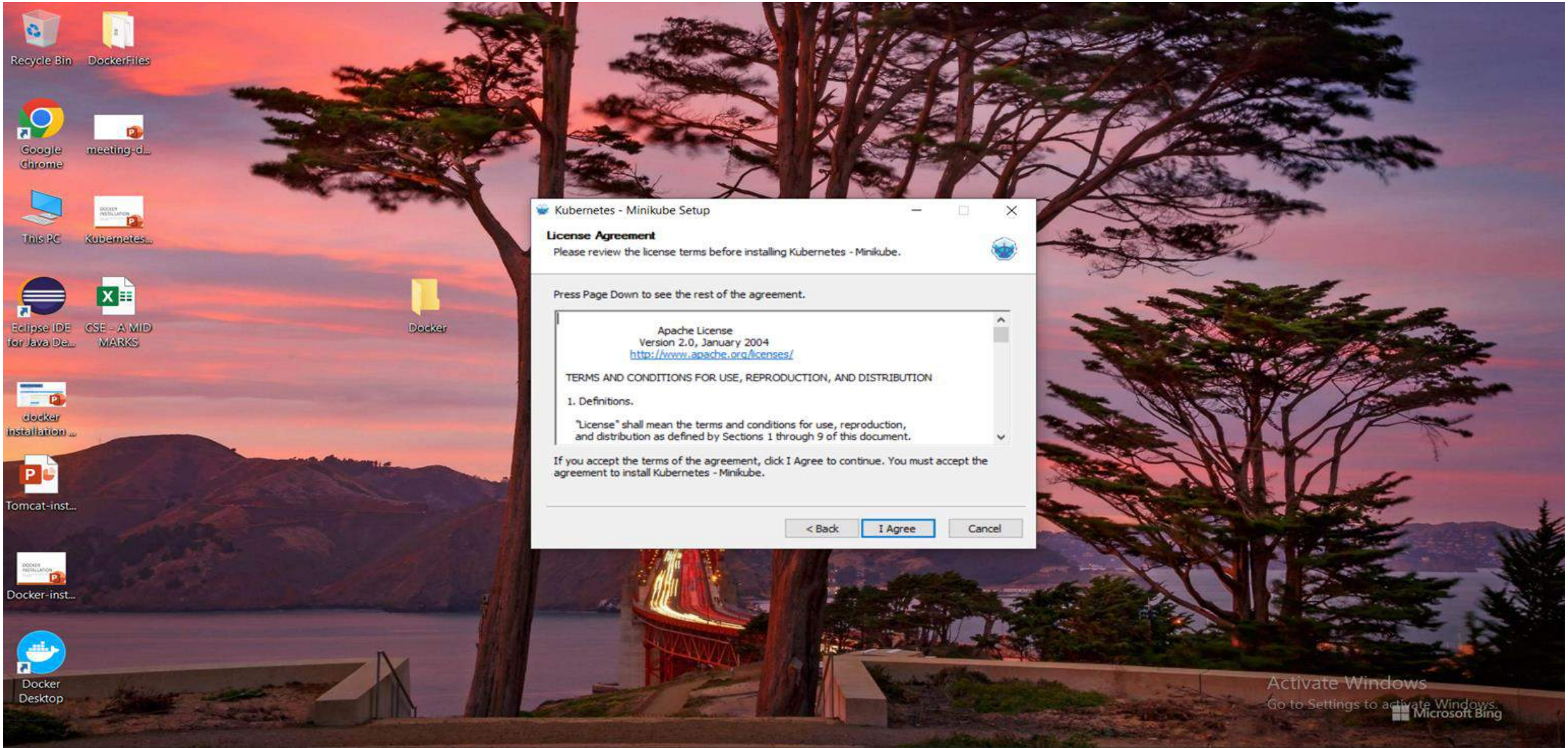
Select the language preferable English



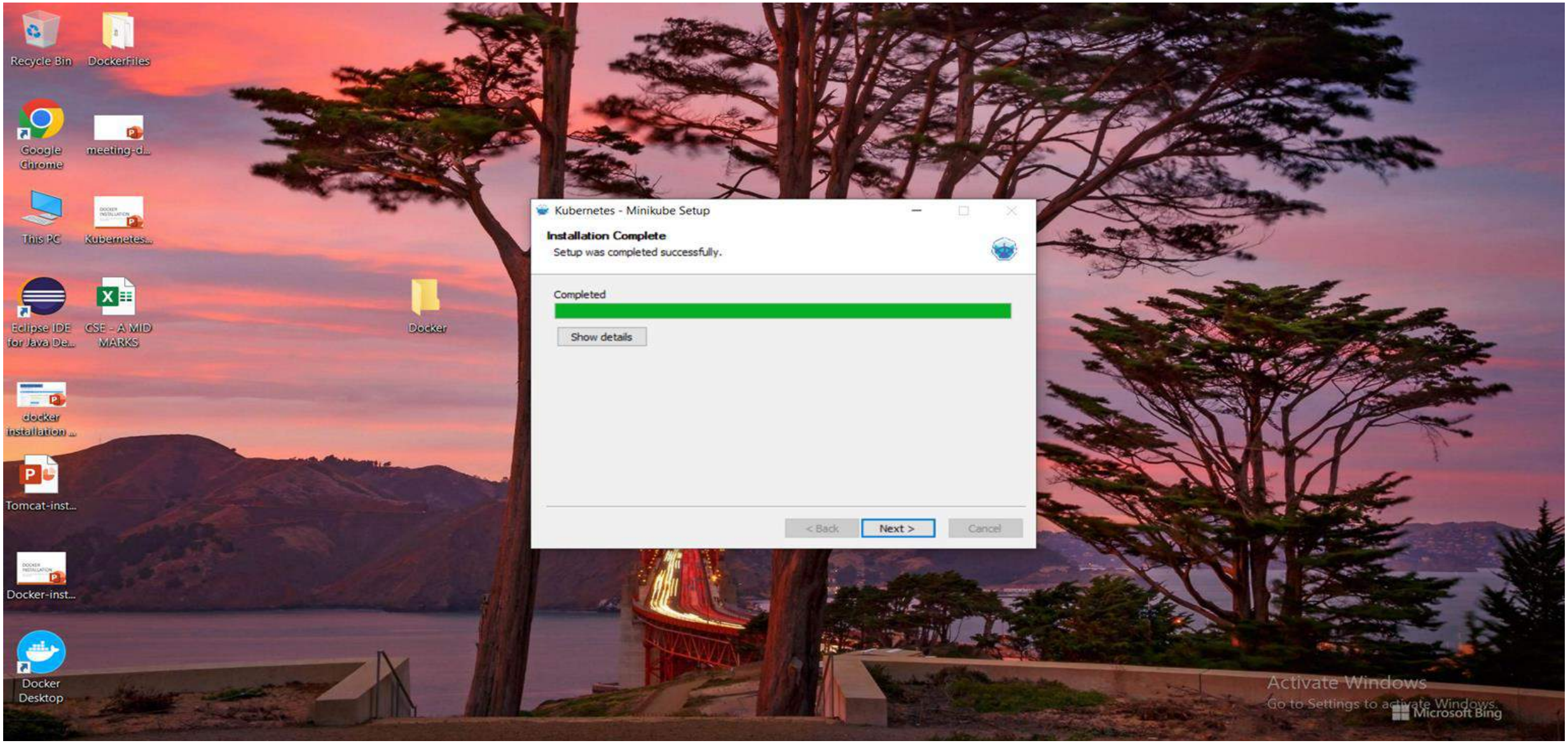
Click on Next



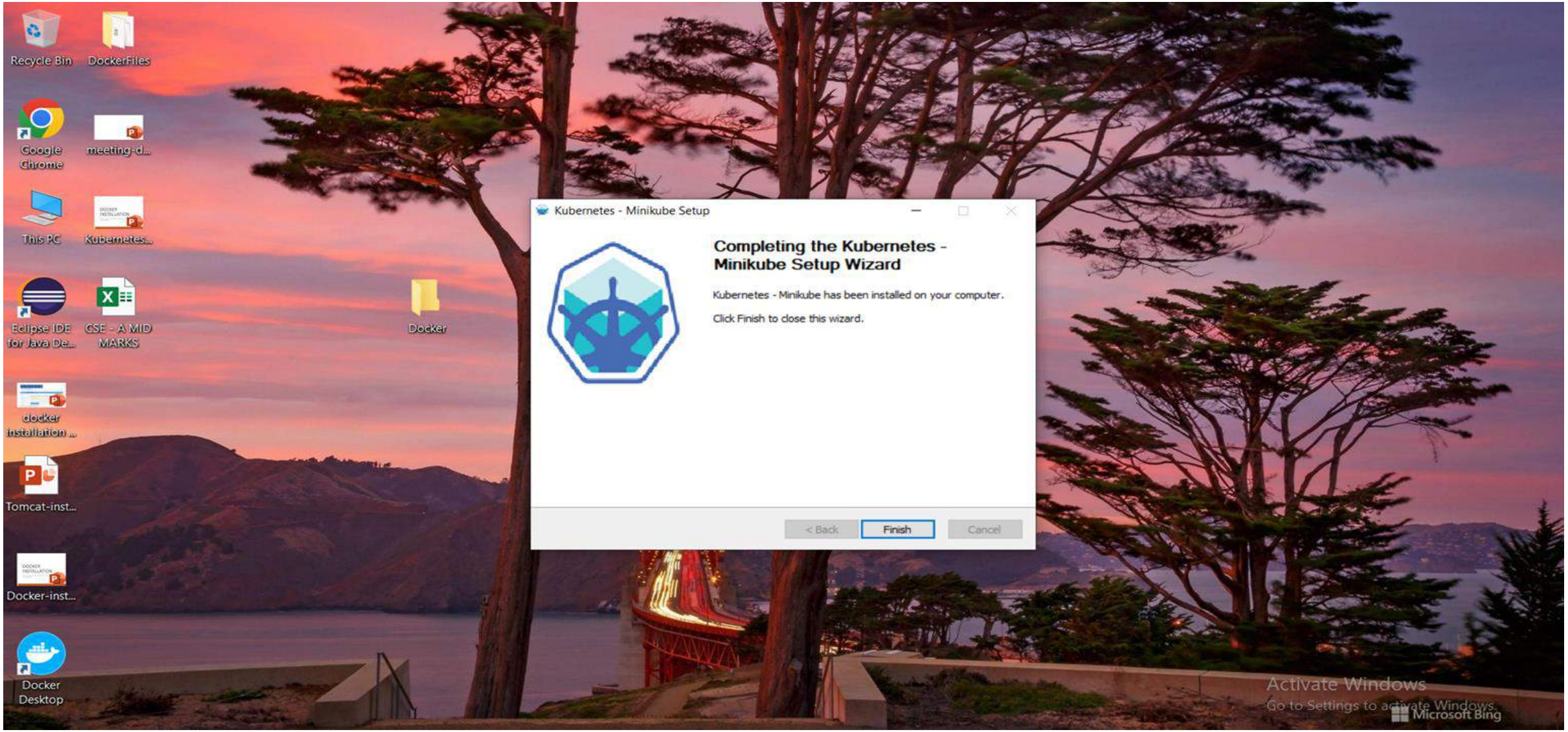
Agree the license



Installation is successfully completed, Click on Next



For learning purpose, Minikube version is sufficient, click on Finish



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

minikube start | minikube

minikube.sigs.k8s.io/docs/start/

Python Java YouTube Maps News Gmail Trinetra | Log in Conceptual model...

minikube

Community GitHub Search this site...

Search this site...

- Documentation
- Get Started!
- Handbook
  - Basic controls
  - Deploying apps
  - Kubectl
  - Accessing apps
- Addons
  - Configuration
  - Dashboard
  - Pushing images
  - Proxies and VPNs
  - Registries
  - Certificates
  - Offline usage
  - Host access
  - Network Policy

2 Start your cluster

From a terminal with administrator access (but not logged in as root), run:

```
minikube start
```

If minikube fails to start, see the drivers page for help setting up a compatible container or virtual-machine manager.

3 Interact with your cluster

If you already have kubectl installed, you can now use it to access your shiny new cluster:

```
kubectl get po -A
```

Alternatively, minikube can download the appropriate version of kubectl and you should be able to use it like this:

```
minikube kubectl -- get po -A
```

You can also make your life easier by adding the following to your shell config:

Edit this page
Create child page
Create documentation issue

What you'll need
LoadBalancer deployments
Take the next step

Activate Windows
Go to Settings to activate Windows.

minikube-installer.exe

Show all

Copy the first command as shown

minikube start | minikube

minikube.sigs.k8s.io/docs/start/

Python Java YouTube Maps News Gmail Trinetra | Log in Conceptual model...

minikube

Community GitHub Search this site...

Search this site...

