**Week -3**

**Task-1: Deploy Linux and Windows virtual machine and access them using SSH and RDP**

**Step 1: Set Up an Azure Account**

1. Sign in to your Azure account. If you don't have one, you can create a free account at [Azure Free Account](https://azure.microsoft.com/free/).

**Step 2: Create a Linux Virtual Machine**

1. **Navigate to the Azure portal**: [Azure Portal](https://portal.azure.com).
2. **Create a new resource**: Click on "Create a resource" in the left-hand menu.
3. **Select Virtual Machine**: Under "Popular", select "Virtual Machine".
4. **Configure the VM**:
   * **Subscription**: Select your subscription.
   * **Resource group**: Create a new resource group or select an existing one.
   * **Virtual machine name**: Enter a name for your VM.
   * **Region**: Select a region close to you.
   * **Image**: Choose "Ubuntu Server" or any other preferred Linux distribution.
   * **Size**: Select a VM size based on your needs.
   * **Authentication type**: Choose "SSH public key".
   * **Username**: Enter a username.
   * **SSH public key**: Paste your SSH public key (you can generate this using ssh-keygen on your local machine).
5. **Networking**: In the "Networking" tab, ensure the following:
   * **Virtual network**: Create or select a virtual network.
   * **Public IP**: Ensure a public IP is assigned.
   * **Inbound ports**: Allow SSH (port 22).
6. **Review and create**: Click "Review + create", review your settings, and then click "Create".

**Step 3: Create a Windows Virtual Machine**

1. **Navigate to the Azure portal**: [Azure Portal](https://portal.azure.com).
2. **Create a new resource**: Click on "Create a resource" in the left-hand menu.
3. **Select Virtual Machine**: Under "Popular", select "Virtual Machine".
4. **Configure the VM**:
   * **Subscription**: Select your subscription.
   * **Resource group**: Create a new resource group or select an existing one.
   * **Virtual machine name**: Enter a name for your VM.
   * **Region**: Select a region close to you.
   * **Image**: Choose "Windows Server" (e.g., Windows Server 2019 Datacenter).
   * **Size**: Select a VM size based on your needs.
   * **Authentication type**: Choose "Password".
   * **Username**: Enter a username.
   * **Password**: Enter a strong password.
5. **Networking**: In the "Networking" tab, ensure the following:
   * **Virtual network**: Create or select a virtual network.
   * **Public IP**: Ensure a public IP is assigned.
   * **Inbound ports**: Allow RDP (port 3389).
6. **Review and create**: Click "Review + create", review your settings, and then click "Create".

**Step 4: Access the Linux VM using SSH**

1. **Obtain the public IP address** of your Linux VM from the Azure portal.
2. **SSH into the VM**: Open a terminal on your local machine and run the following command:

ssh vaidikjetani@192.168.1.3

**Step 5: Access the Windows VM using RDP**

1. **Obtain the public IP address** of your Windows VM from the Azure portal.
2. **Connect using RDP**:
   * **Windows**: Open Remote Desktop Connection and enter the public IP address.
   * **macOS**: Use Microsoft Remote Desktop from the App Store, enter the public IP address.
   * **Linux**: Use an RDP client like remmina, enter the public IP address.

Log in with the username and password you specified during the VM setup.

**Task-2 : Create an App Service Plan**

**Step 1: Create an App Service Plan**

1. **Navigate to Create a Resource**:
   * Click on "Create a resource" in the left-hand menu.
2. **Search for App Service Plan**:
   * In the "New" window, search for "App Service Plan" in the search bar.
   * Select "App Service Plan" from the results.
3. **Configure Basic Settings**:
   * **Subscription**: Choose your Azure subscription.
   * **Resource Group**: Create a new resource group or select an existing one.
   * **Name**: Enter a name for your App Service Plan.
   * **Region**: Select a region close to your users. (e.g., East US, West Europe)
   * **Operating System**: Choose either Windows or Linux based on your requirements.
   * **Pricing Tier**: Select a pricing tier based on your needs. Click on "Change size" to see the available options and select the appropriate one.
4. **Review and Create**:
   * Click on "Review + create".
   * Review your settings and click "Create" to deploy the App Service Plan.

