

Name: Kriti

College: Lovely Professional University

Week2 Assignment

Email: kritijulia@gmail.com

## **Q1 Deploy Linux and Windows virtual machine and access them using SSH and RDP**

**Answers:** I created a new account on azure portal then click on create resources then created virtual machine using free versions then created and deployed linux virtual machine using ssh. For windows operating system used RDP. (Create a resource → Compute → Virtual machine )

### **Configure Networking (Open Ports)**

**In the "Networking" tab:**

- Select **Virtual Network/Subnet** (default is okay)
- Enable **Public IP** (required for SSH/RDP access)
- Under **Inbound port rules**:
  - **Linux VM**:
    - Allow **SSH (TCP 22)**
  - **Windows VM**:
    - Allow **RDP (TCP 3389)**

### **Connect to VMs**

#### **Linux via SSH**

After VM deployment:

1. Copy the public IP of your Linux VM.
2. Run the following command from your terminal:

```
ssh -i ~/Downloads/ubuntu_key.pem azureuser@<Public-IP>
```

Replace <Public-IP> with my VM IP.

#### **Windows via RDP**

After VM deployment:

1. Go to Windows VM in the Azure portal.
2. Click on Connect → RDP → Download RDP File.
3. Open the downloaded .rdp file in Remote Desktop Connection.
4. Enter:
  - Username: azureuser
  - Password: the one you set

Home >

## CreateVm-canonical.ubuntu-24\_04-lts-server-20250615172702 | Overview

Deployment

Search X <> Delete Cancel Redeploy Download Refresh

Overview Inputs Outputs Template

Your deployment is complete

Deployment name: CreateVm-canonical.ubuntu-24\_04-lts-server-20250615172702 Start time: 6/15/2025, 5:43:20 PM  
Subscription: Azure for Students Correlation ID: 5c1e25d6-6fcc-4dc

Resource group: azurelinux\_group

Deployment details

Next steps

Setup auto-shutdown Recommended  
Monitor VM health, performance and network dependencies Recommended  
Run a script inside the virtual machine Recommended

Go to resource Create another VM

Give feedback

Tell us about your experience with deployment

Microsoft Azure

Search resources, services, and docs (G+/)

Copilot

krithi.12217521@plin.LOVELY PROFESSIONAL UNIVERS...

Home > Create a resource >

Create a virtual machine

Validation passed

Help me create a low cost VM Help me create a VM optimized for high availability Help me choose the right VM size for my workload

Basics

Subscription: Azure for Students  
Resource group: azurelinux\_group  
Virtual machine name: ubuntu  
Region: East Asia  
Availability options: Self-select  
Zone options: 1  
Availability zone: Trusted location: Yes  
Security type: Enable secure boot: Yes  
Enable vTPM: No  
Integrity monitoring: Ubuntu Server: x64  
Image: Standard B1s (1 vcpu, 1 GiB memory)  
VM architecture: No  
Enable Hibernation: SSH public key: azuser  
Authentication type: Username: azuser

Generate new key pair

An SSH key pair contains both a public key and a private key. Azure doesn't store the private key. After the SSH key resource is created, you won't be able to download the private key again. [Learn more](#)

Download private key and create resource

Return to create a virtual machine

Download a template for automation Give feedback

Copilot

Submitting deployment...  
Submitting the deployment template for resource group 'azurelinux\_group'.  
Help me create a low cost VM

Copilot

I can help guide you through three options to lower the cost of your virtual machine.

1. Azure Spot
2. Low Cost Size
3. Auto-shutdown

Copilot

Option 1: Use Azure Spot Virtual Machines to take advantage of our unused capacity at a significant cost savings. At any point in time when Azure needs the capacity back, the Azure infrastructure will evict Azure Spot Virtual Machines.

