Week-2 Assignment

Task 1: Deploy Linux and Windows Virtual Machines on Azure and Access via SSH and RDP

This task involves creating both Linux (Ubuntu) and Windows (Server 2019/Windows 10) Virtual Machines (VMs) on Microsoft Azure, configuring their network settings, and accessing them securely via SSH (for Linux) and RDP (for Windows).

Part 1: Deploying and Accessing a Linux Virtual Machine

Step 1.1: Create a Linux VM on Azure Portal

1. Log in to Azure Portal:
   Navigate to https://portal.azure.com.

2. Search for "Virtual Machines" in the top search bar and select it.

3. Click on "+ Create" → "Azure virtual machine".

4. In the Basics tab, fill in the required details:

   Subscription: Select your Azure subscription.
   Resource Group: Create a new one (e.g., `VM-Resources`) or select an existing one.
   Virtual Machine Name: `LinuxVM`
   Region: Choose a region near your location (e.g., `East US`).
   Availability Options: Leave as default (`No infrastructure redundancy required`).
   Image: Select `Ubuntu 20.04 LTS - Gen 1`
   Size: Choose `Standard B1s` or an appropriate size based on your needs.
   Authentication Type: Choose `SSH public key` (recommended) or `Password`.

   Username: `azureuser`
   SSH Public Key: Paste your existing key or generate a new one within Azure.
   Inbound port rules: Select `Allow selected ports`, then choose `SSH (22)`.

5. Click Next through the remaining tabs (Disks, Networking, etc.), or modify as needed.

6. Click Review + create → Create.
   The deployment will begin, and your VM will be provisioned in a few minutes.

Step 1.2: Accessing the Linux VM via SSH

Locate the Public IP Address:

Go to Virtual Machines → LinuxVM → Overview.
Copy the Public IP address.

Access via Terminal (macOS/Linux/WSL):

```bash
ssh azureuser@<Public_IP>
```

> Example:

```bash
ssh azureuser@20.81.XX.XX
```

The first time you connect, type `yes` when prompted to trust the host.

Access via PuTTY on Windows:

1. Download and install PuTTY if not already installed.
2. Convert your private key (`.pem`) to `.ppk` using PuTTYgen if needed.
3. Open PuTTY:

   Host Name (or IP address): `<Public_IP>`
   Port: `22`
   Under `Connection → SSH → Auth`, browse and load your `.ppk` private key.
4. Click Open to start the session.
5. Log in as `azureuser`.

Part 2: Deploying and Accessing a Windows Virtual Machine

Step 2.1: Create a Windows VM on Azure Portal

1. Go to Azure Portal → Search and click on "Virtual Machines".

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

3. In the Basics tab, provide the following:

   Subscription: Select your subscription.
   Resource Group: Use the same or create a new one.
   Virtual Machine Name: `WindowsVM`
   Region: Select the same or nearby region.
   Availability Options: Leave as default.
   Image: Choose `Windows Server 2019 Datacenter` or `Windows 10 Pro`.
   Size: Choose `Standard B2s` or higher.
   Administrator Account:

     Username: `azureadmin`
     Password: Enter a strong password.
   Inbound port rules: Select `Allow selected ports`, then choose `RDP (3389)`.

4. Click Next through other tabs as needed or customize.

5. Click Review + create → Create.

Step 2.2: Accessing the Windows VM via RDP

Locate the Public IP Address:

Go to Virtual Machines → WindowsVM → Overview.
Copy the Public IP address.

Access via Remote Desktop Connection (Windows OS):

1. Press `Win + R`, type `mstsc`, and press Enter.
2. In the Remote Desktop Connection window:

   Computer: Enter the VM's public IP.
   Click Connect.
3. When prompted:

   Username: `azureadmin`
   Password: Enter the password set during VM creation.
4. Accept any certificates and proceed.

Access via Microsoft Remote Desktop (macOS):

1. Install Microsoft Remote Desktop from the Mac App Store.
2. Open the app → Click Add PC.
3. Enter:

   PC name: VM's public IP
   User account: `azureadmin` + password
4. Save and double-click to start the connection.

 # Task 2: Create an App Service Plan, Provision a Web App, and Deploy a Welcome Page

---

Part 1: Create an Azure App Service Plan

An App Service Plan defines the region (Datacenter), pricing tier, and compute resources used for your Web App.

Step-by-Step Instructions:

1. Log in to the [Azure Portal](https://portal.azure.com).

2. In the Search bar, type App Service Plans and click the result.

3. Click on "+ Create" to start creating a new App Service Plan.

4. In the Basics tab, configure the following settings:

   Subscription: Select your Azure subscription.
   Resource Group: Use an existing group or click Create new (e.g., `WebAppResources`).
   Name: Provide a name for your App Service Plan (e.g., `MyAppServicePlan`).
   Operating System: Choose `Windows` or `Linux` (Linux is recommended for HTML/static sites).
   Region: Select a region closest to your location or users.
   Pricing Tier: Click Change Size → Choose:

    `Free (F1)` – suitable for testing
    Or choose `B1` or higher for production use

5. Once all fields are filled, click Review + create, then Create.

Part 2: Provision a Web App in the App Service Plan

A Web App is a platform-as-a-service (PaaS) offering to host web applications, REST APIs, and mobile backends.