**Step 2: Verify the Deployment**

1. **Check the Deployment Status**:
   * Once you click "Create", Azure will start deploying your App Service Plan. You can monitor the progress in the "Notifications" area.
   * After the deployment is complete, you will get a notification.
2. **Access the App Service Plan**:
   * Go to "Resource groups" in the left-hand menu.
   * Select the resource group where you created the App Service Plan.
   * You should see your new App Service Plan listed among the resources.

**Step 3: Create a Web App in the App Service Plan**

1. **Create a Web App**:
   * In the Azure portal, click on "Create a resource".
   * Search for "Web App" and select it.
   * Click "Create" to start the Web App creation process.
2. **Configure the Web App**:
   * **Subscription**: Select your subscription.
   * **Resource Group**: Select the same resource group as your App Service Plan.
   * **Name**: Enter a name for your web app.
   * **Publish**: Choose "Code" or "Docker Container" based on your application.
   * **Runtime stack**: Select the runtime stack for your application (e.g., .NET, Java, Node.js).
   * **Region**: Choose the same region as your App Service Plan.
   * **App Service Plan**: Select "App Service Plan", then click "Browse" to choose your existing App Service Plan.
3. **Review and Create**:
   * Click "Review + create".
   * Review your settings and click "Create" to deploy the web app.

**Task – 3: Provision a Web App in the existing App Service Plan and deploy a simple welcome page on it .**

**Step 1: Create a Web App**

1. **Navigate to the Azure Portal**:
   * Sign in to the [Azure Portal](https://portal.azure.com).
2. **Create a Web App**:
   * Click on "Create a resource" in the left-hand menu.
   * Search for "Web App" and select it.
   * Click "Create" to start the Web App creation process.
3. **Configure the Web App**:
   * **Subscription**: Select your subscription.
   * **Resource Group**: Select the resource group where your App Service Plan is located.
   * **Name**: Enter a unique name for your web app (e.g., mywebapp123).
   * **Publish**: Choose "Code".
   * **Runtime stack**: Select the runtime stack for your application (e.g., .NET, Java, Node.js, PHP, Python). For simplicity, let's choose "PHP 7.4".
   * **Operating System**: Choose "Windows" or "Linux" based on your App Service Plan. We'll choose "Linux" for this example.
   * **Region**: Choose the same region as your App Service Plan.
   * **App Service Plan**: Click "App Service Plan" and then "Browse" to select your existing App Service Plan.
4. **Review and Create**:
   * Click "Review + create".
   * Review your settings and click "Create".

**Step 2: Deploy a Simple Welcome Page**

1. **Navigate to the Deployment Center**:
   * Once your Web App is created, navigate to it from the "Resource groups" section or directly from the "All resources" section.
   * In the Web App's left-hand menu, under "Deployment," select "Deployment Center".
2. **Set Up Local Git Deployment**:
   * In the Deployment Center, choose "Local Git" as the source control method.
   * Click "Save". Azure will provide you with a Git clone URL for your repository.
3. **Clone the Repository and Deploy Your Code**:
   * Open a terminal or command prompt on your local machine.

Task – 4 : Create ACR and pull image from ACR and Create a container from it

**Step 1: Create Azure Container Registry (ACR)**

1. **Sign in to the Azure Portal**: [Azure Portal](https://portal.azure.com).
2. **Create a new resource**:
   * Click on "Create a resource" in the left-hand menu.
   * Search for "Container Registry" and select it.
   * Click "Create".
3. **Configure the ACR**:
   * **Subscription**: Select your subscription.
   * **Resource Group**: Select an existing resource group or create a new one.
   * **Registry Name**: Enter a unique name for your registry (e.g., myacr123).
   * **Region**: Choose the same region as your other resources.
   * **SKU**: Choose a SKU (e.g., Basic for testing purposes).
4. **Review and Create**:
   * Click "Review + create".
   * Review your settings and click "Create".