Enable Skip Cancel

I want to ... 0 / 500

Created and deployed windows virtual machine using RDP

## Create a virtual machine ...

 Validation passed



Help me create a low cost VM

Help me create a VM optimized for high availability

Resource group	(new) azure_window
Virtual machine name	window
Region	East Asia
Availability options	Availability zone
Zone options	Self-selected zone
Availability zone	1
Security type	Trusted launch virtual machines
Enable secure boot	Yes
Enable vTPM	Yes
Integrity monitoring	No
Image	Windows Server 2025 Datacenter: Azure Edition - Gen2
VM architecture	x64
Size	Standard B1s (1 vcpu, 1 GiB memory)
Enable Hibernation	No
Username	azureuser
Public inbound ports	RDP
Already have a Windows license?	No
Azure Spot	No

## ✓ Your deployment is complete

Deployment name: CreateVm-MicrosoftWindowsServer.WindowsSe... Start time: 6/15/2025, 5:49:57  
Subscription: Azure for Students Correlation ID: 86f06f6d-34ee-4...  
Resource group: azure\_window

### ✓ Deployment details

### ✗ Next steps

[Setup auto-shutdown](#) Recommended

[Monitor VM health, performance and network dependencies](#) Recommended

[Run a script inside the virtual machine](#) Recommended

Home > Compute infrastructure

Compute infrastructure | Virtual machines

Virtual machines

Search

Overview All resources Infrastructure

Virtual machines

Virtual Machine Scale Set (VMSS) Compute Fleet Disks + images Capacity + placement Related services Help

Create Switch to classic Reservations Manage view Refresh Export to CSV Open query Assign tags Start ... Group by

You are viewing a new version of Browse experience. Some features may be missing. Click here to access the old experience.

Filter for any field... Subscription equals all Type equals all Resource Group equals all Location equals all Add filter

Name	Subscription	Resource Group	Location	Status	Operating syst...	Size	Public IP addre...	Disk
ubuntu	Azure for Stud...	azurelinux_group	East Asia	Running	Linux	Standard_B1s	20.255.57.172	1
window	Azure for Stud...	azure_window	East Asia	Running	Windows	Standard_B1s	20.2136.238	1

Connecting to azure virtual machine with my virtual machine

```
vboxuser@ubuntu:~/Downloads$ chmod 400 ~/Downloads/ubuntu_key.pem
vboxuser@ubuntu:~/Downloads$ ssh -i ~/Downloads/ubuntu_key.pem azureuser@20.255.57.172
The authenticity of host '20.255.57.172 (20.255.57.172)' can't be established.
ED25519 key fingerprint is SHA256:M26FE5SFKk9AFJ9MM+THKhtNy3HG56RK/05mQbm1RxM.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '20.255.57.172' (ED25519) to the list of known hosts.
Connection reset by 20.255.57.172 port 22
vboxuser@ubuntu:~/Downloads$
```

## **Q2. Create an App Service Plan Provision a Web App in the existing App Service Plan and deploy a simple welcome page on it .**

**Answer:**

**What We'll Do:**

1. Create an App Service Plan (hosting environment)
2. Deploy a Web App with a Node.js runtime
3. Upload a basic "Welcome" web page

### **◆ Step a: Create App Service Plan**

 *"First, I log in to the Azure Portal at [portal.azure.com](https://portal.azure.com). Now I'm creating an App Service Plan."*

1. Go to Create a resource → Search for App Service Plan
2. Click Create
3. Fill in the form:
  - Subscription: (Your active subscription)
  - Resource Group: Create new or use existing (WebAppDemoRG)
  - Name: MyServicePlan
  - Operating system: windows
  - Region: Select your nearest location (e.g., Central India)
  - Pricing Tier: Click Change size → Select Free (F1)

## Screenshot to Take: App Service Plan creation form (with name, region, pricing)

Home > Create a resource > Marketplace > App Service Plan >

### Create App Service Plan ...

App Service plans give you the flexibility to allocate specific apps to a given set of resources and further optimize your Azure resource utilization. This way, if you want to save money on your testing environment you can share a plan across multiple apps. [Learn more](#)

