Cloud Computing Journal

Practical No.1

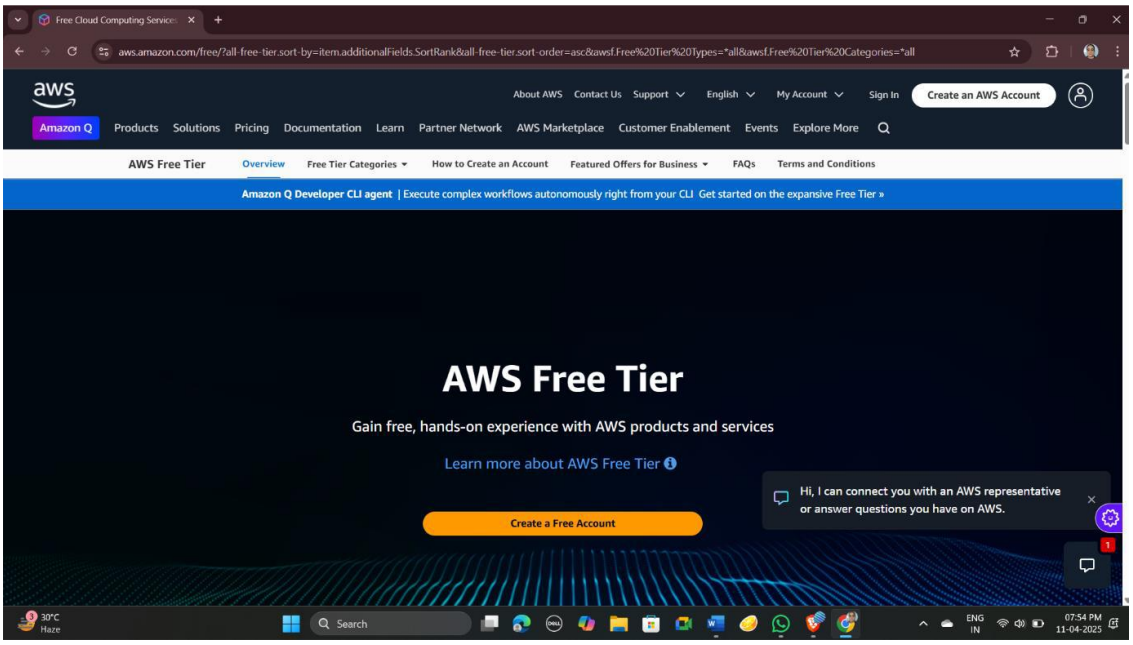
Aim: Deploy a Virtual Machine on VirtualBox or AWS Free Tier

Step:

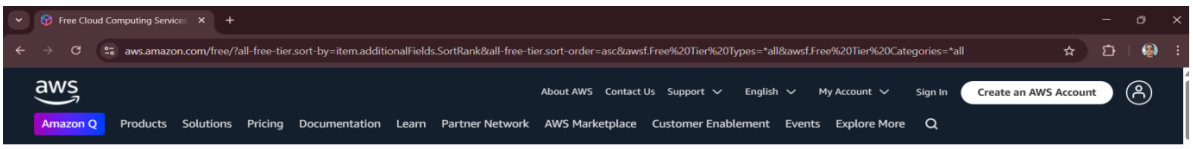
Step-by-Step Guide to Deploy a VM on Azure (Student Login)

Step 01: Sign Up for AWS

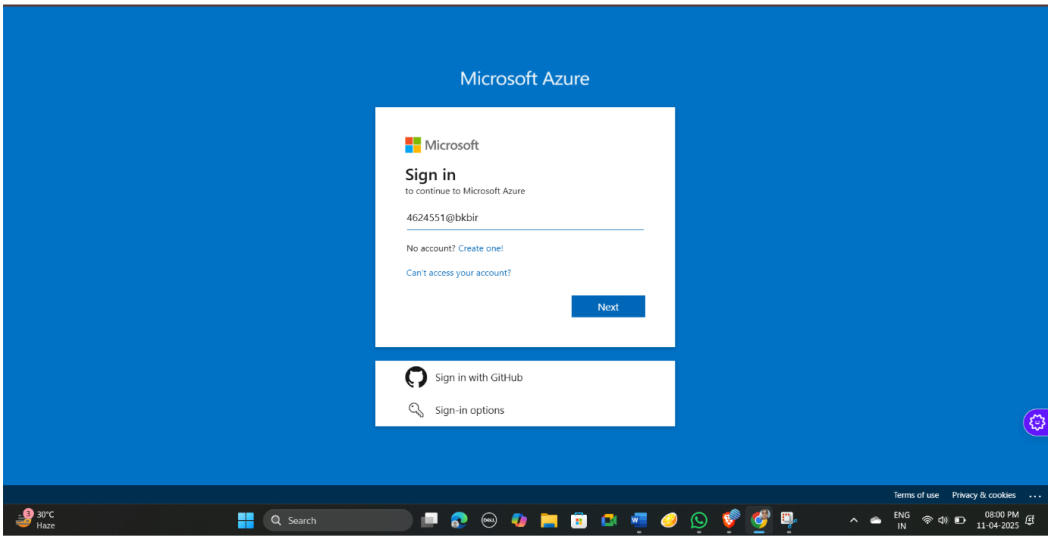
- Go to: **https://azure.microsoft.com/en-us/free/students**



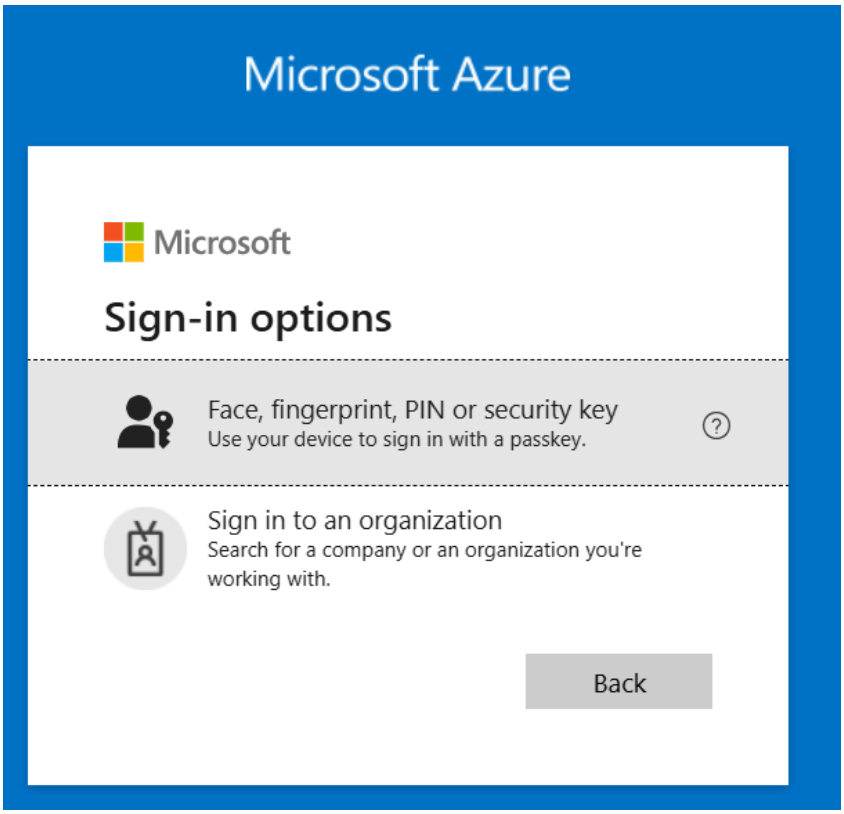
- **Click on Sign-in option.**



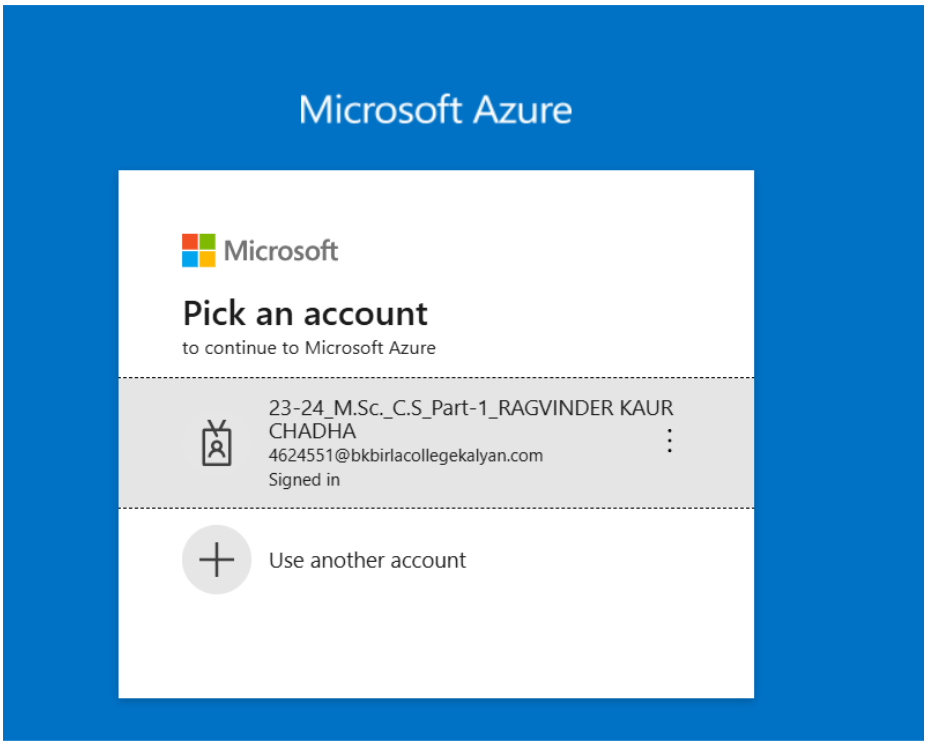
- **After clicking on sign-in this screen will appear. → Go to sign in options**

