Scenario 1: Your team needs to deploy a virtual machine in azure portal to test a new software application. The team has requested both windows and Linux virtual machines.

Question: How could you set up these virtual machines and what considerations are needed for pricing and OS license?

Answer:

To set up both Windows and Linux virtual machines (VMs) in the Azure portal, follow these steps:

Setting Up a Windows Virtual Machine

1. **Sign in to Azure**: Go to the Azure portal and sign in with your credentials.
2. **Create a Virtual Machine**:
    a. In the Azure portal, search for "Virtual machines" and select it.
    b. Click on "Create" and then "Azure virtual machine".
    c. Fill in the necessary details such as the VM name, region, and image. For the image, select "Windows Server 2022 Datacenter".
    d. Choose the VM size based on your requirements.
    e. Configure the administrator account by providing a username and password.
    f. Under "Inbound port rules", select RDP (3389) and HTTP (80) to allow remote desktop and web traffic.
    g. Review and create the VM.
3. **Connect to the VM**: Once the VM is created, use Remote Desktop Protocol (RDP) to connect to it. Download the RDP file from the Azure portal and use it to connect to the VM.

Setting Up a Linux Virtual Machine

1. **Sign in to Azure**: Go to the Azure portal and sign in with your credentials.
2. **Create a Virtual Machine**:
    a. In the Azure portal, search for "Virtual machines" and select it.
    b. Click on "Create" and then "Virtual machine".
    c. Fill in the necessary details such as the VM name, region, and image. For the image, select "Ubuntu Server 22.04 LTS".
    d. Choose the VM size based on your requirements.
    e. Configure the administrator account by selecting SSH public key and providing a username.

    f. Under "Inbound port rules", select SSH (22) and HTTP (80) to allow SSH and web traffic.
    g. Review and create the VM.
3. **Connect to the VM**: Once the VM is created, use SSH to connect to it. Use the SSH key pair generated during the VM creation process to establish the connection.

Pricing Considerations

1. **VM Size**: The cost of a VM depends on its size, which determines the number of CPU cores, memory, and storage resources. Larger VMs with more resources will cost more.
2. **Region**: Pricing varies by region due to differences in local costs, taxes, and market conditions.
3. **Operating System**: Windows VMs generally cost more than Linux VMs due to the licensing fees included in the per-minute cost.
4. **Usage Hours**: You are billed based on the number of hours the VM is running. Stopping the VM when not in use can help reduce costs.

OS License Considerations

1. **Windows VMs**: The license to run Windows Server in Azure is included in the per-minute cost of the VM. You can also use the Azure Hybrid Benefit to save on licensing costs if you have existing Windows Server licenses with Software Assurance.
2. **Linux VMs**: Linux VMs typically do not have additional licensing costs, making them more cost-effective for many use cases.

Scenario 2: The IT security team has requested the sensitive data stored in an azure storage account is encrypted to meet the compliance requirements.

Question: How could you ensure the data stored in azure storage is encrypted and what encryption types are available

To ensure that sensitive data stored in an Azure storage account is encrypted, you can use Azure Storage Service Encryption (SSE). Here's how you can set it up and the types of encryptions available:

Setting Up Encryption in Azure Storage

1. **Sign in to Azure**: Go to the Azure portal and sign in with your credentials.

2. **Navigate to Your Storage Account**: In the Azure portal, search for "Storage accounts" and select the storage account you want to encrypt.
3. **Configure Encryption**:
    a. Under the "Settings" section, select "Encryption".
    b. By default, Azure Storage encrypts data at rest using Microsoft-managed keys. You can continue using these keys or opt for customer-managed keys.
    c. To use customer-managed keys, select "Customer-managed keys" and specify the key vault and key to use. The key vault must have purge protection enabled.

Types of Encryptions Available

1. **Service-Side Encryption (SSE)**:
    a. **Microsoft-Managed Keys**: Azure handles the encryption and key management for you. This is the default option and is suitable for most scenarios.
    b. **Customer-Managed Keys**: You manage your own encryption keys using Azure Key Vault. This provides greater control over key management and rotation.
    c. **Infrastructure Encryption**: Adds a second layer of encryption using a different set of keys managed by Azure. This is useful for highly sensitive data.
2. **Client-Side Encryption**:
    a. Data is encrypted by the client application before it is sent to Azure Storage. This ensures that data is encrypted end-to-end and that only the client has access to the encryption keys.