- Documentation
- Get Started!
- Handbook
  - Basic controls
  - Deploying apps
  - Kubectl
  - Accessing apps
- Addons
  - Configuration
  - Dashboard
  - Pushing images
  - Proxies and VPNs
  - Registries
  - Certificates
  - Offline usage
  - Host access
  - Network Policy

2 Start your cluster

From a terminal with administrator access (but not logged in as root), run:

```
minikube start
```

Copy

Copy Ctrl+C
Copy link to highlight
Search Secure Search for "minikube start"
Print... Ctrl+P
Nimbus Screenshot
Open Ubuntu online
Inspect

If minikube ... tting up a compatible container or virtual-machine m...

3 ...

If you already have kubectl installed, you can now use it to access your shiny new cluster:

```
kubectl get po -A
```

Alternatively, minikube can download the appropriate version of kubectl and you should be able to use it like this:

```
minikube kubectl -- get po -A
```

You can also make your life easier by adding the following to your shell config:

Edit this page
Create child page
Create documentation issue

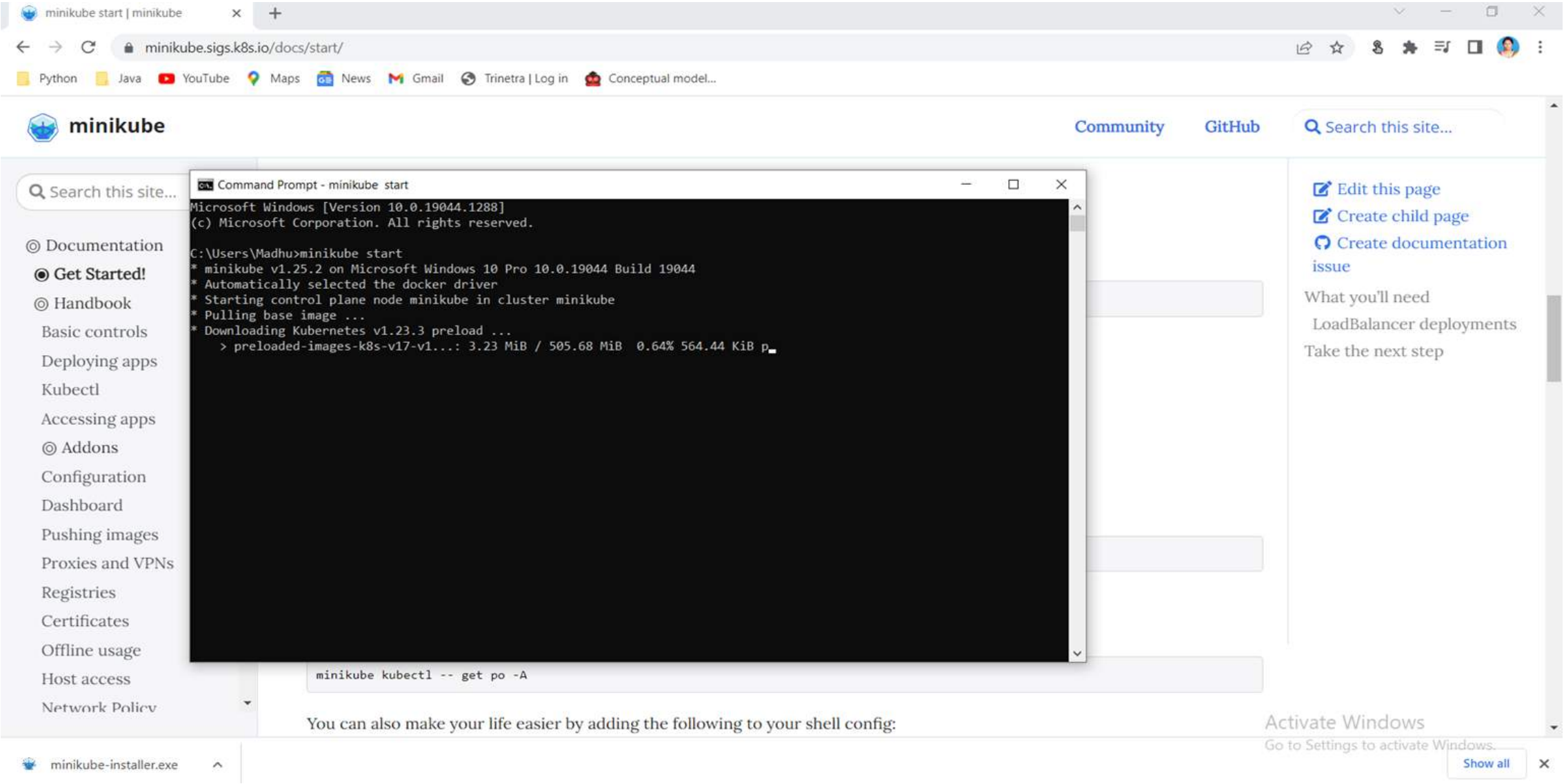
What you'll need
LoadBalancer deployments
Take the next step

Activate Windows
Go to Settings to activate Windows.