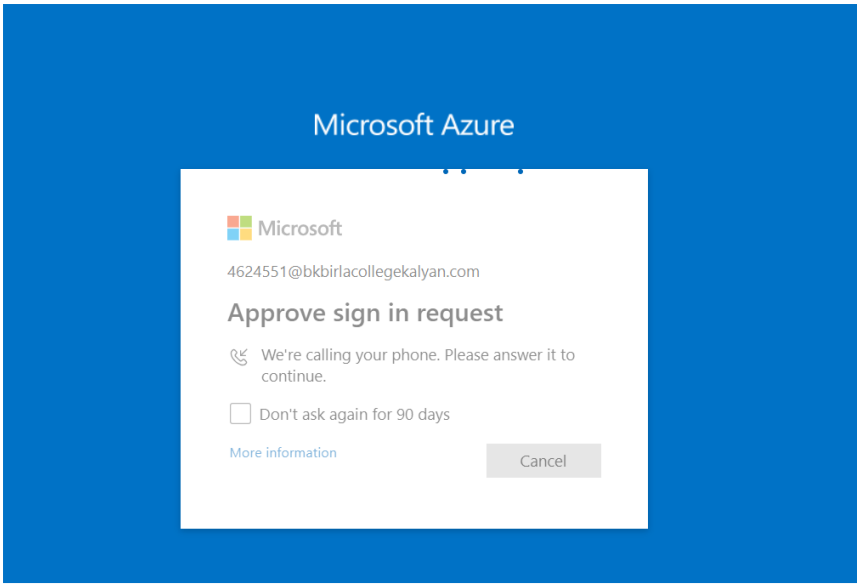


- **After clicking on sign-in options you will get these options → choose sign in to an organization.**

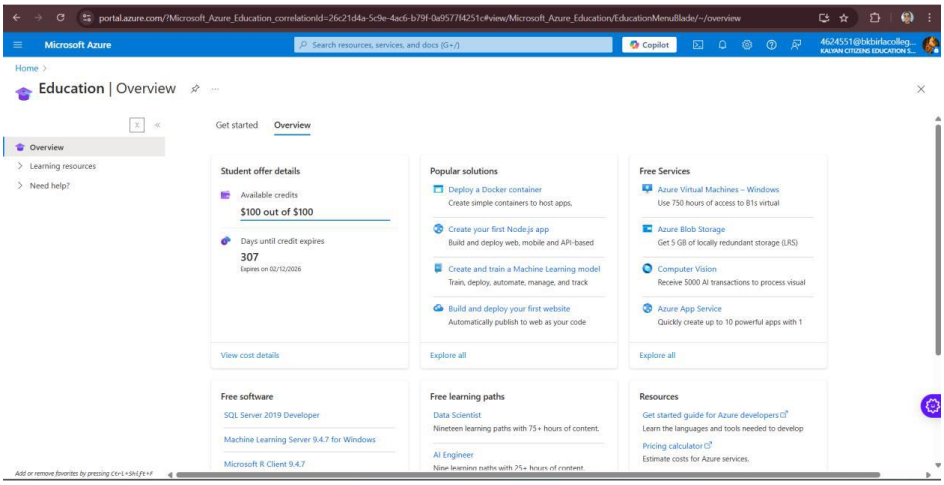


- **After sign-in, you will get this screen → Choose your account**



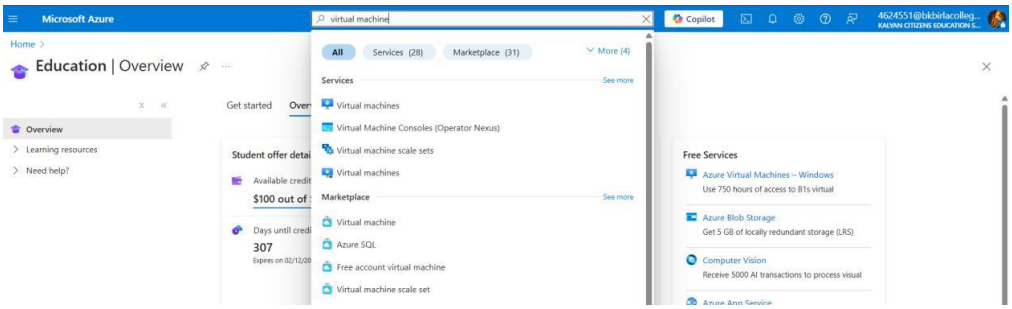


- **Verify its your account by answering the call**
- **After successful sign-in you will be able to see your dashboard**

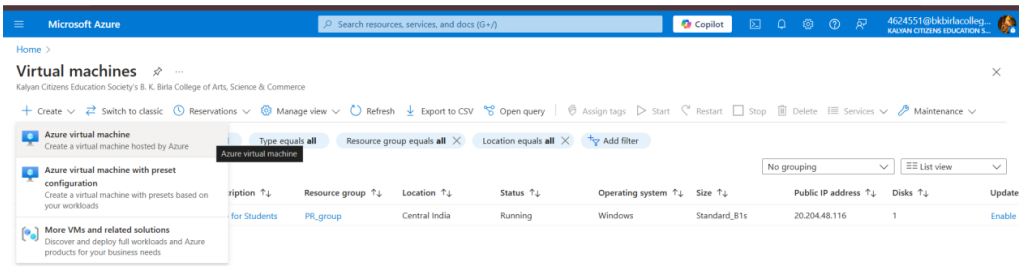


Step 2: Create a Virtual Machine

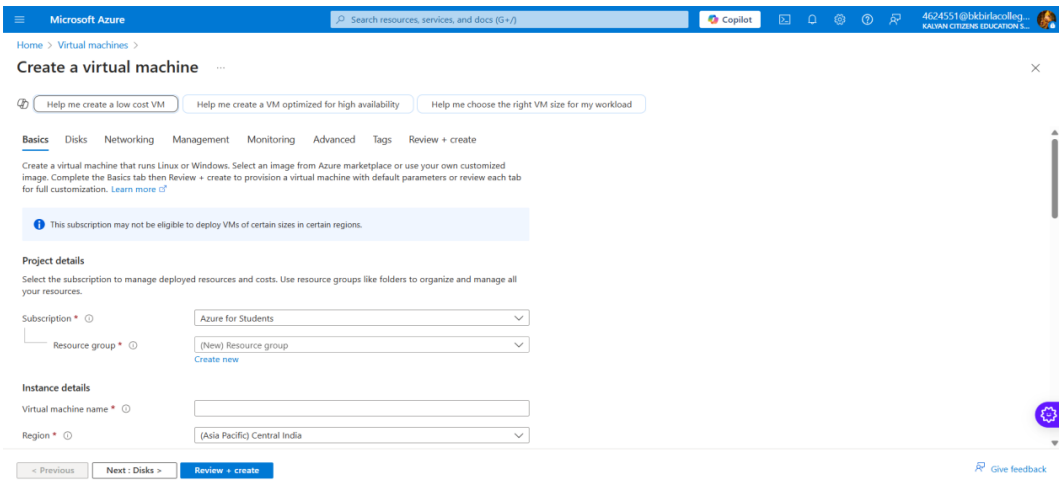
1. **In the search bar, type "Virtual Machines" → Click on it.**



2. **Click + Create → Azure virtual machine**



Step 3: on this screen →Fill in Basic Details



Field	What to Enter
Subscription	Select Azure for Students
Resource Group	Click Create new → Name it (e.g., MyGroup)
Virtual Machine Name	Give a name (e.g., MyTestVM)
Region	Choose nearest (e.g., Central India)
Availability Options	Leave default
Image	Choose OS (e.g., Ubuntu 22.04 or Windows 10)
Size	Select B1s (1 vCPU, 1 GB RAM) – Free tier eligible