#### Project Details

Select a subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *	<input type="text" value="Azure for Students"/>
Resource Group *	<input type="text" value="(New) WebAppDemoRG"/> <a href="#">Create new</a>

#### App Service Plan details

Name *	<input type="text" value="MyServicePlan"/> <span style="color: green;">✓</span>
Operating System *	<input checked="" type="radio"/> Linux <input type="radio"/> Windows
Region *	<input type="text" value="Central India"/>

#### Pricing Tier

App Service plan pricing tier determines the location, features, cost and compute resources associated with your app. [Learn more](#)

Pricing plan	<input type="text" value="Free F1 (Shared infrastructure)"/> <a href="#">Explore pricing plans</a>
--------------	--

#### Zone redundancy

An App Service plan can be deployed as a zone redundant service in the regions that support it. Your initial instance count will be set based on your zone redundancy configuration. To ensure you'll be able to enable zone redundancy at any point in the lifecycle of your app, enable zone redundancy now. You can decrease the instance count after the App Service plan is created. [Learn more](#)

i Zone redundancy is not available on the selected pricing tier.

Zone redundancy	<input type="radio"/> <b>Enabled:</b> Your App Service plan and the apps in it will be zone redundant. The minimum App Service plan instance count will be two. <input checked="" type="radio"/> <b>Disabled:</b> Your App Service plan and the apps in it will not be zone redundant. The minimum App Service plan instance count will be one.
-----------------	--

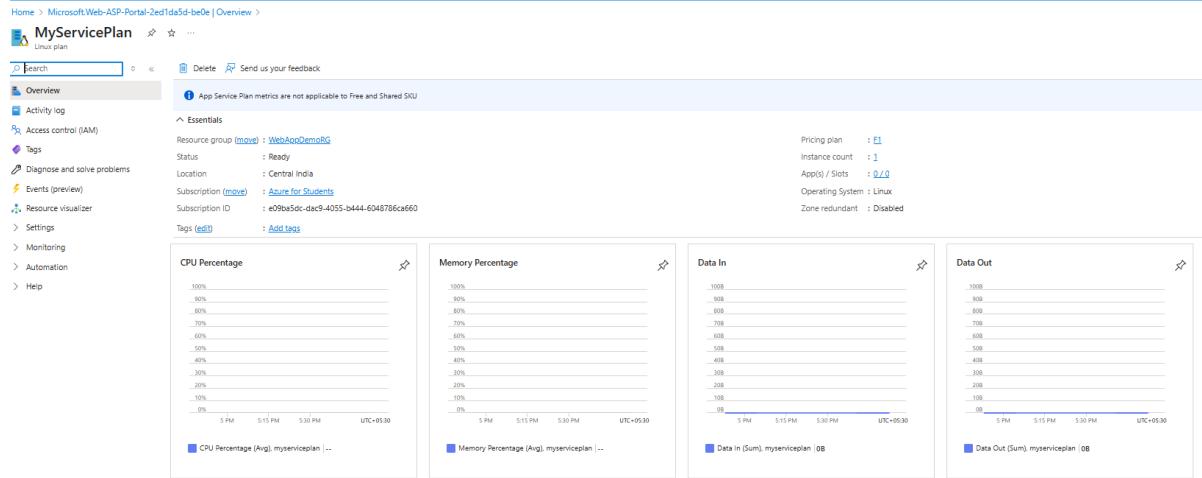
[Review + create](#)

[< Previous](#)

[Next : Tags >](#)

#### 4. Click Review + Create → then Create

"Now the App Service Plan is ready. It defines the compute resources for hosting web apps."



## ◆ Step b: Deploy Web App

"Now I will deploy a web app using the App Service Plan I just created."