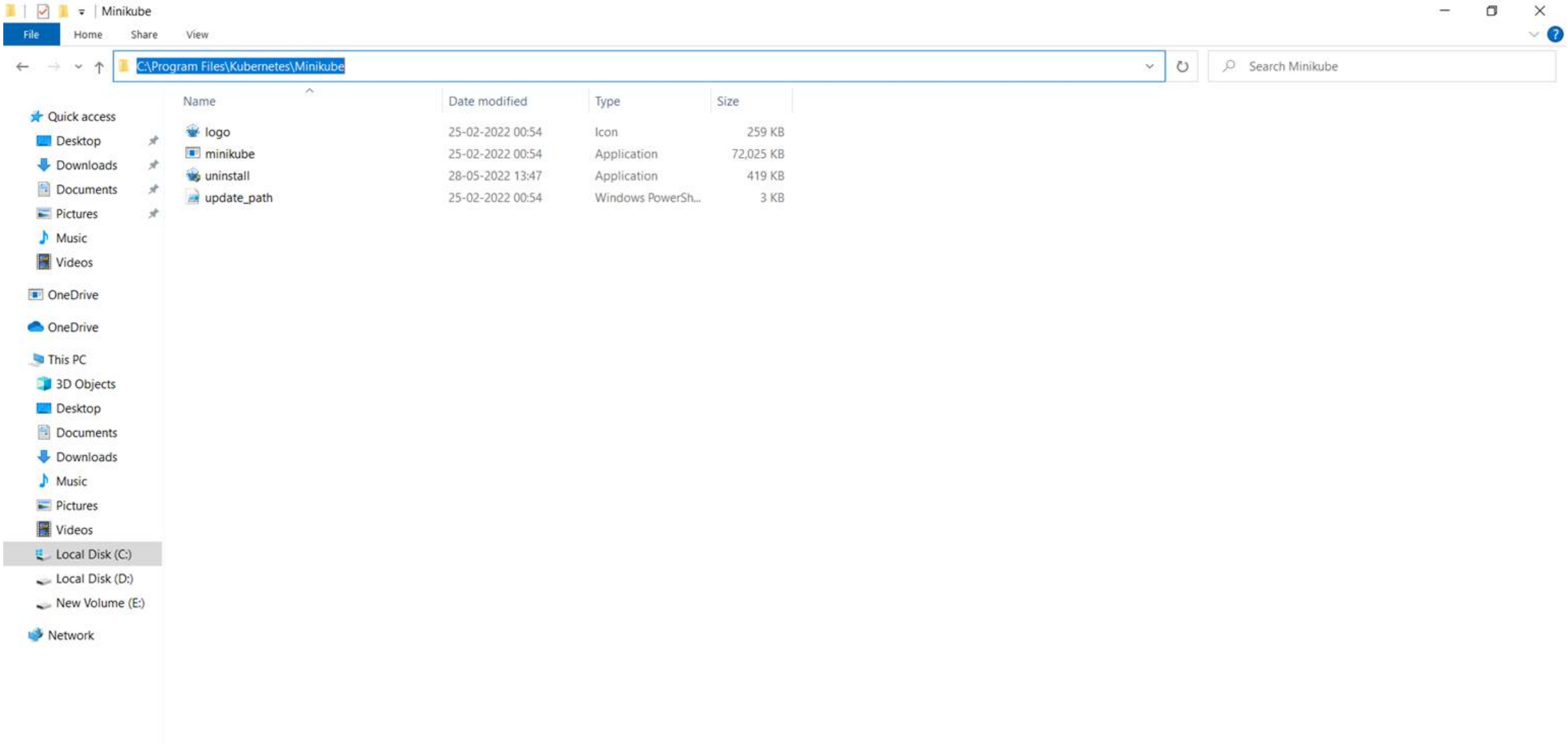
minikube-installer.exe

Show all

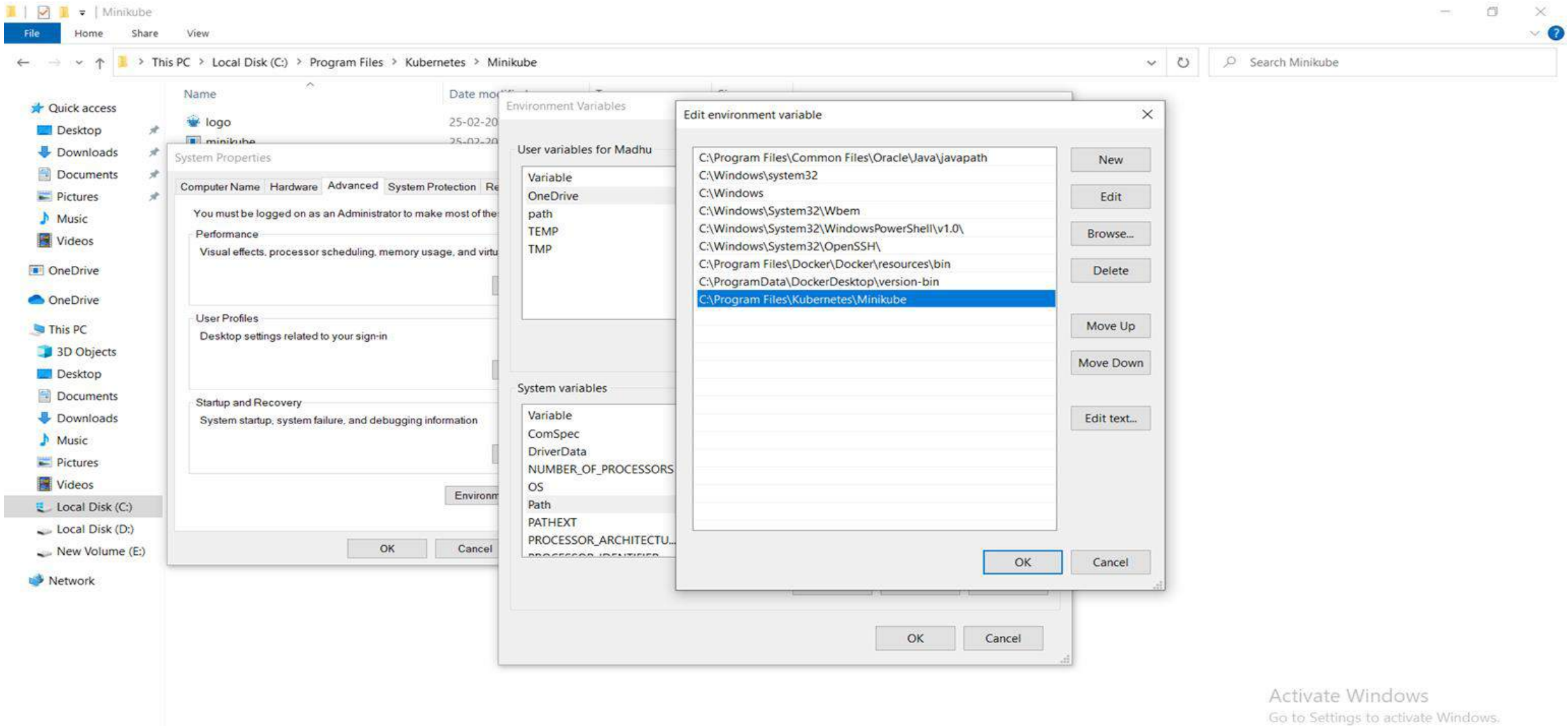
Open the command prompt/ Windows PowerShell always in administrative mode and paste the command as shown