Step 4: Configure Admin Account

- Username: Choose a name (e.g., studentuser)
- Authentication Type:
  - For Linux: Choose SSH public key or Password
  - For Windows: Choose Password
- Set your password or upload an SSH key

Step 5: Configure Inbound Ports

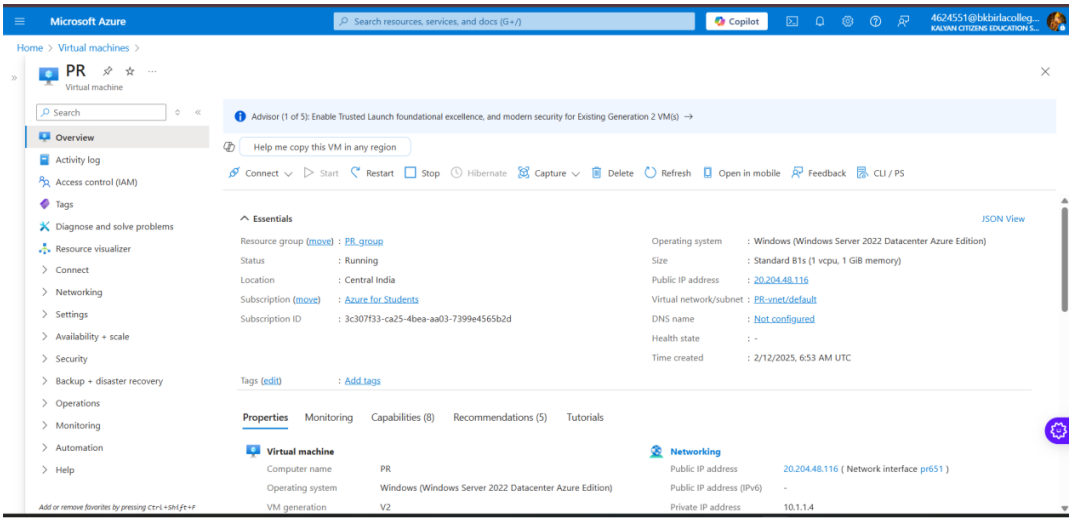
- Open port 22 for Linux (SSH access)
- Open port 3389 for Windows (Remote Desktop)
- You can also allow HTTP (80) and HTTPS (443) if hosting websites

Step 6: Leave Other Settings as Default

- **Disks, Networking, Monitoring — leave default for now**
- **Click Review + Create**
- Azure will validate the settings

Step 7: after successfully Creating the Virtual Machine, yow will be able to see this dashboard.

- **After validation → Click Create**



- Deployment takes a couple of minutes

Step 8: Connect to Your VM

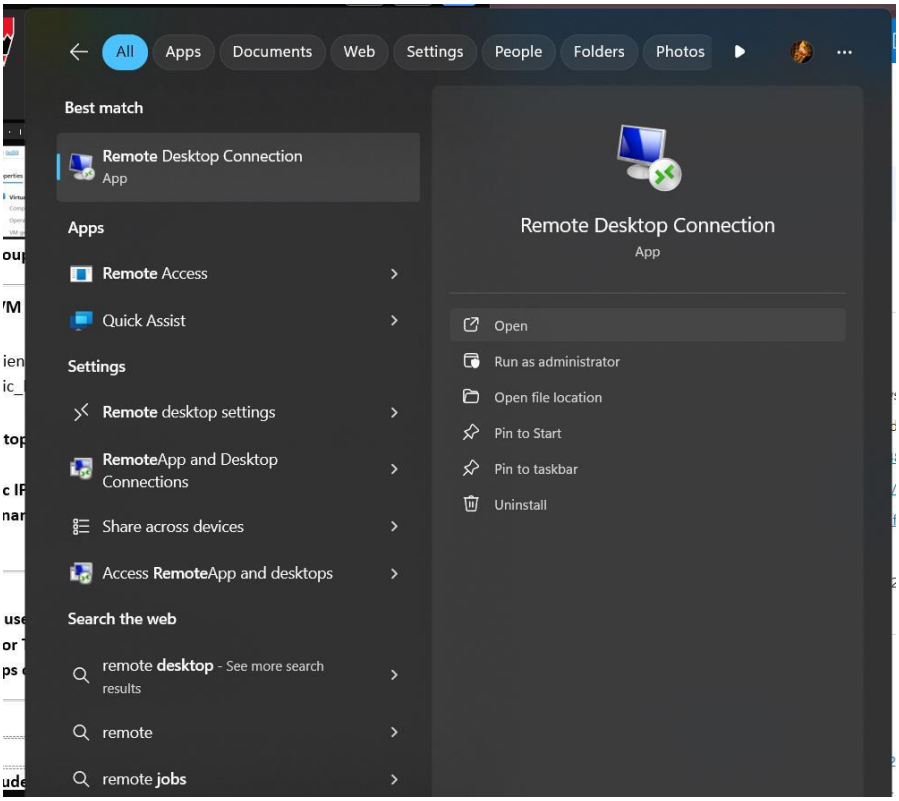
For Linux VM:

- Use terminal or SSH client:

ssh studentuser@<Your_Public_IP>

For Windows VM:

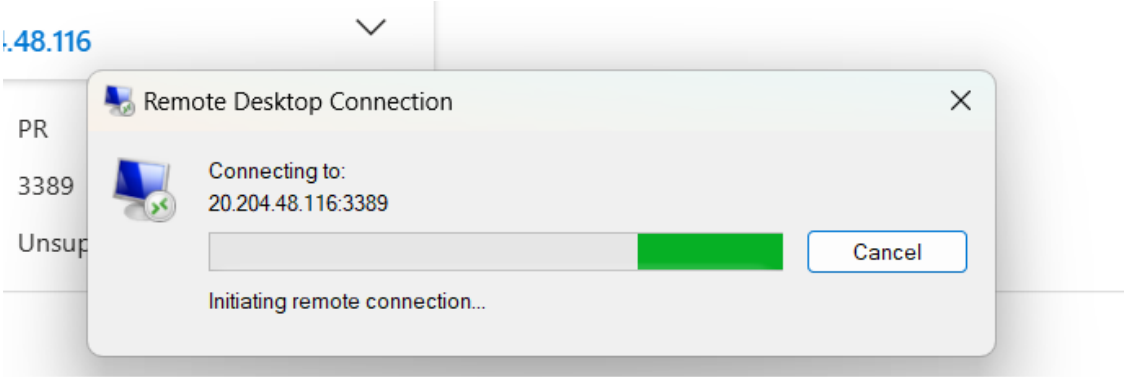
- SEARCH Remote Desktop Connection (RDP) on your PC & enter the following details,
  - Type the public IP address
  - Enter the username and password

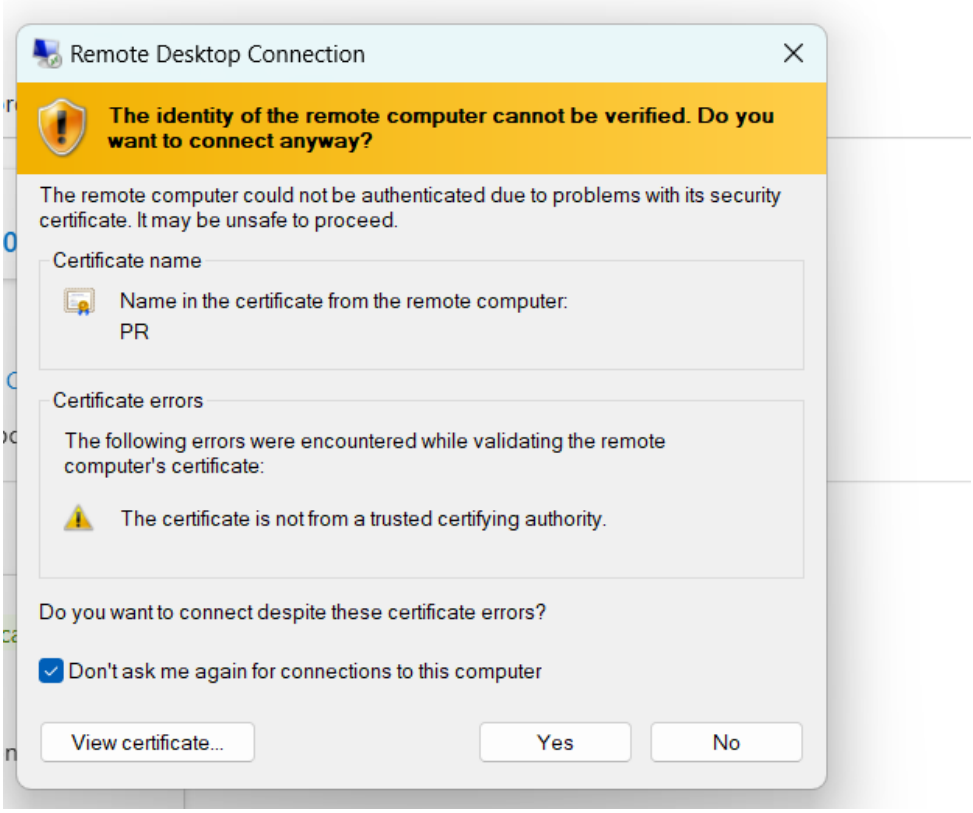


Step 09: After following the above steps properly, you will get the screen to fill the credentials