**Step 2: Push a Docker Image to ACR**

1. **Install Azure CLI**: Make sure you have the Azure CLI installed. If not, follow the [installation instructions](https://docs.microsoft.com/en-us/cli/azure/install-azure-cli).
2. **Log in to Azure**:

az login

1. **Log in to your ACR**:

az acr login --name myacr123

1. **Tag your Docker image**:
   * Assume you have a Docker image named myapp:

docker tag myapp myacr123.azurecr.io/myapp:latest

1. **Push the image to ACR**:

docker push myacr123.azurecr.io/myapp:latest

**Step 3: Create a Container Instance from the Image**

1. **Create the Container Instance**:
   * Go to the Azure portal.
   * Click on "Create a resource".
   * Search for "Container Instances" and select it.
   * Click "Create".
2. **Configure the Container Instance**:
   * **Subscription**: Select your subscription.
   * **Resource Group**: Select the same resource group as your ACR.
   * **Container name**: Enter a name for your container (e.g., mycontainer).
   * **Region**: Choose the same region as your ACR.
   * **Image source**: Select "Azure Container Registry".
   * **Registry**: Select your registry (myacr123).
   * **Image**: Enter the image name (myapp:latest).
   * **OS type**: Choose the appropriate OS type (Linux or Windows).
   * **Size**: Configure the size of the container instance as needed.
3. **Review and Create**:
   * Click "Review + create".
   * Review your settings and click "Create".

**Step 4: Access the Container Instance**

1. **Get the FQDN**:
   * After the deployment is complete, go to the resource group and select your container instance.
   * Note the FQDN (Fully Qualified Domain Name) of your container instance.
2. **Access the Web App**:
   * Open a web browser and navigate to the FQDN of your container instance (e.g., <http://mycontainer.eastus.azurecontainer.io>).

**Task-5 : Create Container Instance and deploy a simple docker application on it.**

**Step 2: Create a Simple Docker Application**

1. **Create a simple Docker application**:
   * Create a directory for your Docker application:

mkdir mydockerapp

cd mydockerapp

* + Create a file named Dockerfile with the following content:

FROM nginx:latest

COPY . /usr/share/nginx/html

* + Create a file named index.html with the following content:

<!DOCTYPE html>

<html>

<head>

<title>Welcome to Azure Container Instances</title>

</head>

<body>

<h1>Hello from your Docker container!</h1>

</body>

</html>

1. **Build the Docker image**:
   * Open a terminal in the mydockerapp directory and run:

docker build -t mydockerapp .

**Step 3: Push the Docker Image to Azure Container Registry (ACR)**

1. **Create an ACR (if you haven't already)**:
   * Log in to Azure:

az login

* + Create a resource group:

az group create --name myResourceGroup --location eastus

* + Create an Azure Container Registry:

az acr create --resource-group myResourceGroup --name myacr123 --sku Basic

1. **Log in to ACR**:

az acr login --name myacr123

1. **Tag your Docker image**:

docker tag mydockerapp myacr123.azurecr.io/mydockerapp:latest

1. **Push the Docker image to ACR**:

docker push myacr123.azurecr.io/mydockerapp:latest

**Step 4: Create a Container Instance from the ACR Image**

1. **Create the Container Instance**:
   * Use the Azure CLI to create a container instance:

az container create --resource-group myResourceGroup --name mycontainer --image myacr123.azurecr.io/mydockerapp:latest --cpu 1 --memory 1 --registry-login-server myacr123.azurecr.io --registry-username <acr-username> --registry-password <acr-password> --dns-name-label mycontainer --ports 80

Replace <acr-username> and <acr-password> with your ACR username and password. You can retrieve these with:

az acr credential show --name myacr123

**Step 5: Access the Deployed Application**

1. **Get the FQDN**:
   * Once the container instance is created, retrieve the FQDN (Fully Qualified Domain Name):

az container show --resource-group myResourceGroup --name mycontainer --query ipAddress.fqdn

1. **Open the Application in a Web Browser**:
   * Copy the FQDN from the previous command's output and open it in a web browser (e.g., http://mycontainer.eastus.azurecontainer.io).