Step-by-Step Instructions:

1. In Azure Portal, search for App Services in the search bar and click on it.

2. Click on "+ Create" to start provisioning a Web App.

3. In the Basics tab, enter the following:

   Subscription: Your Azure subscription
   Resource Group: Select the same group used in the App Service Plan (e.g., `WebAppResources`)
   Name: A globally unique name for your web app (e.g., `my-welcome-webapp`)

   > This will define your web app URL: `https://my-welcome-webapp.azurewebsites.net`
   Publish: Select `Code`
   Runtime Stack: Choose based on your tech stack (e.g., `PHP`, `.NET`, `Node.js`, or `Python`)

   > For HTML-only apps, this doesn't affect the behavior.
   Operating System: Match the one chosen in App Service Plan (Linux or Windows)
   Region: Must match the App Service Plan region
   App Service Plan: Click Change, and select the existing App Service Plan (`MyAppServicePlan`)

4. Click Review + Create, then click Create after validation.

Part 3: Deploy a Simple Welcome Page

There are two methods to deploy your static HTML page: using the App Service Editor or VS Code with Azure Extensions.

Option A: Using App Service Editor (Browser-Based Editor)

1. Go to App Services → Click your Web App (`my-welcome-webapp`).

2. In the left menu, scroll to Development Tools → Click App Service Editor (Preview).

3. Click Go →, and a new browser tab will open.

4. In the App Service Editor, navigate to the folder `/home/site/wwwroot`.

5. Create or edit a file named `index.html`:

   ```html
   <!DOCTYPE html>
   <html>
   <head>
     <title>Welcome</title>
   ```

```
</head>
<body>
  <h1>Welcome to my Azure Web App!</h1>
  <p>Deployed using App Service Plan.</p>
</body>
</html>
```

6. Save the file.

7. Open your browser and go to:
`https://my-welcome-webapp.azurewebsites.net`

Option B: Using Visual Studio Code with Azure App Service Extension

1. Open Visual Studio Code.

2. Install the following extensions:

Azure Account
Azure App Service

3. Sign in to Azure in VS Code.

4. Prepare a local folder containing `index.html` with the same content as above.

5. In the Azure Extension pane:

Right-click on your Web App (`my-welcome-webapp`) → Click Deploy to Web App
Choose the folder containing your `index.html`.

6. Confirm the deployment when prompted.

7. After deployment, open the app URL in a browser:
`https://my-welcome-webapp.azurewebsites.net`

Task 3: Create Azure Container Registry (ACR), Import Image, and Deploy Container

---

Part 1: Create Azure Container Registry (ACR)
Step-by-Step Instructions:

1. Log in to the [Azure Portal](https://portal.azure.com).

2. In the top search bar, type Container Registries, then click on it.

3. Click on + Create.

4. In the Basics tab, provide the following configuration:

   Subscription: Select your Azure subscription.
   Resource Group: Choose existing or click Create new (e.g., `MyContainerGroup`)
   Registry Name: A globally unique name (e.g., `myuniqueregistry123`)
   Location: Select the region closest to you (e.g., `East US`)
   SKU (Pricing Tier): Select `Basic` (sufficient for test/prod use)

5. Click Review + create → Once validated, click Create.

6. After deployment completes, click Go to Resource to open your new ACR.

Part 2: Import Container Image into ACR

There are two options to get an image into your ACR:

Option A: Import Public Image (No Docker Needed — GUI Only)

1. In your ACR resource page, navigate to the Repositories section.

2. Click Import.

3. Fill in the form:

   Source Type: Docker Hub
   Source Image: e.g., `library/nginx` (official nginx image)
   Target Name: e.g., `nginx`
   Tag: (Optional) `latest`

4. Click OK to begin importing the image into your private registry.

This method pulls an image from Docker Hub and stores it in your ACR without requiring Docker on your machine.

Option B: Push Your Own Image (If Using Docker Locally)

> Only use this if you're using your own custom image and have Docker + Azure CLI installed locally.

1. Login to ACR:

   ```bash
   az acr login --name myuniqueregistry123
   ```

2. Tag your image:

   ```bash
   docker tag myapp myuniqueregistry123.azurecr.io/myapp:v1
   ```

3. Push the image:

```bash
docker push myuniqueregistry123.azurecr.io/myapp:v1
```

Your image is now pushed to your private registry.

Part 3: Deploy a Container Using Azure Container Instances (ACI)

Now that the image is in your ACR, deploy it as a container instance.

