Name:Kavitha R

Domain:Testing

AZURE

1) Setting Up Virtual Machines in Azure:

1. **Sign In to Azure Portal**: Log in to the Azure Portal with your credentials.
2. **Create a Windows VM**:
    a. Go to "Virtual Machines" and click "Add."
    b. Choose your subscription, resource group, VM name, and region.
    c. Select Windows OS, VM size, and configure admin username and password.
    d. Set up the disk and networking settings.
    e. Review and click "Create."
3. **Create a Linux VM**:
    a. Follow the same steps as above but choose a Linux OS image.
    b. Use SSH keys for secure login.
    c. Adjust disk and networking settings.

Pricing and Licensing Considerations:

1. **Pricing**:
    a. Costs depend on VM size, region, and usage duration (pay-as-you-go or reserved).
    b. Additional costs for storage and backup.
2. **OS Licensing**:
    a. **Windows VMs**: Licensing costs are included in the VM price.
    b. **Linux VMs**: Most are free, but check specific distribution licensing.
    c. **Azure Hybrid Benefit**: Save on Windows licenses if you already own them.

2) To ensure that sensitive data stored in an Azure Storage account is encrypted, you can utilize several encryption options provided by Azure.

1. **Ensuring Data is Encrypted:**
2. **Azure Storage Service Encryption (SSE)**:

    a. Data is automatically encrypted at rest with 256-bit AES encryption.
    b. Managed by Azure, and enabled by default for all storage accounts.
3. **Client-Side Encryption**:
    a. Encrypt data before sending it to Azure using Azure SDKs (e.g., Azure.Storage.Blobs).
    b. Ensures Azure doesn't access unencrypted data.
4. **Azure Key Vault**:
    a. Manage your own encryption keys using Azure Key Vault.
    b. Use your keys for SSE by configuring a key vault-backed key.
5. **Access Control**:
    a. Use Role-Based Access Control (RBAC) to limit access to sensitive data.

Encryption Types:

6. **SSE**:
    a. Default: Microsoft-managed keys.
    b. Custom: Use your own keys stored in Azure Key Vault.
7. **Client-Side Encryption**:
    a. Encrypt data before sending it to Azure (e.g., using AES encryption).
8. **TLS**:
    a. Encrypts data in transit between your system and Azure Storage.

Verify Encryption:

9. **Azure Portal**: Check the encryption settings in your storage account.
10. **PowerShell/Azure CLI**: Use commands to check SSE and custom key configurations.

3) To set up a DevOps pipeline in Azure DevOps that deploys code to an Azure App Service and notifies the team if the deployment fails

1. ## Steps to Set Up a DevOps Pipeline:
2. **Create the Azure DevOps Pipeline**:
    a. Go to your Azure DevOps project.
    b. Click "Pipelines" > "New pipeline."
    c. Choose where your code is (e.g., GitHub, Azure Repos).
    d. Pick "YAML" or "Classic Editor" (YAML is often preferred).
3. **Define Pipeline Stages**:

    a. In the YAML file (or Classic Editor), define stages for building and deploying your app.
4. **Configure Notifications**:
    a. Go to "Project Settings" > "Service Hooks" or "Notifications."
    b. Set up alerts to notify your team (e.g., via email, Slack) if the build or deployment fails.
    c. Optionally, integrate with Microsoft Teams or Slack for notifications.
5. **Validate the Pipeline**:
    a. Run the pipeline manually or push code to trigger it.
    b. Check that the notifications work if the deployment fails.

4)To migrate an on-premises SQL database to Azure with minimal downtime, you can use Azure SQL Database or Azure SQL Managed Instance, depending on your specific needs for compatibility and features.

Migration Strategy

1. **Azure Database Migration Service (DMS)**:
    a. Use Azure DMS to facilitate the migration process. It supports online migrations, allowing the source database to remain accessible during the migration.
    b. Set up an Azure DMS instance in your Azure environment.
2. **Pre-Migration Steps**:
    a. Assess your on-premises SQL database for compatibility with Azure SQL. Use the Data Migration Assistant to identify any issues that may arise.
    b. Prepare the target Azure SQL Database or Managed Instance, ensuring it is sized correctly based on your workload.
3. **Data Migration**:
    a. Initiate the migration with Azure DMS:
        i. Create a migration project in Azure DMS.
        ii. Select the source (on-premises SQL Server) and target (Azure SQL Database/Managed Instance).
        iii. Choose the “Online” migration option to keep the source database available during the process.
    b. DMS will handle the schema and data migration, continuously syncing changes from the source to the target during the migration.
4. **Cutover**:

    a. After the initial data load, there will be a period of synchronization where DMS keeps the target database updated with changes made to the source.
    b. Once the data is in sync and you are ready to cut over, you can switch your application connections to the new Azure database.
    c. Perform any necessary validation to ensure data integrity and application functionality.
5. **Post-Migration**:
    a. Monitor the performance of the Azure database and make any needed adjustments.
    b. Decommission the on-premises database if no longer needed.

Benefits:

- **Minimal Downtime**: The online migration allows users to continue accessing the on-premises database during the process.
- **Ease of Use**: Azure DMS simplifies the migration process with its built-in tools and resources.

This approach will ensure a smooth transition to Azure with minimal disruption to your operations.