- Username: ‘VM machine name in my case it is PR’
- Password: ‘which you set while creating VM’

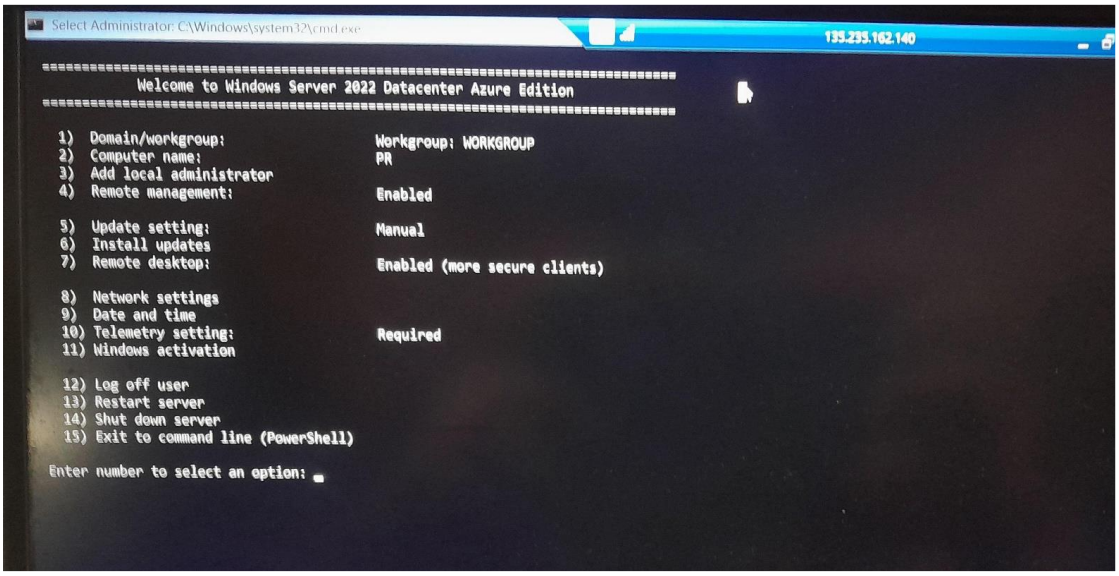
Click on connect





A Virtual Machine screen will display as follows:

OUTPUT



Practical No. 2:

Use Docker Desktop for Containerization

Aim: To install Docker Desktop on Windows and run a containerized application using Docker.

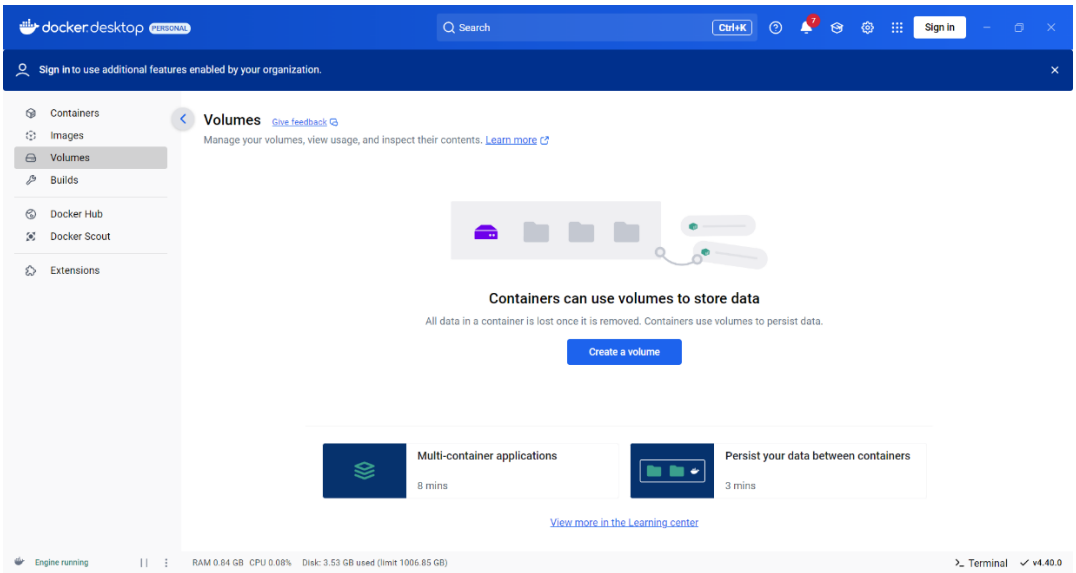
Theory: Docker is an open-source platform designed to automate the deployment, scaling, and management of applications using containerization. A container packages an application and its dependencies together, ensuring that it runs reliably across different environments.
Docker Desktop is a GUI-based tool for managing Docker containers, images, and other Docker components on Windows/macOS systems. It supports a built-in terminal and easy container management.

Steps:
Step 1: Download & Install Docker Desktop

- Visit: https://www.docker.com/products/docker-desktop/
- Download the Windows version.
- Run the installer and complete the setup.
- Restart your system if prompted.

Step 2: Start Docker Desktop

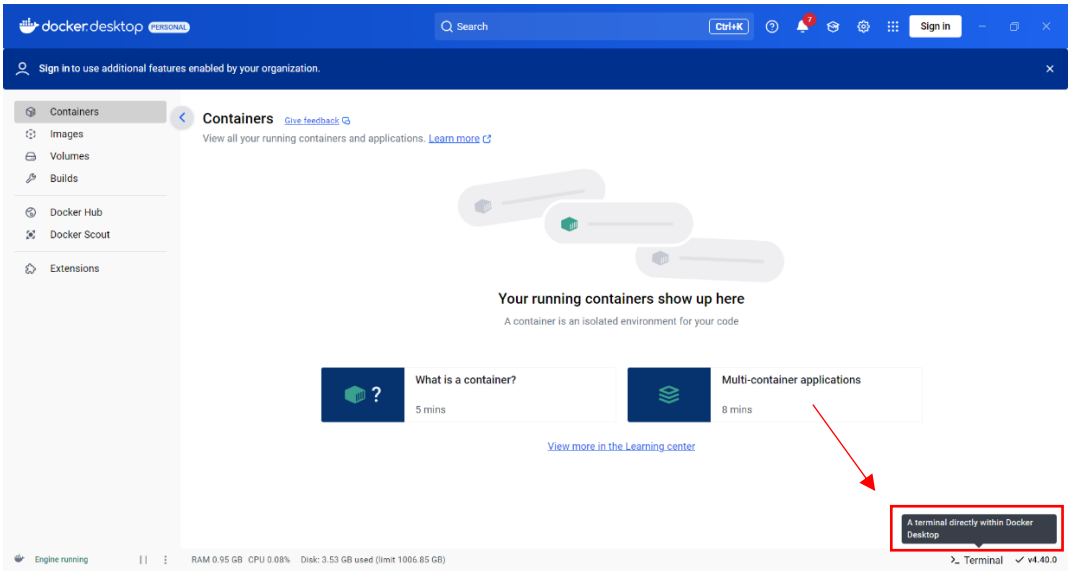
- Launch **Docker Desktop**.



- Wait for the Docker engine to initialize.
- Ensure it shows **"Engine running"** at the bottom.

Step 3: Open Docker Terminal

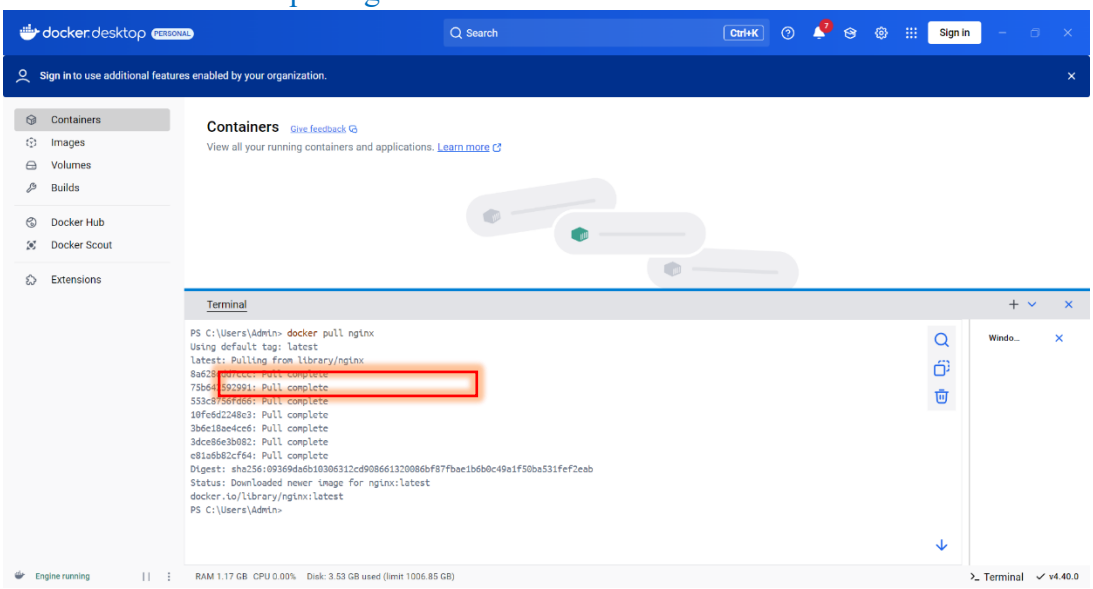
- Click on the **Terminal** tab inside Docker Desktop.