You should see the "Hello from your Docker container!" message displayed in your browser.

**Task – 6: Create Container Groups and test functionality**

**Step 1: Create Docker Images**

First, let's create Docker images for a simple Nginx front end and a backend API. For this example, we'll use a basic Python Flask API for the backend.

1. **Create Nginx Frontend**:
   * Create a directory for the Nginx frontend:

mkdir nginx-frontend

cd nginx-frontend

* + Create an index.html file:

<!DOCTYPE html>

<html>

<head>

<title>Welcome to Azure Container Instances</title>

</head>

<body>

<h1>Hello from the Nginx container!</h1>

</body>

</html>

* + Create a Dockerfile for Nginx:

FROM nginx:latest

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

* + Build the Docker image:

docker build -t nginx-frontend .

1. **Create Python Flask API Backend**:
   * Create a directory for the Flask backend:

mkdir flask-backend

cd flask-backend

* + Create a app.py file:

from flask import Flask

app = Flask(\_\_name\_\_)

@app.route('/')

def hello():

return "Hello from the Flask backend!"

if \_\_name\_\_ == '\_\_main\_\_':

app.run(host='0.0.0.0', port=5000)

* + Create a requirements.txt file:

Flask

* + Create a Dockerfile for Flask:

FROM python:3.9-slim

WORKDIR /app

COPY requirements.txt requirements.txt

RUN pip install -r requirements.txt

COPY app.py app.py

CMD ["python", "app.py"]

* + Build the Docker image:

docker build -t flask-backend .

**Step 2: Push Docker Images to Azure Container Registry (ACR)**

1. **Login to ACR**:
   * Ensure you're logged into your Azure account:

az login

* + Create an ACR (if not already created):

az acr create --resource-group myResourceGroup --name myacr123 --sku Basic

* + Login to ACR:

az acr login --name myacr123

1. **Tag and Push Docker Images**:
   * Tag the images:

docker tag nginx-frontend myacr123.azurecr.io/nginx-frontend:latest

docker tag flask-backend myacr123.azurecr.io/flask-backend:latest

* + Push the images:

docker push myacr123.azurecr.io/nginx-frontend:latest

docker push myacr123.azurecr.io/flask-backend:latest

**Step 3: Create Container Group in Azure Container Instances**

1. **Create Container Group using Azure CLI**:

az container create \

--resource-group myResourceGroup \

--name mycontainergroup \

--image myacr123.azurecr.io/nginx-frontend:latest \

--image myacr123.azurecr.io/flask-backend:latest \

--cpu 1 \

--memory 1.5 \

--registry-login-server myacr123.azurecr.io \

--registry-username <acr-username> \

--registry-password <acr-password> \

--dns-name-label mycontainergroup \

--ports 80 5000

Replace <acr-username> and <acr-password> with your ACR credentials. You can retrieve these with:

az acr credential show --name myacr123

**Step 4: Test the Functionality**

1. **Get the FQDN of the Container Group**:

az container show --resource-group myResourceGroup --name mycontainergroup --query ipAddress.fqdn

1. **Access the Nginx Frontend**:
   * Open a web browser and navigate to the FQDN retrieved in the previous step. You should see the Nginx welcome page.
2. **Access the Flask Backend**:
   * Open a web browser and navigate to the FQDN with port 5000 (e.g., http://mycontainergroup.eastus.azurecontainer.io:5000). You should see the Flask backend message.