Step-by-Step Instructions:

1. In Azure Portal, search for Container Instances and click on it.

2. Click + Create to launch the ACI wizard.

3. Under the Basics tab, fill in:

   Subscription: Select your subscription
   Resource Group: Select the same group (e.g., `MyContainerGroup`)
   Container Name: e.g., `my-nginx-container`
   Region: Same as ACR location (e.g., `East US`)
   Image Source: Select Azure Container Registry
   Registry: Choose your ACR (`myuniqueregistry123`)
   Image: Select the image you imported (e.g., `nginx`)
   Tag: `latest` or the version tag (`v1`)
   OS Type: Linux

4. Under Size, use default or adjust as needed.

5. Under Networking tab:

   DNS Name Label: (Optional) e.g., `my-nginx-demo` (this makes your container accessible at `my-nginx-demo.<region>.azurecontainer.io`)
   Port: Add port `80` if you're deploying a web server like NGINX.

6. Click Review + Create → Then click Create.

Part 4: Access the Running Container

1. After deployment completes, go to the Container Instance resource.

2. Under the Overview tab, look for:

   FQDN (Fully Qualified Domain Name)
    e.g., `my-nginx-demo.eastus.azurecontainer.io`
   Or use the Public IP Address

3. Open the address in your browser:

   You should see the NGINX welcome page or your custom application homepage!

 # Task 4: Deploy a Simple Docker Application Using Azure Container Instance and Configure Container Groups

---

Step-by-Step Deployment Using Azure Portal

Step 1: Create a Container Instance

1. Go to the [Azure Portal](https://portal.azure.com).
2. In the top search bar, search for Container Instances.
3. Click + Create to start creating a new container.

Step 2: Configure Basic Settings

In the Basics tab:

Subscription: Select your active Azure subscription
Resource Group: Choose existing or click Create new (e.g., `myResourceGroup`)
Container Name: `mynginxcontainer`
Region: Choose the same region where ACR is deployed (e.g., `East US`)
Leave Availability zone as default

Click Next: Container.

Step 3: Configure Container Settings

In the Container tab:

Image Source: Select Azure Container Registry
Registry: Choose your ACR (e.g., `Registrynew1`)
Image: Select the image you imported (e.g., `nginx`)
Tag: `latest`
OS Type: `Linux`
Size: Default (1 vCPU, 1.5 GiB RAM)
Port: Add port `80`

Click Next: Networking.

Step 4: Configure Networking

In the Networking tab:

DNS Name Label: Choose a unique DNS prefix (e.g., `mynginx123`)

This will create a public URL:
`http://mynginx123.eastus.azurecontainer.io`
Network Type: Select `Public`
Port: Ensure port `80` is listed

Click Review + create, then click Create to start deployment.

Step 5: Wait for Deployment & Validate Access

Once deployment is complete:

1. Go to the Container Instance resource.
2. In the Overview tab:

   Copy the FQDN (Fully Qualified Domain Name)

   Example: `http://mynginx123.eastus.azurecontainer.io`
3. Paste it in a browser tab to access the application.

You should see the NGINX welcome page if everything is set up correctly.

 # Task 5: Configure Subnet/VNet Topology with Hub-and-Spoke Architecture

---

Part A: Create a VNet with Two Subnets and Deploy VMs and SQL DB

Step A1: Create Virtual Network with 2 Subnets

1. Navigate to Azure Portal → Search "Virtual Networks" → Click + Create.
2. In the Basics tab:

   Subscription: Choose your subscription.
   Resource Group: Create or select (e.g., `MyRG`).
   Name: `MyVnet`
   Region: Choose your Azure region (e.g., `East US`)
3. In IP Addresses tab:

   Address space: `10.1.0.0/16`
   + Add Subnet:

   Name: `Subnet-1`
   Address range: `10.1.1.0/24`

+ Add another Subnet:

Name: `Subnet-2`
Address range: `10.1.2.0/24`
4. Click Review + Create → Then click Create.

Step A2: Deploy Linux VM in Subnet-1

1. Go to Virtual Machines → Click + Create.
2. In the Basics tab:

Name: `LinuxVM`
Image: `Ubuntu Server`
Username/Password: Enter credentials
3. In Networking tab:

VNet: Select `MyVnet`
Subnet: Select `Subnet-1`
4. Leave other settings default and click Review + Create → Create.

Step A3: Deploy Windows VM in Subnet-1

Repeat the same process as Step A2:

Use Windows Server image.
Name: `WindowsVM`
VNet: `MyVnet`, Subnet: `Subnet-1`

Step A4: Deploy SQL Server and SQL Database

> Azure SQL is PaaS, not IaaS. It won't be deployed inside a subnet but can be secured using Private Endpoints or VNet rules.

1. Go to SQL Servers → Click + Create.
2. Provide:

Name: e.g., `my-sqlserver`
Admin Login/Password
Resource Group: `MyRG`
Region: Same as VNet
3. Click Review + Create → Create.
4. Once ready, go to SQL Databases → Click + Create.
5. Link the database to your SQL Server.

You can later configure Private Endpoint to connect SQL to `Subnet-2`.