Key Management

- **Azure Key Vault**: Store and manage your encryption keys in Azure Key Vault. You can use Azure Key Vault to generate, store, and manage keys securely.
- **Key Rotation**: Regularly rotate your encryption keys to enhance security. Azure Key Vault supports automated key rotation.

Scenario 3: You are responsible for setting the devops pipeline in azure devops in your application. The pipeline must deploy the code to an azure app service and notify the team if the deployment fails.

Question: How could you configure this pipeline to meet the requirements?

To configure a DevOps pipeline in Azure DevOps that deploys code to an Azure App Service and notifies the team if the deployment fails, follow these steps:

Setting Up the Pipeline

1. **Sign in to Azure DevOps**: Go to the Azure DevOps portal and sign in with your credentials.
2. **Create a New Pipeline**:
    a. Navigate to your project and select "Pipelines" from the left-hand menu.
    b. Click on "New Pipeline" and choose the location of your source code (e.g., Azure Repos Git, GitHub).
    c. Select your repository and configure the pipeline using the YAML file.
3. **Define the Pipeline YAML**:
    a. Create a `azure-pipelines.yml` file in your repository with the following content:

```
trigger:
   branches:
     include:
       - main

 pool:
   vmImage: 'ubuntu-latest'

 steps:
 - task: UseDotNet@2
   inputs:
     packageType: 'sdk'
     version: '5.x'
     installationPath: $(Agent.ToolsDirectory)/dotnet

 - script: dotnet build --configuration Release
   displayName: 'Build project'

 - task: AzureWebApp@1
   inputs:
     azureSubscription: '<Your Azure Subscription>'
     appType: 'webApp'
     appName: '<Your App Service Name>'
     package: '$(System.DefaultWorkingDirectory)/**/*.zip'
```

b. Replace `<Your Azure Subscription>` and `<Your App Service Name>` with your actual Azure subscription and App Service name.

Setting Up Notifications for Deployment Failures

1. **Create a Notification Subscription**:
    a. In Azure DevOps, go to "Project settings"> "Notifications".
    b. Click on "New subscription" and select "Release deployment completed".
    c. Configure the subscription to notify the team when a deployment fails:
        i. Set the "Event type" to "Release deployment completed".
        ii. Add a filter for "Deployment status" and set it to "Failed".
        iii. Choose the delivery method (e.g., email) and specify the recipients.
2. **Configure Email Notifications**:
    a. Ensure that the email addresses of the team members are correctly configured in Azure DevOps.
    b. You can also set up additional notifications for other events, such as build failures or pull request updates.

Scenario 4: Your organization is moving its on-premises SQL database to azure. The database must remain accessible during migration with minimal downtime.

Question: Which azure service could you use, how could you perform the migration

To migrate on-premises SQL database to Azure with minimal downtime, you can use the **Azure Database Migration Service (DMS)**. This service is designed to handle seamless migrations with minimal disruption to your operations.

Steps to Perform the Migration

1. **Prepare for Migration**:
    a. **Assess the Database**: Use the Data Migration Assistant (DMA) to assess your on-premises SQL Server database. This tool helps identify any compatibility issues and provides recommendations for resolving them.
    b. **Create an Azure SQL Database**: Set up your target Azure SQL Database or Azure SQL Managed Instance in the Azure portal.
2. **Set Up Azure Database Migration Service**:
    a. **Create a Migration Project**: In the Azure portal, search for "Azure Database Migration Service" and create a new migration project. Choose the appropriate subscription, resource group, and region.

    b. **Configure the Source and Target**: Specify your on-premises SQL Server as the source and your Azure SQL Database or Managed Instance as the target.
3. **Perform the Migration**:
    a. **Schema Migration**: Use the DMS to migrate the database schema first. This ensures that the target database structure matches the source.
    b. **Data Migration**: Perform the data migration using DMS. You can choose to use continuous data replication to keep the source and target databases in sync during the migration process.
    c. **Cutover**: Once the data is fully synchronized, perform a cutover to switch the application to the new Azure SQL Database. This step involves minimal downtime as the data is already in sync.

Key Considerations

- **Minimal Downtime**: Using continuous data replication ensures that the source and target databases remain synchronized, minimizing downtime during the cutover.
- **Compatibility**: Ensure that your on-premises SQL Server version is compatible with Azure SQL Database or Managed Instance. The Data Migration Assistant can help with this assessment.
- **Performance**: Monitor the performance of the migration process using Azure Monitor and make any necessary adjustments to optimize the migration.

By following these steps and using Azure Database Migration Service, you can ensure a smooth and efficient migration of your on-premises SQL database to Azure with minimal downtime.