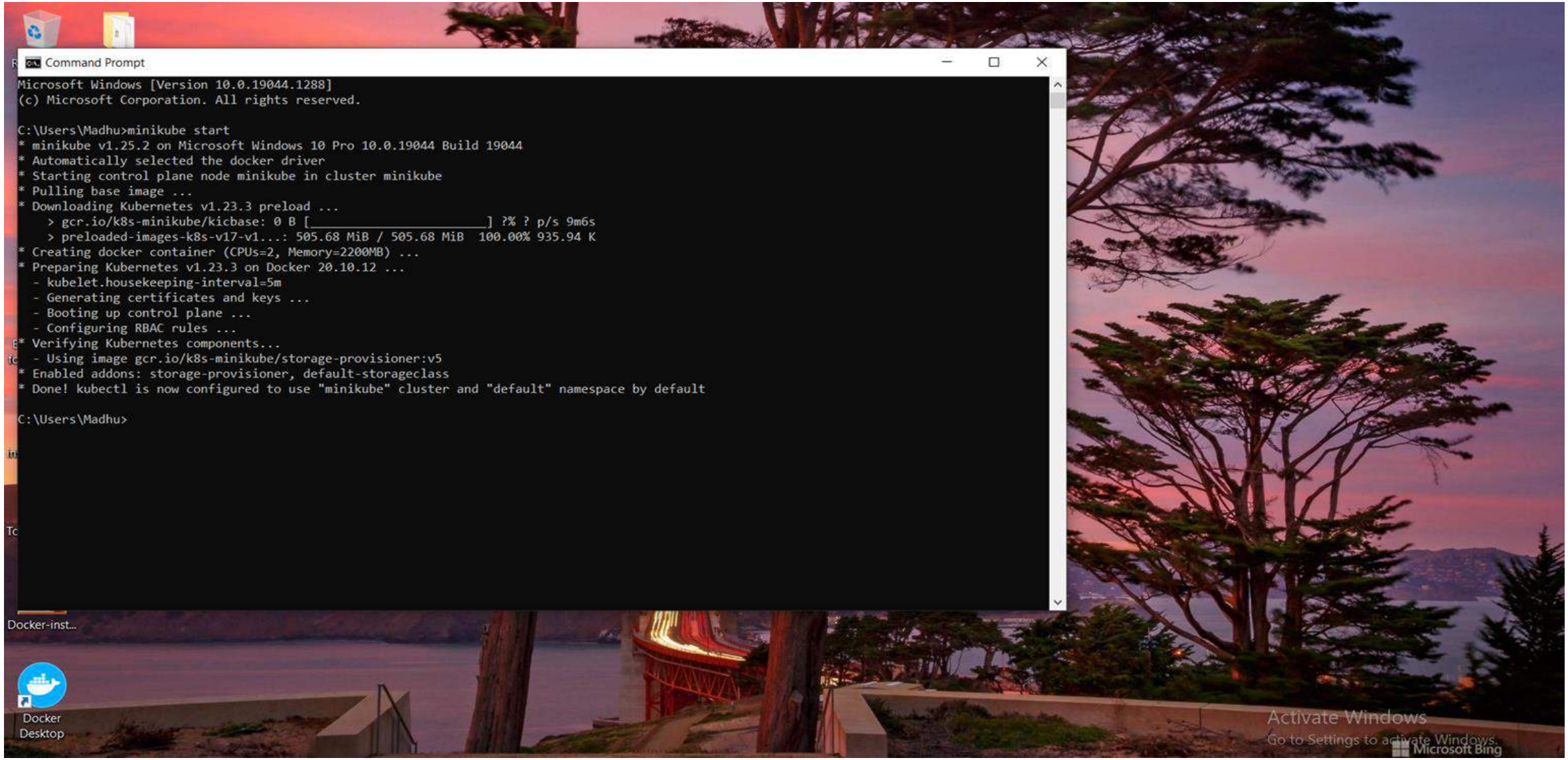
Sometimes it gives an error saying, internal command, in that case we need to set the path in environmental variables, firstly copy the minikube path as shown



Open Environmental variables->DoubleClick path in System variables and paste the copied path as shown->click Ok



Restart the command prompt/Windows PowerShell and type the minikube start command again



What happens when we start minikube?

```
minikube start --driver=virtualbox
```

- It is used to start the cluster on our single-node machine

When running it for the very first time, the following too take place:

- Installs kubernetes
- Installs `kubectl` - the command-line utility to interact with the Kubernetes cluster

Exercise - 1

Open Windows PowerShell in Administrator Mode and login into Docker

Administrator: Windows PowerShell

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Windows\system32> docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https:
//hub.docker.com to create one.
Username: budarajumadhurika
Password:
Login Succeeded

Logging in with your password grants your terminal complete access to your account.
For better security, log in with a limited-privilege personal access token. Learn more at https://docs.docker.com/go/
access-tokens/
PS C:\Windows\system32>
```

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.

Administrator: Windows PowerShell

```
PS C:\Windows\system32> minikube version
minikube version: v1.25.2
commit: 362d5fdc0a3dbee389b3d3f1034e8023e72bd3a7
PS C:\Windows\system32> minikube start
* minikube v1.25.2 on Microsoft Windows 10 Pro 10.0.19044 Build 19044
* Using the hyperv driver based on user configuration
* Starting control plane node minikube in cluster minikube
* Creating hyperv VM (CPUs=2, Memory=2200MB, Disk=20000MB) ...
* Preparing Kubernetes v1.23.3 on Docker 20.10.12 ...
  - kubelet.housekeeping-interval=5m
  - Generating certificates and keys ...
  - Booting up control plane ...
  - Configuring RBAC rules ...
* Verifying Kubernetes components...
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Enabled addons: storage-provisioner, default-storageclass
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
PS C:\Windows\system32>
```

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.

Administrator: Windows PowerShell

```
PS C:\Windows\system32> minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured

PS C:\Windows\system32>
```

Open Windows PowerShell in Administrator Mode and login into Docker

Administrator: Windows PowerShell

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Windows\system32> docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https:
//hub.docker.com to create one.
Username: budarajumadhurika
Password:
Login Succeeded

Logging in with your password grants your terminal complete access to your account.
For better security, log in with a limited-privilege personal access token. Learn more at https://docs.docker.com/go/
access-tokens/
PS C:\Windows\system32>
```

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.

Administrator: Windows PowerShell

```
PS C:\Windows\system32> minikube version
minikube version: v1.25.2
commit: 362d5fdc0a3dbee389b3d3f1034e8023e72bd3a7
PS C:\Windows\system32> minikube start
* minikube v1.25.2 on Microsoft Windows 10 Pro 10.0.19044 Build 19044
* Using the hyperv driver based on user configuration
* Starting control plane node minikube in cluster minikube
* Creating hyperv VM (CPUs=2, Memory=2200MB, Disk=20000MB) ...
* Preparing Kubernetes v1.23.3 on Docker 20.10.12 ...
  - kubelet.housekeeping-interval=5m
  - Generating certificates and keys ...
  - Booting up control plane ...
  - Configuring RBAC rules ...
* Verifying Kubernetes components...
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Enabled addons: storage-provisioner, default-storageclass
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
PS C:\Windows\system32>
```

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.