Part B: Configure Hub and Spoke Architecture

Step B1: Create 4 Virtual Networks with Subnets

Repeat this process 4 times with the following configuration:

VNet Name	Address Space	Subnet Name	Subnet Range
ManagementVnet	10.0.0.0/16	ManagementSubnet	10.0.1.0/24
ProductionVnet	10.1.0.0/16	ProductionSubnet	10.1.1.0/24
TestingVnet	10.2.0.0/16	TestingSubnet	10.2.1.0/24
DevelopmentVnet	10.3.0.0/16	DevelopmentSubnet	10.3.1.0/24

Steps:

1. Azure Portal → Virtual Networks → + Create.
2. In Basics, select RG and region.
3. In IP Addresses, set the address space.
4. + Add Subnet with the correct name and address range.
5. Repeat for all four VNets.

Step B2: Configure VNet Peering (Hub-and-Spoke)

We will peer each Spoke (Production, Testing, Development) with the Hub (ManagementVnet).

Peering: Spoke to Hub

1. Go to ProductionVnet → Peerings → + Add.
2. Fill:

   Name: `ProductionToHub`
   Peer VNet: `ManagementVnet`
   Enable: Allow virtual network access
3. Click Add.

Repeat the same process for:

`TestingVnet` → Peer with `ManagementVnet` (name: `TestingToHub`)
`DevelopmentVnet` → Peer with `ManagementVnet` (name: `DevToHub`)

Peering: Hub to Spokes

1. Go to ManagementVnet → Peerings → + Add
2. Peer with:

   `ProductionVnet` (name: `HubToProduction`)
   `TestingVnet` (name: `HubToTesting`)
   `DevelopmentVnet` (name: `HubToDev`)

Ensure "Allow virtual network access" is enabled for all.

Step B3: Deploy VM in Each Subnet

For each VNet, create one VM:

VM Name	VNet	Subnet
ManagementVM	ManagementVnet	ManagementSubnet
ProductionVM	ProductionVnet	ProductionSubnet
TestingVM	TestingVnet	TestingSubnet
DevelopmentVM	DevelopmentVnet	DevelopmentSubnet

Steps:

1. Go to Virtual Machines → + Create.
2. Choose OS (Ubuntu or Windows), set credentials.
3. In Networking:

   Select the correct VNet and subnet.
4. Deploy.

Step B4: Allow ICMP (Ping) via Network Security Groups

By default, ICMP (ping) is blocked in Azure. Follow these steps to allow it.

Step 1: Create NSG Rule

1. Go to Network Security Groups → + Create

   Name: `AllowICMPNSG`
   Associate with your Resource Group and region
2. In the NSG → Inbound Rules → + Add

   Name: `Allow-ICMP`
   Source: Any
   Destination: Any
   Port: ``
   Protocol: `ICMP`
   Action: Allow
   Priority: 100
3. Click Add

Step 2: Associate NSG with Subnets

1. Go to Virtual Networks → select each VNet → Subnets
2. Click on the subnet → Associate NSG → Choose `AllowICMPNSG`
3. Repeat for all four subnets.

Step B5: Verify Connectivity from Management VM

1. Go to ManagementVM → Connect using SSH (or RDP if Windows).
2. Run the following ping tests using private IPs of other VMs:

   `ping <ProductionVM_Private_IP>`
   `ping <TestingVM_Private_IP>`
   `ping <DevelopmentVM_Private_IP>`

> You can get each private IP from the VM → Networking tab.

If ping works, Hub-and-Spoke peering is successful.

 # Task 6: Create an Internal and External Load Balancer in Azure

Step 1: Create a Virtual Network

1. Navigate to Azure Portal → Search Virtual Networks → Click + Create.
2. In the Basics tab:

   Name: `MyVNet`
   Address Space: `10.0.0.0/16`
3. In the IP Addresses tab:

   Click + Add Subnet
   Subnet Name: `WebSubnet`
   Subnet Address Range: `10.0.1.0/24`
4. Review and Create the VNet.

Step 2: Create Two Virtual Machines

Repeat the following steps to create both VM1 and VM2:

1. Navigate to Virtual Machines → Click + Create.
2. In the Basics tab:

   Name: `VM1` (or `VM2`)
   Image: Ubuntu Server (or Windows)
   Authentication: Password or SSH
3. In the Networking tab:

   VNet: `MyVNet`
   Subnet: `WebSubnet`
   Public IP:

     Set to None for ILB testing
     Set to Yes for ELB testing (optional)
4. Click Review + Create → Create VM.