- You can now run Docker commands.

Step 4: Pull the Nginx Image

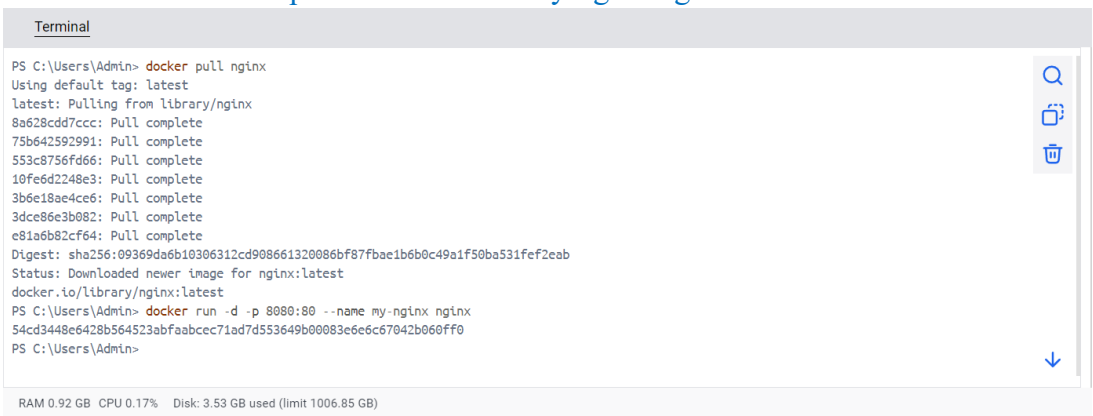
- Type the following command to download the Nginx container image from Docker Hub:
  CMD: docker pull nginx



Step 5: Run the Nginx Container

- Type this command to start a container in detached mode:

CMD: docker run -d -p 8080:80 --name my-nginx nginx

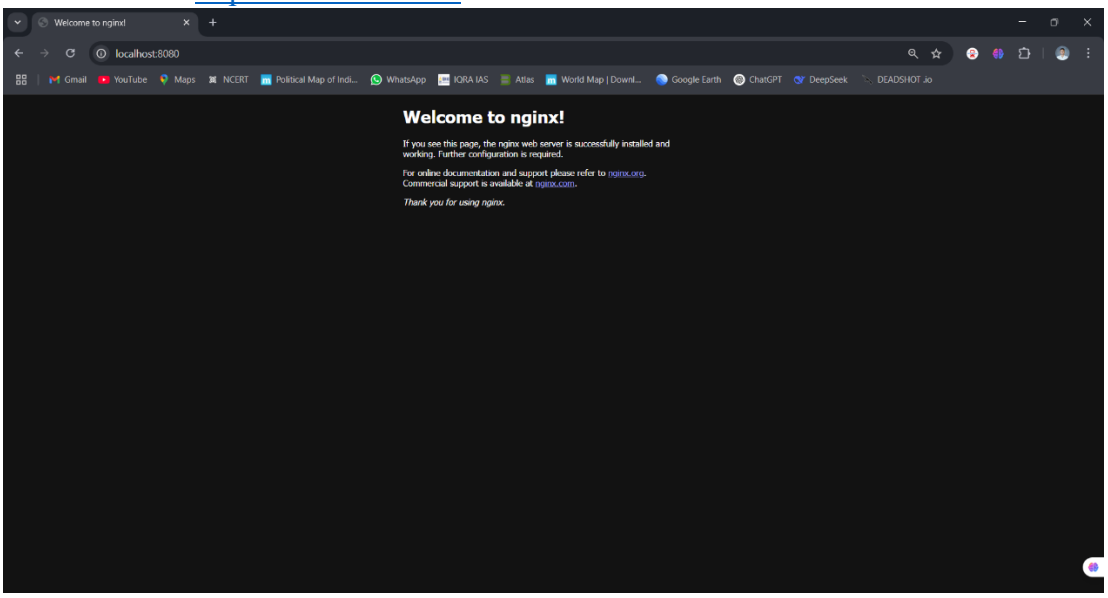


This maps **localhost:8080** on your machine to **port 80** inside the container.

Step 6: Verify the Container is Running

- Visit your browser and go to:

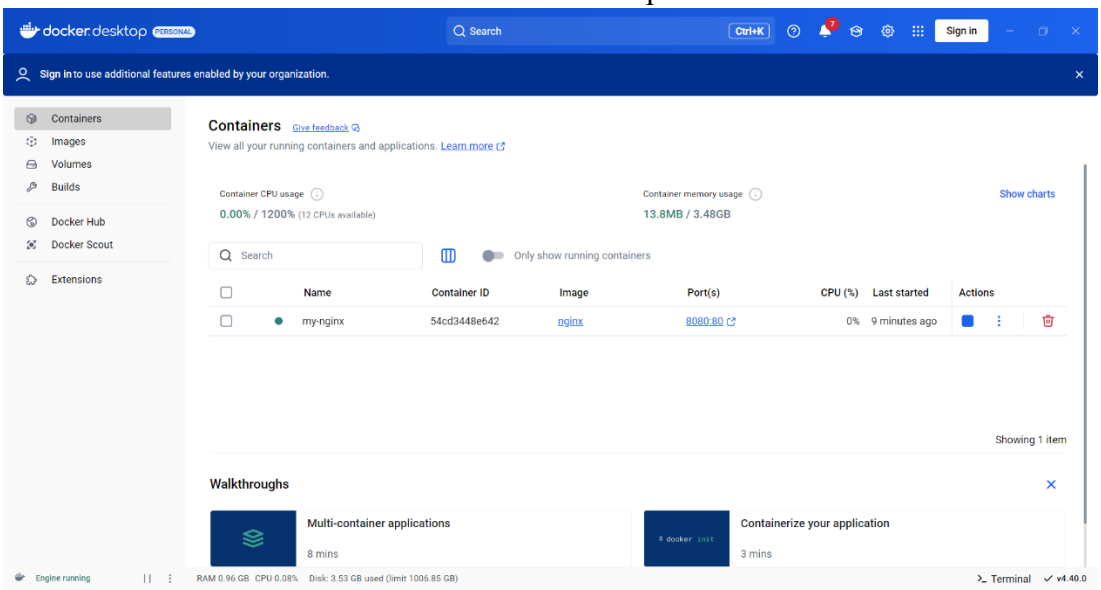
BROWSER: http://localhost:8080



You should see the default **Nginx Welcome Page**.

Step 7: See Container in Docker Desktop GUI

- Go to **"Containers"** tab in Docker Desktop



- You'll see my-nginx listed as **running**
- Click on it to:
  - View logs
  - Access terminal
  - Restart or stop container

Step 8: When Done (Stop & Remove Container)

➔ **To stop:**

CMD: docker stop my-nginx

➔ **To remove:**

CMD: docker rm my-nginx

Practical No.3

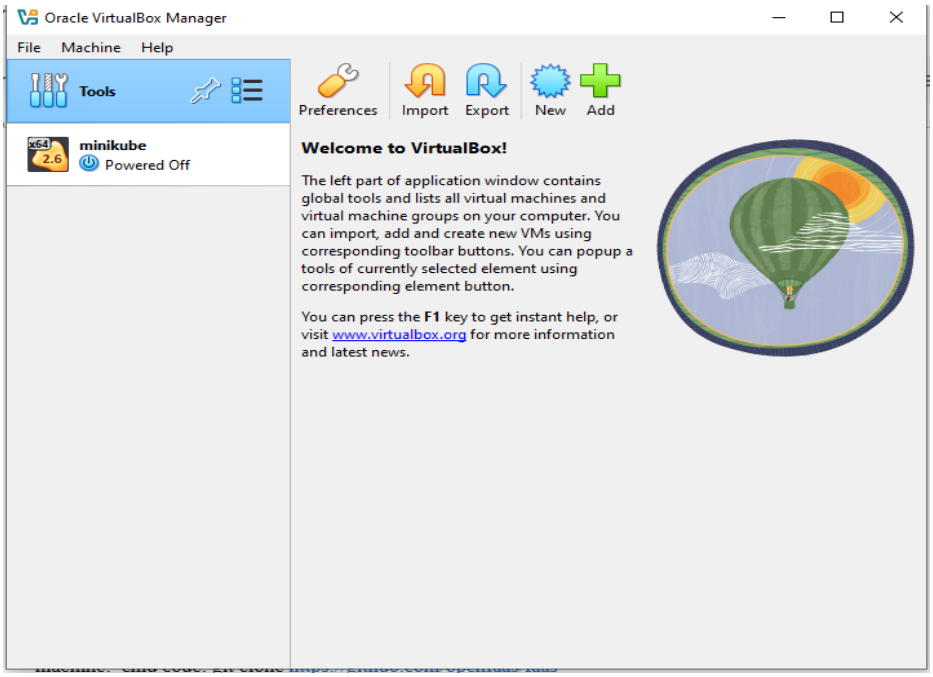
Aim: Set up a kubernates cluster on minikube

Steps:

Step 1: Install VirtualBox or Docker

If you haven't already installed VirtualBox, download and install it from the official website

Once installed, make sure VirtualBox is running and accessible on your system.