Administrator: Windows PowerShell

```
PS C:\Windows\system32> minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured

PS C:\Windows\system32>
```

Lets deploy an application in Kubernetes

kubectl create deployment myngnix --image=nginx

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.

Select Administrator: Windows PowerShell

```
PS C:\Windows\system32> minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured

PS C:\Windows\system32> kubectl create deployment mynginx --image=nginx
```

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 "myngnix" 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.

Administrator: Windows PowerShell

```
PS C:\Windows\system32> minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured

PS C:\Windows\system32> kubectl create deployment mynginx --image=nginx
deployment.apps/mynginx created
PS C:\Windows\system32>
```

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

Administrator: Windows PowerShell

```
PS C:\Windows\system32> minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured

PS C:\Windows\system32> kubectl create deployment mynginx --image=nginx
deployment.apps/mynginx created
PS C:\Windows\system32> kubectl get deployment
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
mynginx   0/1     1            0           35s
PS C:\Windows\system32> kubectl get pods
NAME                       READY   STATUS              RESTARTS   AGE
mynginx-6b78685d4d-8cqxr   0/1     ContainerCreating   0          92s
PS C:\Windows\system32>
```

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.

Administrator: Windows PowerShell

```
PS C:\Windows\system32> kubectl describe pods
Name:             mynginx-6b78685d4d-8cqxr
Namespace:        default
Priority:         0
Node:             minikube/172.23.45.150
Start Time:       Thu, 15 Dec 2022 11:53:58 +0530
Labels:           app=mynginx
                  pod-template-hash=6b78685d4d
Annotations:      <none>
Status:           Running
IP:               172.17.0.3
IPs:
  IP:           172.17.0.3
Controlled By:  ReplicaSet/mynginx-6b78685d4d
Containers:
  nginx:
    Container ID:   docker://7cc34f8ebbcd648c9688126cf3c8511eacaa66f1b62199b26199f02b76b735ca
    Image:          nginx
    Image ID:       docker-pullable://nginx@sha256:75263be7e5846fc69cb6c42553ff9c93d653d769b94917dbda71d42d3f3c00d3
    Port:           <none>
    Host Port:      <none>
    State:          Running
      Started:      Thu, 15 Dec 2022 11:56:22 +0530
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
```

Continuation of previous

Administrator: Windows PowerShell

```
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-qkjsz (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  kube-api-access-qkjsz:
    Type:                    Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds:  3607
    ConfigMapName:           kube-root-ca.crt
    ConfigMapOptional:       <nil>
    DownwardAPI:             true
QoS Class:                   BestEffort
Node-Selectors:              <none>
Tolerations:                 node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                             node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type    Reason     Age   From               Message
  ----    ------     ----  ----               -------
  Normal  Scheduled  3m4s  default-scheduler  Successfully assigned default/mynginx-6b78685d4d-8cqxr to minikube
  Normal  Pulling    3m3s  kubelet            Pulling image "nginx"
  Normal  Pulled     40s   kubelet            Successfully pulled image "nginx" in 2m23.084699052s
  Normal  Created    40s   kubelet            Created container nginx
  Normal  Started    40s   kubelet            Started container nginx
PS C:\Windows\system32> kubectl get deployment
```

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.

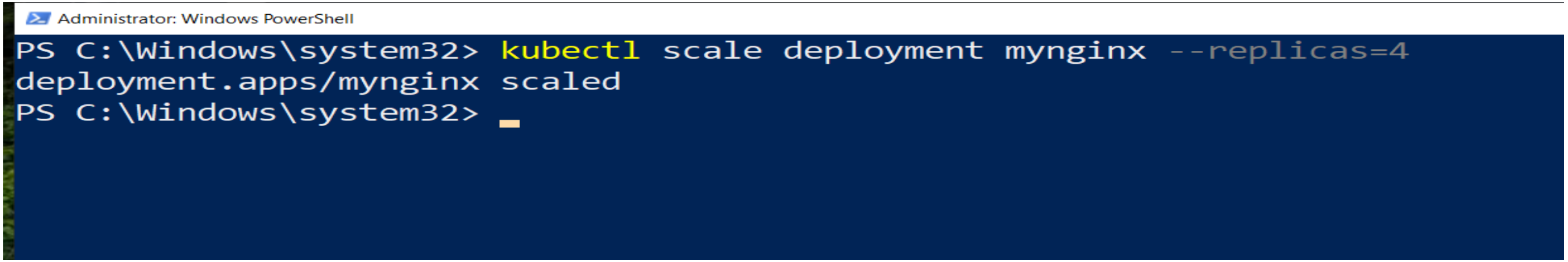
```
Administrator: Windows PowerShell
  PodScheduled      True
Volumes:
  kube-api-access-qkjsz:
    Type:                    Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds:  3607
    ConfigMapName:           kube-root-ca.crt
    ConfigMapOptional:       <nil>
    DownwardAPI:             true
QoS Class:                   BestEffort
Node-Selectors:              <none>
Tolerations:                 node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                             node.kubernetes.io/unreachable:NoExecute op=Exists for 300s

Events:
  Type    Reason     Age   From               Message
  ----    ------     ----  ----               -------
  Normal  Scheduled  3m4s  default-scheduler  Successfully assigned default/mynginx-6b78685d4d-8cqxr to minikube
  Normal  Pulling    3m3s  kubelet            Pulling image "nginx"
  Normal  Pulled     40s   kubelet            Successfully pulled image "nginx" in 2m23.084699052s
  Normal  Created    40s   kubelet            Created container nginx
  Normal  Started    40s   kubelet            Started container nginx
PS C:\Windows\system32> kubectl get deployment
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
mynginx   1/1     1            1           3m15s
PS C:\Windows\system32> kubectl get pods
NAME                       READY   STATUS    RESTARTS   AGE
mynginx-6b78685d4d-8cqxr   1/1     Running   0          3m20s
PS C:\Windows\system32>
```

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.