Step 3: Install Web Server (Optional but Recommended for Testing)

> SSH into each VM and install NGINX.

```bash
```

```
sudo apt update
sudo apt install -y nginx
echo "This is VM1" | sudo tee /var/www/html/index.html    # On VM1
echo "This is VM2" | sudo tee /var/www/html/index.html    # On VM2
```

Step 4: Create Internal Load Balancer (ILB)

A. Create Load Balancer

1. Navigate to Load Balancers → + Create.
2. In Basics tab:

   Name: `InternalLB`
   Type: `Internal`
   SKU: `Standard`
   Region: Same as VNet
3. In Frontend IP Configuration:

   Click + Add
   Name: `PrivateFrontend`
   VNet: `MyVNet`
   Subnet: `WebSubnet`
   IP address assignment: Static or Dynamic
4. In Backend Pools:

   Name: `ILBBEPool`
   Add VM1 and VM2
   Target NIC IP configuration: Default
5. In Health Probes:

   Name: `Probe80`
   Protocol: TCP
   Port: 80
6. In Load Balancing Rules:

   Name: `LBRule80`
   Frontend IP: `PrivateFrontend`
   Backend Pool: `ILBBEPool`
   Protocol: TCP
   Frontend Port: 80
   Backend Port: 80
   Session persistence: None
7. Click Review + Create → Create the ILB.

Step 5: Create External Load Balancer (ELB)

A. Create a Static Public IP

1. Go to Public IP addresses → Click + Create.
2. Set:

   Name: `MyPublicIP`

   SKU: `Standard`
   Assignment: `Static`
3. Click Review + Create → Create

B. Create the External Load Balancer

1. Navigate to Load Balancers → + Create.
2. In Basics tab:

   Name: `ExternalLB`
   Type: `Public`
   SKU: `Standard`
3. In Frontend IP Configuration:

   Use the previously created `MyPublicIP`
4. In Backend Pools:

   Name: `ELBBEPool`
   Add `VM1` and `VM2`
5. In Health Probes:

   Name: `Probe80`
   Protocol: TCP
   Port: 80
6. In Load Balancing Rules:

   Name: `LBRule80`
   Frontend IP: `MyPublicIP`
   Backend Pool: `ELBBEPool`
   Frontend/Backend Port: 80
   Protocol: TCP
   Session Persistence: None
7. Click Review + Create → Create the ELB

Step 6: Update NSG to Allow HTTP (Port 80)

> Ensure traffic can reach the backend VMs

1. Go to Network Security Groups > Select the NSG assigned to VM NIC or Subnet.
2. Under Inbound Security Rules → Click + Add.
3. Rule Details:

   Priority: 100
   Source: Any
   Protocol: TCP
   Port: 80
   Action: Allow
   Name: `Allow-HTTP`
4. Click Add

Step 7: Test Load Balancers

A. Test Internal Load Balancer

1. From a VM inside the VNet, run:

```bash
curl http://<ILB_PRIVATE_IP>
```

You should see alternating responses (e.g., "This is VM1" / "This is VM2").

B. Test External Load Balancer

1. From your local system/browser:

```bash
curl http://<ELB_PUBLIC_IP>
```

Or open the public IP in a browser. You should see traffic served from either VM1 or VM2.

 # Task 7: Create and Test Azure Application Gateway

Step-by-Step Implementation

Step 1: Create the Virtual Network and Subnets

> Skip this step if already configured.

1. Navigate to Azure Portal > Virtual networks > + Create.
2. Under Basics:

   Resource Group: `AppGW-RG`
   Name: `AppGW-VNet`
   Region: Same for all resources
3. Under IP Addresses:

   Address space: `10.0.0.0/16`
   Subnets:

    `BackendSubnet` (10.0.1.0/24)
    `AppGWSubnet` (10.0.2.0/24) ← Required for Application Gateway
4. Click Review + Create → Create

Step 2: Create Backend Virtual Machines (VMs)

1. Go to Virtual Machines > + Create.
2. Create two Ubuntu VMs:

   Name: `WebVM1` and `WebVM2`
   Image: Ubuntu LTS
   Size: B1s or equivalent
   Authentication: SSH or password
   Public IP: Optional (for initial setup only)
   Network:

     VNet: `AppGW-VNet`
     Subnet: `BackendSubnet`
3. Repeat the process for both VMs.

Step 3: Install and Configure Web Server

SSH into each VM and install a basic NGINX web server.

```bash
# On both VMs
sudo apt update
sudo apt install -y nginx

# Customize each VM's home page
echo "Welcome to VM1" | sudo tee /var/www/html/index.html  # On WebVM1
echo "Welcome to VM2" | sudo tee /var/www/html/index.html  # On WebVM2
```

Test via browser or `curl` using each VM's private IP address:

```bash
curl http://<VM-Private-IP>
```

Step 4: Create the Azure Application Gateway

1. Go to Application Gateways > + Create
2. Under Basics:

   Name: `MyAppGateway`
   Tier: `Standard v2` or `WAF v2`
   Region: Same as VNet
   Availability zone: No preference (or select for HA)
3. Under Frontend IP configuration:

   Choose Public
   Create new Public IP:

     Name: `MyAppGWPublicIP`
     SKU: Standard
     Assignment: Static
4. Under VNet Configuration:

   VNet: `AppGW-VNet`
   Subnet: `AppGWSubnet` (mandatory requirement)

