AZURE SCENARIOS

1. Scenario:

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 machine

Question:

How could you set up these virtual machines and what considerations are needed for pricing and os license

Answer:

1. Steps to Deploy Virtual Machines in Azure Portal

A. Deploy a Windows Virtual Machine

1. Log in to Azure Portal (https://portal.azure.com).
2. Navigate to "Virtual Machines" and click "Create" > "Azure virtual machine."
3. Select Image:
    a. Choose Windows Server (e.g., Windows Server 2022) or a Windows 10/11 image.
4. Configure Basic Settings:
    a. Set VM name, Region, and Size (e.g., Standard D2s v3).
    b. Select Administrator account (Username and Password).
5. Networking:
    a. Choose public IP if remote access (RDP) is needed.
    b. Enable RDP (Port 3389) for access.
6. Review and Deploy the VM.

B. Deploy a Linux Virtual Machine

1. Go to "Virtual Machines" and create a new VM.
2. Choose a Linux Image:

    a. Select Ubuntu, CentOS, or Red Hat Enterprise Linux (RHEL).
3. Configure Basic Settings:
    a. Choose a VM size based on workload.
    b. Set authentication: SSH key (preferred) or password.
4. Networking:
    a. Enable SSH (Port 22) for remote access.
5. Review and deploy.

2. Pricing and Licensing Considerations

A. Pricing Considerations

- VM Size & Type: Choose based on CPU, RAM, and workload requirements.
- Region: Prices vary by Azure region.
- Storage Type: Use Standard SSD for cost-efficiency or Premium SSD for performance.
- Reserved Instances: If running long-term, Reserved VM Instances (RIs) reduce costs.

B. OS Licensing Considerations

- Windows VM:
    - Windows VMs incur OS licensing costs as part of the VM price.
    - Use Azure Hybrid Benefit to save costs if you have existing Windows Server licenses.
- Linux VM:
    - Ubuntu, Debian, and CentOS are free.
    - Red Hat and SUSE Linux require paid subscriptions.

2.Scenario:

The IT security team has requested the sensitive data stored in an azure storage account be encrypted to meet complaints requirements?

Question:

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

1. How to Encrypt Data in Azure Storage

Azure provides built-in encryption options for securing data at rest and in transit.

A. Enabling Encryption for Data at Rest (Automatic)

- Azure Storage automatically encrypts all data at rest using Microsoft-managed keys by default.
- You can also use Customer-managed keys (CMK) in Azure Key Vault for more control.
- Steps:
    - Go to Azure Portal > Storage Account.
    - Navigate to Encryption under "Security + Networking."
    - Choose either:
        - Microsoft-managed keys (default) – Azure manages the encryption keys.
        - Customer-managed keys (CMK) – Store and manage keys in Azure Key Vault.
    - Save the changes.

B. Encrypting Data in Transit (Network Encryption)

To secure data during transmission, use:

- TLS (Transport Layer Security) 1.2 or higher for encrypted data transfer.
- Azure Storage Service Encryption ensures encryption when accessing via HTTPS.
- Private Link & VPNs to restrict traffic from public networks.

2. Types of Encryptions Available in Azure Storage

A. Encryption at Rest

1. Storage Service Encryption (SSE) – Automatic encryption of data stored in Blob, File, Table, and Queue storage.
    a. Uses AES-256 encryption (industry standard).
    b. Supports Microsoft-managed keys (MMK) or Customer-managed keys (CMK).
2. Azure Disk Encryption (ADE) – Encrypts OS and data disks for VMs.
    a. Uses BitLocker (Windows) or dm-crypt (Linux).
    b. Keys are stored in Azure Key Vault.

B. Encryption in Transit

1. Transport Layer Security (TLS) – Ensures secure data transfer over HTTPS.

2. Azure Private Link – Encrypts and isolates storage traffic within the Azure network.

3. Best Practices for Compliance

- Use Customer-Managed Keys (CMK) if compliance requires key ownership.
- Enable Azure Key Vault Logging to track access to encryption keys.
- Restrict access using RBAC (Role-Based Access Control) and Private Endpoints.
- Regularly rotate encryption keys to enhance security.

3. Scenario

You are responsible for setting up the devops pipeline in azure devops for your application. The pipeline must deploy 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?