- Now if we check the following command,

kubectl get deployment

We can see that 4 instances of mynginx are available

Administrator: Windows PowerShell

```
PS C:\Windows\system32> kubectl scale deployment mynginx --replicas=4
deployment.apps/mynginx scaled
PS C:\Windows\system32> kubectl get deployment
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
mynginx   4/4     4            4           8m23s
PS C:\Windows\system32>
```

- Now if we check the following command,

kubectl get pods

We can see that 4 instances of mynginx are running

Administrator: Windows PowerShell

```
PS C:\Windows\system32> kubectl scale deployment mynginx --replicas=4
deployment.apps/mynginx scaled
PS C:\Windows\system32> kubectl get deployment
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
mynginx   4/4     4            4           8m23s
PS C:\Windows\system32> kubectl get pods
NAME                       READY   STATUS    RESTARTS   AGE
mynginx-6b78685d4d-8cqxr   1/1     Running   0          9m3s
mynginx-6b78685d4d-hrc9h   1/1     Running   0          77s
mynginx-6b78685d4d-nk76q   1/1     Running   0          77s
mynginx-6b78685d4d-qlr5k   1/1     Running   0          77s
PS C:\Windows\system32>
```

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)**

Select Administrator: Windows PowerShell

```
PS C:\Windows\system32> kubectl scale deployment mynginx --replicas=4
deployment.apps/mynginx scaled
PS C:\Windows\system32> kubectl get deployment
NAME       READY   UP-TO-DATE   AVAILABLE   AGE
mynginx    4/4     4            4           8m23s
PS C:\Windows\system32> kubectl get pods
NAME                          READY   STATUS    RESTARTS   AGE
mynginx-6b78685d4d-8cqxr      1/1     Running   0          9m3s
mynginx-6b78685d4d-hrc9h      1/1     Running   0          77s
mynginx-6b78685d4d-nk76q      1/1     Running   0          77s
mynginx-6b78685d4d-qlr5k      1/1     Running   0          77s
PS C:\Windows\system32> kubectl describe pod mynginx-6b78685d4d-hrc9h
```

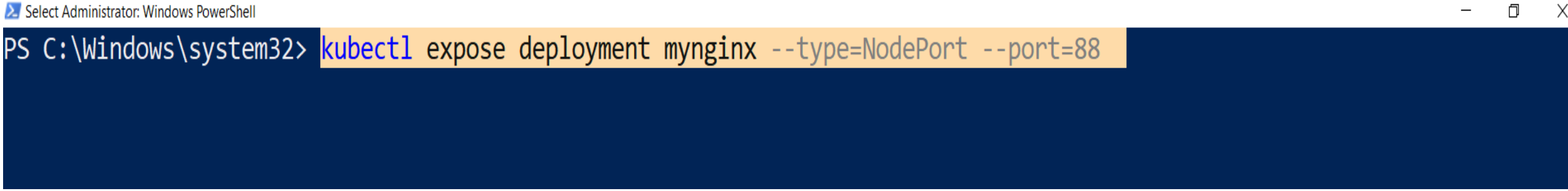
Select Administrator: Windows PowerShell

```
  IP:           172.17.0.6
Controlled By:  ReplicaSet/mynginx-6b78685d4d
Containers:
  nginx:
    Container ID:   docker://54d65c50c3abab016f139a68e356578ecaa94bc987651af515f18bd605df5137
    Image:          nginx
    Image ID:       docker-pullable://nginx@sha256:75263be7e5846fc69cb6c42553ff9c93d653d769b94917dbda71d42d3f3c00d3
    Port:           <none>
    Host Port:      <none>
    State:          Running
      Started:      Thu, 15 Dec 2022 12:01:54 +0530
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-4bc8t (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  kube-api-access-4bc8t:
    Type:                    Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds:  3607
    ConfigMapName:           kube-root-ca.crt
```

Select Administrator: Windows PowerShell

```
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-4bc8t (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  kube-api-access-4bc8t:
    Type:                    Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds:  3607
    ConfigMapName:           kube-root-ca.crt
    ConfigMapOptional:       <nil>
    DownwardAPI:             true
QoS Class:                   BestEffort
Node-Selectors:              <none>
Tolerations:                 node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                             node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type    Reason     Age    From               Message
  ----    ------     ----   ----               -------
  Normal  Scheduled  3m30s  default-scheduler  Successfully assigned default/mynginx-6b78685d4d-hrc9h to minikube
  Normal  Pulling    3m29s  kubelet            Pulling image "nginx"
  Normal  Pulled     3m21s  kubelet            Successfully pulled image "nginx" in 8.150664403s
  Normal  Created    3m21s  kubelet            Created container nginx
  Normal  Started    3m20s  kubelet            Started container nginx
PS C:\Windows\system32>
```

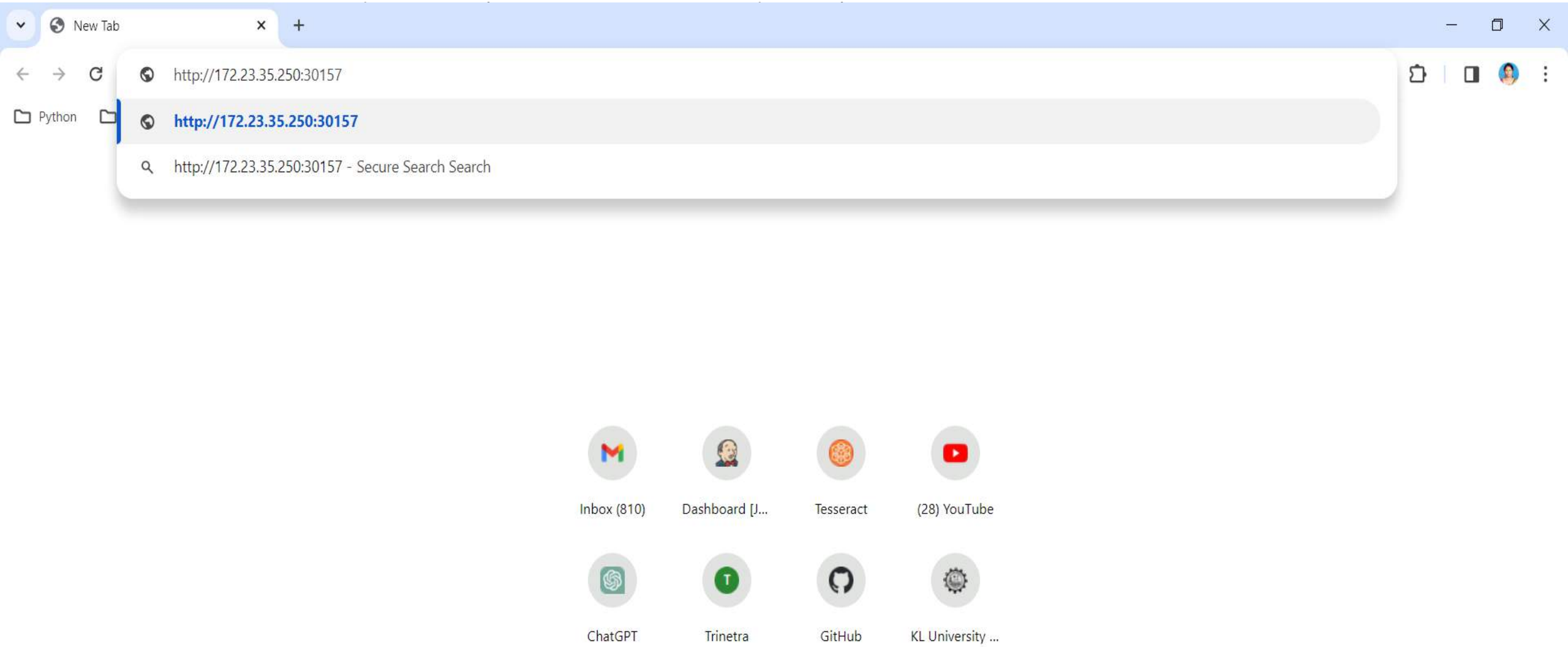
- **kubectl expose deployment mynginx --type=NodePort --port=88**
- This command is used to make a service publicly accessible from outside the Kubernetes cluster.
- **expose deployment mynginx:** Creates a service to expose the "mynginx" deployment.
- **--type=NodePort:** Specifies the type of service. 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.