Step 5: Configure Backend Pool

1. Name: `AppGWPool`
2. Click + Add backend targets

   Target type: IP address or FQDN
   Add private IPs of both WebVM1 and WebVM2
3. Click Add

Step 6: Configure HTTP Settings and Routing Rule

Backend HTTP Settings

1. Name: `AppGatewayBackendSetting`
2. Protocol: HTTP
3. Port: 80
4. Additional options: Leave default or enable cookie-based affinity if needed

Listener

1. Name: `AppGatewayListener`
2. Protocol: HTTP
3. Port: 80
4. Frontend IP: Select the previously created Public IP

Routing Rule

1. Name: `HTTPRule`
2. Listener: `AppGatewayListener`
3. Backend Target: `AppGWPool`
4. HTTP Setting: `AppGatewayBackendSetting`

Click Add Rule, then Next → Review + Create → Create

Step 7: Testing the Application Gateway

1. Navigate to Application Gateway > MyAppGateway > Frontend IP Configuration
2. Copy the assigned Public IP
3. Open a browser or terminal on your local machine:

```bash
curl http://<AppGatewayPublicIP>
```

You should see either:

 "Welcome to VM1"
 "Welcome to VM2"

Refresh multiple times to confirm round-robin load balancing behavior.

Step 8: NSG Rules (If Not Already Configured)

Ensure NSG attached to `BackendSubnet` allows inbound HTTP traffic:

1. Go to Network Security Groups
2. Select NSG attached to `BackendSubnet`
3. Add Inbound Rule:

   Source: Any
   Destination: Any
   Port: 80
   Protocol: TCP
   Action: Allow
   Priority: 100

 # Task 8: Set Up a Domain, Deploy a Web Server on Azure VM, and Route Traffic Using Azure DNS

Step-by-Step Implementation

Step 1: Create a Virtual Machine (Web Server)

1. Navigate to Azure Portal → Virtual Machines → + Create.
2. Basics Tab:

   Subscription: Select your active subscription.
   Resource Group: Create new or use existing (e.g., `WebServerRG`).
   Virtual machine name: `MyWebVM`
   Region: Choose closest or preferred region.
   Image: Ubuntu 20.04 LTS (or Windows if preferred).
   Size: `Standard B1s` or appropriate for your load.
   Authentication type: SSH public key or password.
   Inbound ports: Select HTTP (80) and SSH (22).
3. Networking Tab:

   Create or use existing VNet.
   Ensure Network Security Group (NSG) allows inbound traffic on port 80.
4. Click: Review + Create → Create.

Step 2: Install a Web Server on the VM

Once the VM is deployed:

1. SSH into the VM:

   ```bash
   ssh <username>@<public-ip>
   ```

2. Install NGINX (for Ubuntu):

   ```bash
   sudo apt update
   sudo apt install -y nginx
   ```

3. Create a custom welcome page:

   ```bash
   echo "Welcome to My Web Server!" | sudo tee /var/www/html/index.html
   ```

4. Ensure NGINX is running:

   ```bash
   sudo systemctl status nginx
   ```

Step 3: Reserve a Static Public IP

> Dynamic IPs can change on VM restart. Assign a static IP to ensure consistent DNS routing.

1. Go to Azure Portal → Public IP Addresses.
2. Select the public IP associated with your VM.
3. Click Configuration:

   Change Assignment from Dynamic to Static.
4. Click Save.
5. Note down the static IP (e.g., `20.50.60.70`).

Step 4: Create an Azure DNS Zone

1. Navigate to Azure Portal → DNS Zones → + Create.
2. Basics Tab:

   Subscription: Select your subscription.
   Resource Group: Use the same as VM (e.g., `WebServerRG`).
   Name: Your domain name (e.g., `example.com`).
3. Click Review + Create → Create.

Step 5: Create an A Record in the DNS Zone

1. Open your DNS Zone (e.g., `example.com`).
2. Click + Record Set.
3. Configure A Record:

   Name: `@` (for root domain) or `www` (for subdomain like `www.example.com`)
   Type: A
   TTL: 3600 (or as needed)
   IP Address: Static public IP of your VM (e.g., `20.50.60.70`)
4. Click OK.

Step 6: Update Domain Registrar Name Servers

> Point your domain to Azure's DNS service by updating the domain registrar with Azure's DNS nameservers.

1. Go to your domain registrar (e.g., GoDaddy, Namecheap).

2. Locate DNS or Name Server settings.

3. Replace existing name servers with the Azure-provided nameservers found in the DNS zone:

   ```
   ns1-01.azure-dns.com
   ns2-01.azure-dns.net
   ns3-01.azure-dns.org
   ns4-01.azure-dns.info
   ```

4. Save changes. DNS propagation can take from 5 minutes to 48 hours.

Step 7: Verify DNS Resolution and Web Access

Once propagation is complete:

Use Terminal:

```bash
ping example.com
curl http://example.com
```

Or open a web browser:

```
http://example.com
```

> You should see the custom message: "Welcome to My Web Server!"