1. Set Up the Azure DevOps Pipeline

1. Create a Pipeline:
    a. In Azure DevOps, go to Pipelines > New Pipeline.
    b. Select your source repository (Azure Repos, GitHub, Bitbucket).
    c. Choose either YAML-based or Classic Editor for configuration.
2. Define Pipeline Stages:
    a. Build: Compile the application code and check for errors.
    b. Test: Run automated tests to ensure functionality.
    c. Publish: Package the application for deployment.
    d. Deploy: Deploy the package to Azure App Service.

2. Configure Deployment to Azure App Service

1. Use an Azure Service Connection:

    a. Set up a service connection in Azure DevOps to authenticate deployments securely.
2. Select the App Service:
    a. Choose the target Azure App Service where the application will be deployed.
3. Configure Deployment Strategy:
    a. Blue-Green Deployment (optional) for zero-downtime updates.
    b. Slot Deployment (use staging slots to test before production release).

3. Set Up Notifications for Deployment Failures

1. Enable Email Alerts:
    a. Go to Project Settings > Notifications in Azure DevOps.
    b. Create a new subscription for failed deployments.
    c. Add the team's email addresses.
2. Integrate with Slack or Microsoft Teams (Optional):
    a. Use Azure DevOps Service Hooks to send failure alerts to Slack or Teams.
    b. Configure a webhook to notify the team when a deployment fails.

4. Ensure Security and Best Practices

- Use Role-Based Access Control (RBAC): Restrict deployment permissions.
- Store Secrets Securely: Use Azure Key Vault or DevOps Library Variables for sensitive credentials.
- Enable Logging & Monitoring: Use Azure Application Insights to track failures and performance issues.

5. Test and Validate the Pipeline

- Push a code change to trigger the pipeline.
- Monitor the execution in Azure DevOps Pipelines.
- Verify deployment logs in Azure App Service.
- Ensure notifications are sent if the deployment fails.

4.Scenario:

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?

1. Azure Service to Use

Azure Database Migration Service (DMS) is the best choice because:

- It supports online (minimal downtime) and offline migrations.
- It automates schema and data transfer.
- It provides continuous data synchronization to reduce downtime.

2. Migration Steps

A. Pre-Migration Planning

1. Assess the Existing Database:
    a. Use Data Migration Assistant (DMA) to check compatibility issues.
    b. Fix any schema changes or deprecated features.
2. Set Up the Azure SQL Environment:
    a. Choose the right Azure SQL service:
        i. Azure SQL Database (single database, PaaS).
        ii. Azure SQL Managed Instance (for full SQL Server compatibility).
        iii. SQL Server on Azure VM (if you need full control).
    b. Configure networking and security for connectivity.

B. Configure Azure Database Migration Service (DMS)

1. Create a DMS Instance in Azure Portal:
    a. Go to Azure Portal > Database Migration Service > Create Service.
    b. Select Online Migration for minimal downtime.
2. Set Up the Migration Project:
    a. Choose the source database (on-premises SQL Server).
    b. Choose the target (Azure SQL Database or Managed Instance).
    c. Establish a secure connection using VPN or ExpressRoute.

C. Perform the Migration

1. Migrate Schema First:
    a. Use DMA or DMS to migrate the database schema.
    b. Validate and fix any issues.
2. Perform Initial Data Migration:

    a. DMS migrates existing data while the source database is still running.
    b. Applications continue using the on-premises database.
3. Enable Continuous Data Replication:
    a. DMS synchronizes any new transactions from the on-premises database.
4. Final Cutover to Azure:
    a. Schedule a cutover window.
    b. Stop traffic to the on-premises database.
    c. Perform a final data sync.
    d. Redirect applications to the Azure database.

3. Post-Migration Validation

1. Verify Data Consistency:
    a. Compare row counts and test queries.
2. Update Application Connection Strings:
    a. Modify applications to connect to the new Azure database.
3. Monitor Performance & Security:
    a. Use Azure Monitor, SQL Insights, and Azure Security Center for ongoing monitoring.

4. Benefits of Azure DMS for Minimal Downtime Migration

✓ Continuous data synchronization avoids major downtime.

✓ Automated migration tools simplify the process.

✓ Secure migration with Azure-based encryption and authentication.