- **minikube service mynginx –url**
- This command is used with Minikube, a tool for running Kubernetes locally, to get the URL that you can use to access a service deployed in your Minikube cluster.
- **minikube:** The command-line tool for managing and running a local Kubernetes cluster using a virtual machine.
- **service mynginx:** Refers to the Kubernetes service named "mynginx."
- **--url:** Requests Minikube to provide the URL that you can use to access the service.
- When you run this command, Minikube will return the full URL, including the IP address and port, that you can use in your web browser or any HTTP client to interact with the "mynginx" service running in your local Kubernetes cluster. This makes it easy to test and access applications as if they were running in a real Kubernetes environment.

Select Administrator: Windows PowerShell

```
PS C:\Windows\system32> kubectl expose deployment mynginx --type=NodePort --port=88
service/mynginx exposed
PS C:\Windows\system32> minikube service mynginx --url
http://172.23.45.150:31242
PS C:\Windows\system32>
```


New Tab
http://172.23.35.250:30157
Python
http://172.23.35.250:30157
http://172.23.35.250:30157 - Secure Search Search
Inbox (810)
Dashboard [J...
Tesseract
(28) YouTube
ChatGPT
Trinetra
GitHub
KL University ...

The nginx server is now accessible

Welcome to nginx!

Not secure | 172.23.35.250:30157

Python Java YouTube Maps News Gmail Trinetra | Log in Conceptual model...

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

- **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.

```
PS C:\Windows\system32> minikube dashboard
* Enabling dashboard ...
  - Using image docker.io/kubernetesui/metrics-scraper:v1.0.8
  - Using image docker.io/kubernetesui/dashboard:v2.7.0
* Some dashboard features require the metrics-server addon. To enable all features please run:

        minikube addons enable metrics-server


* Verifying dashboard health ...
* Launching proxy ...
* Verifying proxy health ...
* Opening http://127.0.0.1:52304/api/v1/namespaces/kubernetes-dashboard/services/http:kubernetes-dashboard:/proxy/ in
 your default browser...
```

Kubernetes Dashboard

127.0.0.1:53336/api/v1/namespaces/kubernetes-dashboard/services/http:kubernetes-dashboard:/proxy/#/workloads?namespace=default

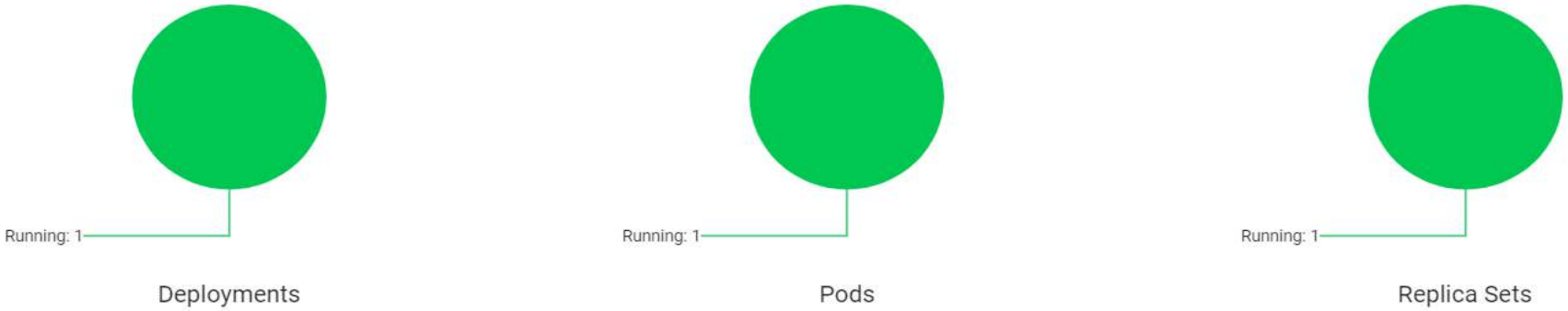
Python | Java | YouTube | Maps | News | Gmail | Trinetra | Log in | Conceptual model...

kubernetes | default | Search

Workloads

- Workloads
  - Cron Jobs
  - Daemon Sets
  - Deployments
  - Jobs
  - Pods
  - Replica Sets
  - Replication Controllers
  - Stateful Sets
- Service
  - Ingresses
  - Ingress Classes
  - Services
- Config and Storage
  - Config Maps
  - Persistent Volume Claims

Workload Status



Deployments

Name	Images	Labels	Pods	Created
mynginx5	nginx	app: mynginx5	1 / 1	17 minutes ago

- **kubectl delete deployment myngnix1**
- this command is telling Kubernetes to delete the deployment named "myngnix1" 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.

```
PS C:\Windows\system32> kubectl delete deployment mynginx1
deployment.apps "mynginx1" deleted
PS C:\Windows\system32> minikube stop
* Stopping node "minikube"  ...
* Powering off "minikube" via SSH ...
* 1 node stopped.
PS C:\Windows\system32>
```