 # Task 9: Azure Storage Account Full Walkthrough

---

1. Create an Azure Storage Account

1. Go to Azure Portal → Search Storage Accounts → Click + Create.

2. Basics Tab:

   Subscription: Select your active Azure subscription.
   Resource Group: Use existing or create new (e.g., `StorageRG`).
   Storage Account Name: Must be globally unique (e.g., `mystorage12345`).
   Region: Select closest Azure region.
   Performance: Standard
   Redundancy: Choose based on requirement:

     LRS (Locally Redundant Storage)
     GRS (Geo-Redundant Storage) for high availability

3. Networking Tab:

   Allow access from all networks or restrict to specific VNets/subnets.

4. Data Protection Tab:

   Enable soft delete, versioning, blob change feed (optional).

5. Advanced / Tags (optional):

   Configure hierarchical namespace if required for Data Lake Gen2.
   Set tags for cost management.

6. Click Review + Create → Create.

---

2. Upload and Access a Blob

1. Navigate to Storage Account → Containers → + Container.

   Name: `testcontainer`
   Public access level: Private (default), Blob, or Container

2. Open the container → Click Upload.

   Select and upload any file (e.g., `sample.txt`).

3. Once uploaded:

   Click the blob → Copy its URL.
   Test access:

```bash
curl "<blob-url>"
```

---

3. Explore Authentication Methods

A. Access Keys

1. Go to Storage Account → Access Keys.
2. Copy Key1 or Key2 and Storage Account Name.
3. Use them with tools like Azure Storage Explorer, CLI, or SDKs.

B. Shared Access Signature (SAS)

1. Navigate to Storage Account → Shared access signature.
2. Configure:

   Services: Blob, File, etc.
   Permissions: Read, Write, List, Delete
   Start/Expiry Time
3. Click Generate SAS and connection string.
4. Copy SAS Token or URL and test:

```bash
curl "<blob-url>?<sas-token>"
```

C. Azure AD Authentication

1. Assign role to user or service principal:

   Go to Storage Account → IAM → Add Role Assignment.
   Role: Storage Blob Data Reader
   Assign to: Your user or service principal
2. Authenticate using Azure CLI:

```bash
az login
az storage blob list \
  --account-name <account-name> \
  --container-name testcontainer \
  --auth-mode login
```

---

4. Use Azure Storage Explorer

1. Download from https://azure.microsoft.com/en-us/features/storage-explorer

2. Install and sign in with Azure account OR connect using:

   Access key

   SAS token
   Azure AD authentication

3. Browse blobs, containers, upload/download data.

---

5. Generate a SAS Token and Test Scope

1. Go to Storage Account → Shared access signature.
2. Set:

   Service: Blob
   Permissions: Read only
   Expiry time: e.g., +1 hour
3. Generate SAS → Copy the SAS URL.
4. Test:

   Try reading blob (should work).
   Try deleting blob (should fail if delete permission not given).

---

6. Create a Stored Access Policy (SAP)

1. Navigate to Container → Access policy → + Add policy.

2. Set:

   Name: e.g., `readonlypolicy`
   Permissions: Read, List
   Start/Expiry: Set valid time range

3. Save policy.

4. Generate a SAS token referencing this policy:

   Use Azure CLI or Portal
   Policy ID is passed in the SAS URL

5. Test SAS-based access.

6. Try removing policy → Previously issued tokens should stop working.

---

7. Understand Access Tiers

Access Tier	Use Case	Cost	Latency
Hot	Frequently accessed data	High	Low
Cool	Infrequently accessed	Lower	Medium
Archive	Long-term, rarely accessed	Lowest	High

How to Set:

Per-blob:

```bash
az storage blob set-tier \
  --account-name <name> \
  --container-name testcontainer \
  --name sample.txt \
  --tier Cool
```

Default tier:

Set in Storage Account > Configuration > Default Access Tier

---

8. Apply Lifecycle Management Policy

1. Go to Storage Account → Lifecycle Management → + Add Rule.
2. Define rule:

   Name: `cool-tier-policy`
   Filters: Apply to all blobs or with a prefix
   Add Actions:

     Move to Cool after 30 days
     Delete after 365 days
3. Save and test:

   Upload blobs
   Wait for conditions or simulate using logs

---

9. Configure Object Replication

1. Pre-requisites:

   Two storage accounts
   Same Azure region
   Blob versioning enabled on both

2. On Source Storage Account → Object replication → + Add Rule:

   Select Destination Storage Account
   Map Source and Destination containers

3. Save and test:

   Upload to source container
   Wait for replication
   Verify in destination container

---

10. Create and Use Azure File Share

1. Go to Storage Account → File Shares → + File Share.

Name: `myshare`
Quota: e.g., 5 GB

2. After creation:

Upload files via portal
Mount on Windows/Linux using SMB

Mount Example (Windows):

```powershell
net use Z: \\<storage-account-name>.file.core.windows.net\myshare /u:<storage-account-name>
<storage-key>
```