Step 2: Install Kubectl

Kubectl is used to interact with the Kubernetes cluster.

On Windows/macOS/Linux: Go to the official page:
https://kubernetes.io/docs/tasks/tools/

Step 3: Install Minikube

Minikube runs a local Kubernetes cluster.

Download from:
https://minikube.sigs.k8s.io/docs/start/

Step 4 : Start Minikube

Open terminal or command prompt.

Run this command to start Minikube:

minikube start

```
Command Prompt
Microsoft Windows [Version 10.0.19045.3803]
(c) Microsoft Corporation. All rights reserved.

C:\Users\admin>minikube start
* minikube v1.35.0 on Microsoft Windows 10 Pro 10.0.19045.3803 Build 19045.3803
* Automatically selected the virtualbox driver
* Downloading VM boot image ...
    > minikube-v1.35.0-amd64.iso....:  65 B / 65 B [---------] 100.00% ? p/s 0s
    > minikube-v1.35.0-amd64.iso:  345.38 MiB / 345.38 MiB  100.00% 10.40 MiB p
* Starting "minikube" primary control-plane node in "minikube" cluster
* Downloading Kubernetes v1.32.0 preload ...
    > preloaded-images-k8s-v18-v1...:  333.57 MiB / 333.57 MiB  100.00% 10.51 M
* Creating virtualbox VM (CPUs=2, Memory=2200MB, Disk=20000MB) ...
! Failing to connect to https://registry.k8s.io/ from inside the minikube VM
* To pull new external images, you may need to configure a proxy: https://minikube.sigs.k8s.io/docs/reference/networking
/proxy/
* Preparing Kubernetes v1.32.0 on Docker 27.4.0 ...
  - Generating certificates and keys ...
  - Booting up control plane ...
  - Configuring RBAC rules ...
* Configuring bridge CNI (Container Networking Interface) ...
* 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

C:\Users\admin>
```

Check if Minikube and Kubernetes are running:

minikube status

```
C:\Users\admin>minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured
```

Check Kubernetes node:

kubectl get nodes

```
C:\Users\admin>kubectl get nodes
NAME       STATUS   ROLES           AGE     VERSION
minikube   Ready    control-plane   6m40s   v1.32.0

C:\Users\admin>
```

Step 5: Deploy an Application

Command:

kubectl create deployment nginx-deployment --image=nginx

kubectl get pods

```
C:\Users\admin>kubectl get pods
NAME                               READY   STATUS              RESTARTS   AGE
nginx-deployment-6cfb98644c-t94d8  0/1     ContainerCreating   0          4s

C:\Users\admin>
```

Step 5: Expose the Application

Command:

```
C:\Users\admin>kubectl expose deployment nginx-deployment --type=NodePort --port=80
service/nginx-deployment exposed

C:\Users\admin>kubectl get services
NAME              TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
kubernetes        ClusterIP   10.96.0.1       <none>        443/TCP        12m
nginx-deployment  NodePort    10.111.125.84   <none>        80:31564/TCP   1s

C:\Users\admin>
```

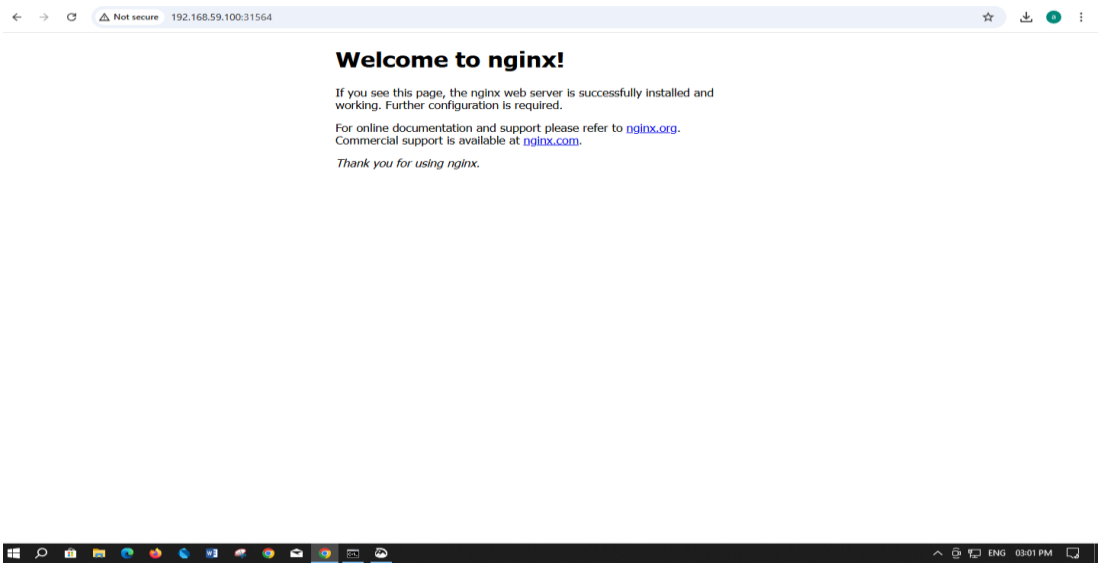
Step 6: Access the Application

Command:

minikube service nginx-deployment --url

```
C:\Users\admin>minikube service nginx-deployment --url
http://192.168.59.100:31564

C:\Users\admin>_
```



Step 7: Stop or Delete the Cluster

Commands:

minikube stop

minikube delete

```
C:\Users\admin>minikube stop
* Stopping node "minikube"  ...
* 1 node stopped.

C:\Users\admin>minikube delete
* Deleting "minikube" in virtualbox ...
* Removed all traces of the "minikube" cluster.

C:\Users\admin>
```

Practical No.04

Aim: To deploy a Serverless function using Openfaas

Theory:

OpenFaaS (Function as a Service) is a framework that allows users to deploy and manage serverless functions. It simplifies the process of deploying microservices by providing an easy-to-use interface, scalability, and a simple mechanism for function deployment.

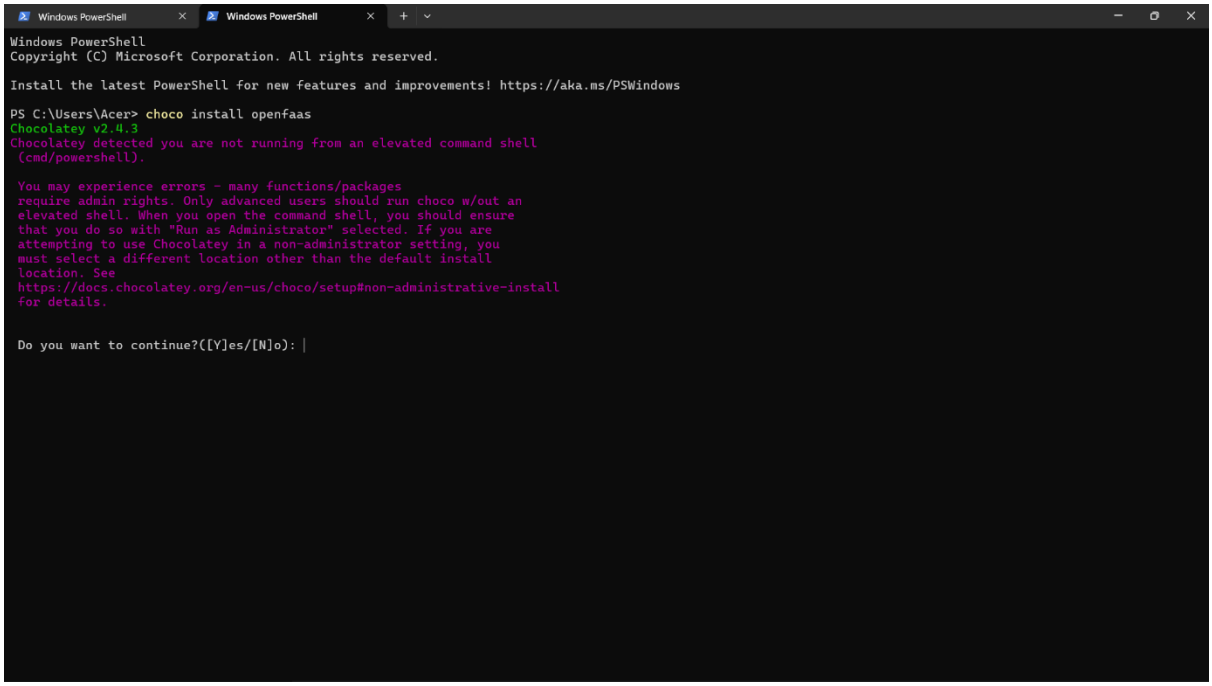
Docker is a platform that automates the deployment of applications in lightweight, portable containers. These containers encapsulate an application and all of its dependencies, making it easier to run applications consistently across different environments.