1. In Azure Portal, click Create a resource → Search Web App → Click Create
2. Fill in the form:
  - Subscription: Same as before
  - Resource Group: Use WebAppDemoRG
  - Name: welcome-webapp-kriti (*must be globally unique*)
  - Publish: Code
  - Runtime stack: Node.js (e.g., Node 22 LTS)
  - Operating system: linux
  - Region: Same as App Service Plan (central india)
  - App Service Plan: Select Existing → Choose MyServicePlan

## Web App creation form with runtime and selected App Service Plan

Microsoft Azure

Home > Create a resource >

### Create Web App

Basics Database Deployment Networking Monitor + secure Tags Review + create

App Service Web Apps lets you quickly build, deploy, and scale enterprise-grade web, mobile, and API apps running on any platform. Meet rigorous performance, scalability, security and compliance requirements while using a fully managed platform to perform infrastructure maintenance. [Learn more](#)

**Project Details**

Select a subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription \*  Resource Group \*  [Create new](#)

**Instance Details**

Name  -gsgme6ekgfbgceff.centralindia-01.azurewebsites.net

Secure unique default hostname on. [More about this update](#)

Publish \*  Code  Container

Runtime stack \*

Operating System \*  Linux  Windows

Region \*    
Not finding your App Service Plan? Try a different region or select your App Service Environment.

**Pricing plans**

App Service plan pricing tier determines the location, features, cost and compute resources associated with your app. [Learn more](#)

Linux Plan (Central India) \*  [Create new](#)

Pricing plan

**Zone redundancy**

An App Service plan can be deployed as a zone redundant service in the regions that support it. Your initial instance count will be set based on your zone redundancy configuration. To ensure you'll be able to enable zone redundancy at

[Review + create](#) [< Previous](#) [Next : Database >](#)

3. Click Review + Create → then Create

 "Now my web app is being deployed on Azure under the Node.js runtime."

◆ **Step c: Upload a Simple Welcome Page**

 "Now I'll upload a simple HTML welcome page to the web app using the Azure Portal editor."

1. Once the Web App is deployed, go to:
  - o App Services → Click on your app (e.g., welcome-webapp-kriti)
2. In the left menu, go to:
  - o Development Tools → Advanced Tools (Kudu) → Click Go
3. In the new tab (Kudu), go to:
  - o Debug console → CMD
  - o Navigate to: site/wwwroot/
4. Click Upload, and upload a file named index.html with the following content

## index.html in Kudu's wwwroot directory

```
Last login: Fri Jun 20 12:41:18 2025 from 169.254.129.2
```

```
/ \ \ \ / | \ \ / 
 \ \ \ \ / | / \ \ \ \ 
 \ \ \ / \ \ \ / | \ \ \ \ 
 \ \ \ \ \ \ \ \ \ \ \ \ \ \ 
A P P S E R V I C E O N L I N U X >
```

```
Documentation: http://aka.ms/webapp-linux
```

```
NodeJS quickstart: https://aka.ms/node-qs
```

```
NodeJS Version : v22.15.0
```

```
Note: Any data outside '/home' is not persisted
```

```
root@welcome-we-54a487af:/home# ls
ASP.NET LogFiles site
root@welcome-we-54a487af:/home# cd site
root@welcome-we-54a487af:/home/site# ls
deployments locks repository wwwroot
root@welcome-we-54a487af:/home/site# cd wwwroot
root@welcome-we-54a487af:/home/site/wwwroot# ls
hostingstart.html
root@welcome-we-54a487af:/home/site/wwwroot# cat > hostingstart.html
^C
root@welcome-we-54a487af:/home/site/wwwroot# cat hostingstart.html
root@welcome-we-54a487af:/home/site/wwwroot# cat > hostingstart.html
<!DOCTYPE html>
<html>
<head>
<title>Welcome Page</title>
</head>
<body>
<h1>Welcome to my Azure Web App!</h1>
<p>Deployed using App Service Plan.</p>
</body>
</html>
^C
root@welcome-we-54a487af:/home/site/wwwroot# cat hostingstart.html
<!DOCTYPE html>
```

```
☰ Menu ssh://root@169.254.129.3:2222 SSH CONNECTION ESTABLISHED
```

5. Now visit your web app URL:

(<https://welcome-webapp-kriti-gsgme6ekgfbgc01.azurewebsites.net/wwwroot/hostingstart.html>)

**Final web page showing your welcome message in the browser**



## Welcome to my Azure Web App!

Deployed using App Service Plan.