---

11. Configure Azure File Sync

1. On On-Prem Server:

Install Azure File Sync Agent
Register server using Storage Sync Service

2. In Azure:

Create Storage Sync Service
Register on-prem server
Create Sync Group:

Add Cloud endpoint: Azure file share
Add Server endpoint: Local folder path

3. Test:

Add file on server → syncs to Azure file share
Add file to Azure → syncs to on-prem folder

 # Task 10: Azure Storage Account - Full Feature Exploration

Step 1: Create a Storage Account

Instructions:

1. Navigate to Azure Portal → Search for Storage Accounts → Click on + Create.
2. Basics:

Subscription: Select your Azure subscription.
Resource Group: Choose an existing one or create new (e.g., `StorageRG`).
Storage Account Name: e.g., `mystorageacctdemo001` (must be globally unique).
Region: Choose the region nearest to you.
Performance: Standard.
Redundancy: Choose `LRS` (Locally Redundant), `GRS`, or `ZRS`.

3. Advanced Tab:

Enable Blob public access (if testing public blobs).
Enable Soft delete and Blob versioning.

4. Networking Tab:

Allow access from All Networks or a selected VNet.

5. Data Protection: Optionally enable blob recovery options.
6. Review + Create → Click Create.

Step 2: Upload and Access Blob Data

Instructions:

1. Open the Storage Account → Containers → + Container

Name: `testcontainer`
Public Access Level: `Private` or `Blob`

2. Go inside the container → Upload → Select a file → Upload.
3. After uploading:

Click on the file → Copy Blob URL.
Test using browser/cURL:

```bash
curl <blob-url>
```

Step 3: Explore Authentication Techniques

3.1 Access Keys

1. Navigate to Access Keys section.
2. Copy either `key1` or `key2` and the Connection String.
3. Use this in Azure Storage Explorer or scripts.

3.2 Shared Access Signature (SAS)

1. Go to Shared Access Signature section.
2. Configure:

Services: Blob/File/Queue/Table
Permissions: Read/Write/Delete/List
Expiry time

3. Generate the SAS Token/URL.
4. Access the Blob using SAS token in a browser:

```bash
curl "<blob-url>?<sas-token>"
```

3.3 Azure AD Authentication

1. Assign RBAC Role: `Storage Blob Data Reader` to a user/service principal.
2. Authenticate via CLI:

```bash
az login
az storage blob list --account-name mystorageacctdemo001 --container-name testcontainer --auth-mode login
```

Step 4: Azure Storage Explorer

Setup:

1. Download from: https://azure.microsoft.com/en-us/features/storage-explorer/
2. Install and Sign in using:

   Azure Account OR
   Access Key / SAS Token
3. Features:

   Manage blob containers, file shares, tables, and queues
   Upload/download/delete blobs
   View metadata and access tiers

Step 5: Shared Access Signature & Stored Access Policy

5.1 Create SAS

1. From container level → Click Shared access signature
2. Define permissions and expiry → Generate SAS Token
3. Use in a script or browser to test permissions

5.2 Stored Access Policy

1. Container → Access policy → + Add Policy

   Define: Identifier, Permissions, Start/Expiry time
2. Generate SAS with reference to policy ID
3. Test access with SAS token.
4. Revoke policy → SAS will stop working

Step 6: Access Tiers (Hot, Cool, Archive)

Blob-Level Tiers:

1. Upload blob
2. Click on the blob → Change Access Tier

   Hot → For frequent access
   Cool → Less frequent
   Archive → Long-term (rehydration required to read)

Step 7: Apply Lifecycle Management Policy

1. Go to Storage Account → Lifecycle Management
2. Add Rule:

   Move blobs to Cool tier after 30 days
   Delete blobs after 365 days
3. Apply and monitor policies on blob lifecycle

Step 8: Object Replication (Blob Replication)

Pre-Requisites:

Two storage accounts in same region
Enable versioning on both

Instructions:

1. Source Account → Object Replication → Add Rule
2. Choose:

   Source + Destination account
   Map containers
3. Upload blob in source → Check replication in destination

Step 9: Create File Share

1. Storage Account → File Shares → + File Share

   Name: `myfileshare`
   Quota: e.g., 5 GB
2. Upload files directly via portal
3. Use Mount command (Windows):

  ```cmd
  net use Z: \\<storageaccount>.file.core.windows.net\myfileshare /u:<storageaccount> <accesskey>
  ```

Step 10: Configure Azure File Sync

10.1 On-Prem Setup

1. Use Windows Server 2012 R2 or later
2. Install [Azure File Sync Agent](https://learn.microsoft.com/en-us/azure/storage/files/storage-sync-files-agent-install)

10.2 Azure Setup

1. Go to Storage Sync Services → + Create
2. Register your on-prem server

10.3 Create Sync Group

1. Inside the Sync Service:

   Add your File Share (cloud endpoint)
   Add registered server and path (server endpoint)

10.4 Validate

Create files on-prem → Confirm sync to Azure
Upload to Azure File Share → Sync to server