Steps:

Step 1: Install Docker

If you haven't already installed Docker, download and install it from the official website

Once installed, make sure Docker is running and accessible on your system.



Step 2: Install the OpenFaaS CLI

Download and Install faas-cli: OpenFaaS CLI is used to interact with OpenFaaS

For **Windows**, you can download the binary directly from the GitHub release page

Step 3: Run OpenFaaS Using Docker Compose

OpenFaaS provides an easy way to deploy using Docker Compose. Docker Compose allows you to define and manage multi-container Docker applications.

Clone OpenFaaS GitHub Repository: Download the OpenFaaS repository to your local machine: cmd code: git clone https://github.com/openfaas/faas

cd faas

Step 4 : Run Docker Compose: OpenFaaS provides a docker-compose.yml file in the repository. This file defines the services you need to run.

Start OpenFaaS using Docker Compose by running the following:

Cmd : docker-compose up –d

```
Windows PowerShell   Windows PowerShell
PS C:\Users\Acer\my-function> docker ps
CONTAINER ID   IMAGE                                  COMMAND                  CREATED          STATUS          PORTS
                                                                                                 NAMES
425f39b87172   gcr.io/k8s-minikube/kicbase:v0.0.46   "/usr/local/bin/entr…"   47 minutes ago   Up 47 minutes   127.0.0.1:59041->22/tcp, 127.0.0.1:59042->237
6/tcp, 127.0.0.1:59044->5000/tcp, 127.0.0.1:59045->8443/tcp, 127.0.0.1:59043->32443/tcp   minikube
PS C:\Users\Acer\my-function> docker ps -a
CONTAINER ID   IMAGE                                  COMMAND                  CREATED          STATUS          PORTS
                                                                                                 NAMES
425f39b87172   gcr.io/k8s-minikube/kicbase:v0.0.46   "/usr/local/bin/entr…"   48 minutes ago   Up 48 minutes   127.0.0.1:59041->22/tcp, 127.0.0.1:59042->237
6/tcp, 127.0.0.1:59044->5000/tcp, 127.0.0.1:59045->8443/tcp, 127.0.0.1:59043->32443/tcp   minikube
PS C:\Users\Acer\my-function> minikube start
* minikube v1.35.0 on Microsoft Windows 11 Home Single Language 10.0.26100.3624 Build 26100.3624
* Using the docker driver based on existing profile
* Starting "minikube" primary control-plane node in "minikube" cluster
* Pulling base image v0.0.46 ...
* Updating the running docker "minikube" container ...
! Failing to connect to https://registry.k8s.io/ from inside the minikube container
* To pull new external images, you may need to configure a proxy: https://minikube.sigs.k8s.io/docs/reference/networking/proxy/
* Preparing Kubernetes v1.32.0 on Docker 27.4.1 ...
* 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:\Users\Acer\my-function> minikube ip
192.168.49.2
PS C:\Users\Acer\my-function> kubectl get pods -n openfaas
NAME                                READY   STATUS    RESTARTS      AGE
alertmanager-5865857cbd-m6x77       1/1     Running   1 (78s ago)   48m
gateway-7fbbc5785b-6zpth            2/2     Running   4 (37s ago)   48m
nats-6ddf479847-x8w6s               1/1     Running   1 (78s ago)   48m
prometheus-6c8f8c4c66-nh9tk         1/1     Running   1 (78s ago)   48m
queue-worker-7d4599bf65-dkgr4       1/1     Running   3 (78s ago)   48m
PS C:\Users\Acer\my-function> ^C
PS C:\Users\Acer\my-function> curl http://192.168.49.2:8080/system/functions
curl : Unable to connect to the remote server
At line:1 char:1
+ curl http://192.168.49.2:8080/system/functions
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : InvalidOperation: (System.Net.HttpWebRequest:HttpWebRequest) [Invoke-WebRequest], WebExc
   eption
    + FullyQualifiedErrorId : WebCmdletWebResponseException,Microsoft.PowerShell.Commands.InvokeWebRequestCommand

PS C:\Users\Acer\my-function> kubectl get svc -n openfaas
NAME                TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
```

Step 5: Access OpenFaaS UI

using Minikube, use the external IP

Powrshell : http://< http://127.0.0.1/>:<59041>

Login to OpenFaaS: The default credentials are:

- Username: admin
- Password: admin

Step 6: Deploy Functions Using faas-cli

Create a New Function: Once the OpenFaaS Gateway is up, you can use the faas-cli to create and deploy functions.

```
Windows PowerShell    Windows PowerShell
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Acer> faas-cli version
  ___                   _____       ____
 / _ \ _ __  ___ _ __ |_   _|_ _  __ _/ ___|
| | | | '_ \ / _ \ '_ \  | |/ _` |/ _` \___ \
| |_| | |_) |  __/ | | |  | | (_| | (_| |___) |
 \___/| .__/ \___|_| |_|_|  \__,_|\__,_|____/
      |_|


CLI:
 commit:  c024f164bc047e0f690099a7a76abae1958114c7
 version: 0.17.3
PS C:\Users\Acer> faas-cli new --lang python3 my-function
Template: "python3" was not found in the templates folder or in the store
PS C:\Users\Acer> faas-cli template pull
Fetch templates from repository: https://github.com/openfaas/templates.git
Wrote 8 template(s) : [dockerfile java11 java11-vert-x java17 node18 node20 php7 php8] from https://github.com/openfaas/templates.git
PS C:\Users\Acer> faas-cli new --lang python3 my-function
Template: "python3" was not found in the templates folder or in the store
PS C:\Users\Acer> faas-cli template list
Allows browsing templates from store or pulling custom templates

Usage:
  faas-cli template [command]

Examples:
  faas-cli template pull https://github.com/custom/template
  faas-cli template store list
  faas-cli template store ls
  faas-cli template store pull ruby-http
  faas-cli template store pull openfaas-incubator/ruby-http

Available Commands:
  pull        Downloads templates from the specified git repo
  store       Command for pulling and listing templates from store

Flags:
  -h, --help   help for template

Global Flags:
      --filter string   Wildcard to match with function names in YAML file
      --regex string    Regex to match with function names in YAML file
  -f, --yaml string     Path to YAML file describing function(s)
```

Example to create a Python function:

powershell: faas-cli new --lang python3 my-python-function

Build the Function: Navigate to your function directory and build the function:

Powershell: faas-cli build -f my-python-function.yml

Deploy the Function: After building, deploy the function to OpenFaaS

Powershell: faas-cli deploy -f my-python-function.yml

Invoke Function by using:

Powershell: faas-cli invoke my-python-function

```
Windows PowerShell    Windows PowerShell
dotnet8-csharp           [x]       openfaas           C# template using WebApplication
golang-middleware        [x]       openfaas           HTTP middleware interface in Go
java11-vert-x            [x]       openfaas           Java 11 Vert.x template
node20                   [x]       openfaas           HTTP-based Node 20 template
php8                     [x]       openfaas           Classic PHP 8 template
python3-http             [x]       openfaas           Python 3 with Flask and HTTP
python3-http-debian      [x]       openfaas           Python 3 with Flask and HTTP based on Debian
ruby-http                [x]       openfaas           Ruby 3.3.6 HTTP template
cobol                    [ ]       devries            COBOL Template
perl-alpine              [ ]       tmiklas            Perl language template based on Alpine image
powershell-http-template [ ]       openfaas-incubator Powershell Core HTTP Ubuntu:16.04 template
powershell-template      [ ]       openfaas-incubator Powershell Core Ubuntu:16.04 template
quarkus-native           [ ]       pmlopes            Quarkus.io native image template
rust                     [ ]       openfaas-incubator Community Rust template
rust-http                [ ]       openfaas-incubator Community Rust template with HTTP bindings
bun-express              [ ]       openfaas           HTTP-based template using bun
golang-http              [ ]       openfaas           Request/response style HTTP template
java11                   [ ]       openfaas           Java 11 template
java17                   [ ]       openfaas           Java 17 template
node18                   [ ]       openfaas           HTTP-based Node 18 template
php7                     [ ]       openfaas           Classic PHP 7 template
puppeteer-nodelts        [ ]       alexellis          A puppeteer template for headless Chrome
python3-flask            [ ]       openfaas           Python 3 Flask template
python3-flask-debian     [ ]       openfaas           Python 3 Flask template based on Debian

PS C:\Users\Acer> faas-cli template store pull python3
Template with name: `python3` does not exist in the repo
PS C:\Users\Acer> faas-cli template store pull python3-flask
Fetch templates from repository: https://github.com/openfaas/python-flask-template
Wrote 5 template(s) : [python27-flask python3-flask python3-flask-debian python3-http python3-http-debian] from https://github.com/openfaas/python-flask-tem
plate
PS C:\Users\Acer> faas-cli new --lang python3 my-function
Template: "python3" was not found in the templates folder or in the store
PS C:\Users\Acer> faas-cli new --lang python3-flask my-function
Folder: my-function created.
  ___                   _____       ____
 / _ \ _ __  ___ _ __ |_   _|_ _  __ _/ ___|
| | | | '_ \ / _ \ '_ \  | |/ _` |/ _` \___ \
| |_| | |_) |  __/ | | |  | | (_| | (_| |___) |
 \___/| .__/ \___|_| |_|_|  \__,_|\__,_|____/
      |_|


Function created in folder: my-function
```

Practical No.5

Aim: To implement Infrastructure as Code using **Terraform** on a **Windows** machine, enabling automated and repeatable creation of infrastructure (in this case, a local file) without manual intervention.

Required Tools

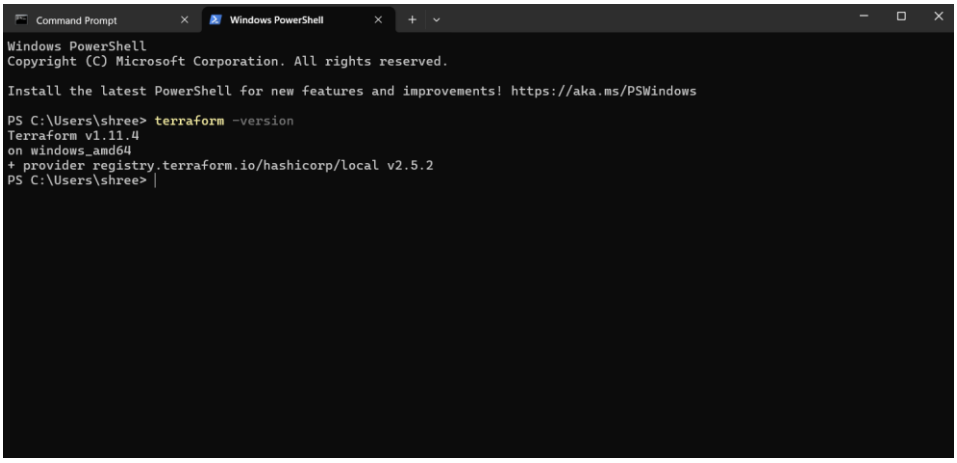
Tool	Purpose	Download
Terraform	Main IaC tool	https://www.terraform.io/downloads
Text Editor (e.g., Notepad/VS Code)	To write Terraform config files	VS Code *(optional)*
PowerShell or Command Prompt	To run commands	Built-in

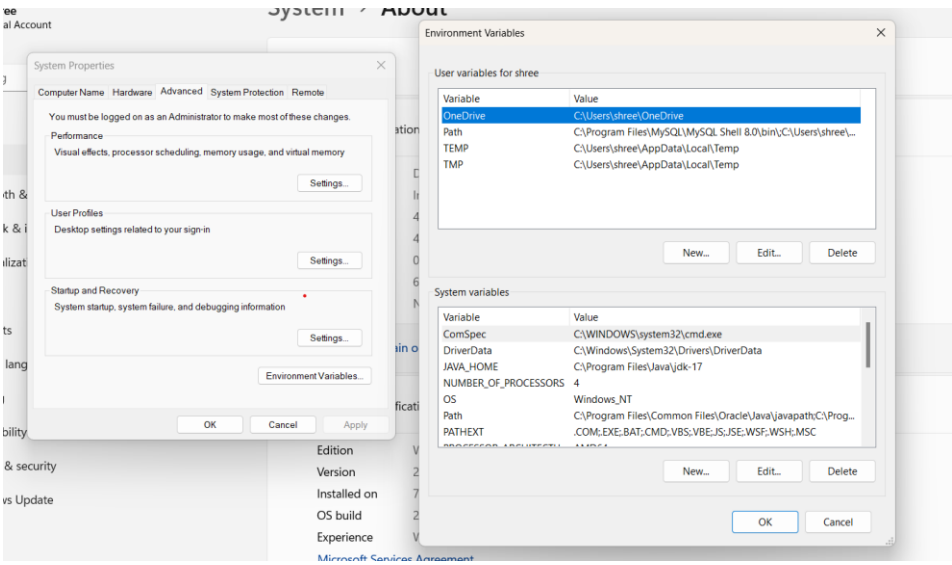
Procedure

✅ Step 1: Install Terraform on Windows

Manual Install:

1. Download Terraform zip for Windows from Terraform Downloads
2. Extract the zip to a folder (e.g., C:\terraform)
3. Add the folder to your system PATH:
    - Search for **Environment Variables**
    - Under **System Variables**, find Path → click **Edit**





Create a Terraform Configuration File:

Step 4: Initialize Terraform:

terraform init

```
Command Prompt    Windows PowerShell

C:\Users\shree>terraform init
Terraform initialized in an empty directory!

The directory has no Terraform configuration files. You may begin working
with Terraform immediately by creating Terraform configuration files.

C:\Users\shree>cd C:\Projects\terraform-demo
The system cannot find the path specified.

C:\Users\shree>dir
 Volume in drive C has no label.
 Volume Serial Number is 6490-020F

 Directory of C:\Users\shree

04/09/2025  07:18 PM    <DIR>          .
07/22/2023  05:47 PM    <DIR>          ..
05/04/2023  01:33 PM    <DIR>          .cache
11/14/2023  06:59 PM    <DIR>          .conda
05/04/2023  10:41 AM                25 .condarc
05/04/2023  10:41 AM    <DIR>          .continuum
05/04/2023  01:33 PM    <DIR>          .eclipse
05/24/2023  06:21 PM               164 .gitconfig
10/09/2024  12:22 PM    <DIR>          .idlerc
04/09/2025  01:15 PM    <DIR>          .ipynb_checkpoints
05/04/2023  11:11 AM    <DIR>          .ipython
05/04/2023  11:11 AM    <DIR>          .jupyter
05/06/2023  08:30 AM    <DIR>          .matplotlib
12/06/2023  08:43 PM    <DIR>          .ms-ad
05/28/2024  04:04 PM    <DIR>          .p2
```

Step 5: Plan the Infrastructure

terraform plan:

```
Command Prompt    Windows PowerShell

C:\Users\shree>terraform plan

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the
following symbols:
  + create

Terraform will perform the following actions:

  # local_file.hello will be created
  + resource "local_file" "hello" {
      + content              = "Hello, Terraform from Windows!"
      + content_base64sha256 = (known after apply)
      + content_base64sha512 = (known after apply)
      + content_md5          = (known after apply)
      + content_sha1         = (known after apply)
      + content_sha256       = (known after apply)
      + content_sha512       = (known after apply)
      + directory_permission = "0777"
      + file_permission      = "0777"
      + filename             = "./hello.txt"
      + id                   = (known after apply)
    }

Plan: 1 to add, 0 to change, 0 to destroy.
____________________________________________________________________________________________________

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if
you run "terraform apply" now.
```

Step 6: Apply the Configuration

terraform apply:

```
Command Prompt    Windows PowerShell

  # local_file.hello will be created
  + resource "local_file" "hello" {
      + content              = "Hello, Terraform from Windows!"
      + content_base64sha256 = (known after apply)
      + content_base64sha512 = (known after apply)
      + content_md5          = (known after apply)
      + content_sha1         = (known after apply)
      + content_sha256       = (known after apply)
      + content_sha512       = (known after apply)
      + directory_permission = "0777"
      + file_permission      = "0777"
      + filename             = "./hello.txt"
      + id                   = (known after apply)
    }

Plan: 1 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

local_file.hello: Creating...
local_file.hello: Creation complete after 0s [id=e8b749e22faba22ad94af2c16d9f213b425f8d22]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.

C:\Users\shree>
```