*"I successfully created an App Service Plan, deployed a Node.js web app using it, and uploaded a welcome page. The page is now live and accessible publicly on Azure."*

## Q3. Create ACR and pull image from ACR and Create a container from it

**Answer:**

**Goal:**

- **Create a private Docker registry (ACR) in Azure**
- **Push a Docker image to ACR**
- **Pull and run it from any VM**
- ◆ **Step a) Provision ACR**

*"I'll first create an Azure Container Registry to store my Docker images securely."*

1. Go to <https://portal.azure.com>
2. Click Create a resource → Containers → Container Registry
3. Fill in:
  - Registry name: kritiacr (*must be globally unique*)
  - Resource Group: Create or select (e.g., ACRGroup)
  - Location: (e.g., Central India)
  - **SKU: Choose Basic**
4. **Click Review + Create → Create**

## Final ACR creation page before clicking "Create"

The screenshot shows the 'Create container registry' page in the Microsoft Azure portal. The top navigation bar includes 'Microsoft Azure', a search bar, and a breadcrumb trail: Home > Create a resource > Marketplace > Container Registry > Create container registry.

**Project details:**

- Subscription: Azure for Students
- Resource group: (New) ACRgroup (selected)
- Or Create new

**Instance details:**

- Registry name: kritiacr (selected)
- Location: Central India
- Domain name label scope: Unsecure
- Registry domain name: kritiacr.azurecr.io
- Use availability zones: (checkbox) Availability zones are activated on premium registries and in regions that support availability zones. [Learn more](#)
- Pricing plan: Standard
- Role assignment permissions mode (Preview): RBAC Registry Permissions (selected)

**Networking:**

- Review + create
- < Previous
- Next: Networking >

**"My Azure Container Registry is ready at: kritiacr.azurecr.io"**

### ◊ Step b) Build & Push Image

**"Next, I'll build a Docker image and push it to my ACR from my local machine."**

#### ◊ Pre-requisites:

- Docker installed locally
- Azure CLI authenticated (az login)
- Dockerfile ready (example below)

**Sample Dockerfile:**

**FROM nginx**

**COPY index.html /usr/share/nginx/html/index.html**

**"This image will serve a custom HTML page using Nginx."**

**Terminal Steps:**

# 1. Login to ACR from your local machine

```
az acr login --name kritiacr
```

# OR using Docker directly:

```
docker login kritiacr.azurecr.io
```

# 2. Build Docker image

```
docker build -t kritiacr.azurecr.io/hello:v1 .
```

# 3. Push to ACR

```
docker push kritiacr.azurecr.io/hello:v1
```

Docker login + docker build + docker push terminal output

 “Now my image ‘hello:v1’ is stored in my private registry.”

◆ Step c) Pull & Run Container (from VM)

 “Now I’ll SSH into a Linux VM and pull the image from ACR to run it.”

From your Azure VM:

# 1. Login to ACR

```
docker login kritiacr.azurecr.io
```

# 2. Pull the image

```
docker pull kritiacr.azurecr.io/hello:v1
```

# 3. Run the container

```
docker run -d -p 80:80 kritiacr.azurecr.io/hello:v1
```

 Screenshot to capture: docker pull and docker run output on VM terminal

 “The container is now running on port 80 of the VM.”

Test in Browser

Visit:

<http://<your-vm-public-ip>>

You should see your custom HTML served via the Docker container.

 Screenshot to capture: Final web page opened in browser from your VM

“I successfully created an ACR, pushed a Docker image to it, then logged into a VM, pulled the image, and ran it using Docker. The entire process demonstrates private container deployment in Azure.”

## A. Creating a Virtual Network with 2 Subnets for VMs and Database

I started by creating a Virtual Network (VNet) to logically group and isolate Azure resources for communication.

 Steps I Followed:

1. Go to Azure Portal → Search for Virtual Networks → Click Create.

- 2. In the Basics tab:**
  - **Subscription: Azure for Students**
  - **Resource group: vnet\_project\_group (I created a new one)**
  - **Name: MainVNet**
  - **Region: East Asia**
- 3. In the IP Address tab:**
  - **Left the default IP range: 10.0.0.0/16**
- 4. Then I created two subnets:**
  - **Subnet-1: VMSubnet → 10.0.1.0/24 (For Linux VM and Windows VM)**
  - **Subnet-2: DBSubnet → 10.0.2.0/24 (For SQL Database)**
- 5. I reviewed and created the VNet.**

#### Deployed VMs in Subnet-1

- **Linux VM: Named linuxvm, created in VMSubnet**
- **Windows VM: Named windowsvm, also in VMSubnet**

During VM creation, in the Networking tab, I selected:

- **Virtual Network: MainVNet**
- **Subnet: VMSubnet**

---

#### Planned for SQL DB in Subnet-2

Although I didn't deploy a real SQL Server, I reserved DBSubnet to simulate a secure backend subnet that can later host:

- **Azure SQL Managed Instance**
- **SQL Server VM (if required)**

---

## B. Hub-and-Spoke Network: 4 VNets + VM Communication Test

Now I created 4 VNets to design a Hub-and-Spoke architecture, a common Azure enterprise pattern for network security and control.

VNets I Created:

- 1. Management VNet – HUB**
  - **Name: mgmt-vnet**
  - **Address space: 10.10.0.0/16**
  - **Subnet: mgmt-subnet → 10.10.1.0/24**
- 2. Production VNet – Spoke**
  - **Name: prod-vnet**
  - **Address space: 10.20.0.0/16**
  - **Subnet: prod-subnet → 10.20.1.0/24**
- 3. Testing VNet – Spoke**
  - **Name: test-vnet**
  - **Address space: 10.30.0.0/16**
  - **Subnet: test-subnet → 10.30.1.0/24**
- 4. Development VNet – Spoke**

- **Name: dev-vnet**
  - **Address space: 10.40.0.0/16**
  - **Subnet: dev-subnet → 10.40.1.0/24**
- 

## Created Peering (Hub-and-Spoke Architecture)

Then I configured VNet peering between:

- mgmt-vnet ↔ prod-vnet
- mgmt-vnet ↔ test-vnet
- mgmt-vnet ↔ dev-vnet

➡ This allows VMs in the management network to communicate with VMs in all spokes, but the spokes cannot communicate directly with each other (default behavior of hub-and-spoke).

Steps:

1. Go to mgmt-vnet → Peerings → Add
  2. Peer to prod-vnet, check allow traffic both ways
  3. Repeat for test-vnet and dev-vnet
- 

## Launched VMs in Each VNet

I created 1 VM in each of the 4 VNets:

- vm-mgmt in mgmt-vnet
  - vm-prod in prod-vnet
  - vm-test in test-vnet
  - vm-dev in dev-vnet
- 

## Verified Communication (Ping Test)

Then I logged in to the Management VM (vm-mgmt) using SSH and tested connectivity using ping:

ping 10.20.1.4 # vm-prod IP

ping 10.30.1.4 # vm-test IP

ping 10.40.1.4 # vm-dev IP

Ping was successful to all 3 VMs, which verified that VNet peering and routing were working correctly.

It also demonstrated that the Management VM could control and monitor all other VMs across different VNets, just like a real-world management setup.