Advanced Azure Services Course for Intermediate Students

Module 1: Advanced Compute Solutions

- Deep dive into Azure Kubernetes Service (AKS), including cluster management, scaling, and networking.
- Advanced features of Azure Functions for serverless computing.
- Implementing Azure Container Instances for lightweight container deployment.

Module 2: Networking in Azure

- Advanced virtual networking concepts, including Azure ExpressRoute for dedicated connectivity.
- Implementing and managing Azure VPN Gateway and Azure Application Gateway.
- Network security groups and Application Security Groups (ASG) best practices.

Module 3: Azure Storage Solutions

- Implementing and managing Azure Blob Storage, including access tiers and lifecycle management.
- Advanced configuration of Azure File Sync and Azure Files for hybrid scenarios.
- Understanding and implementing Azure Data Lake Storage Gen2 features.

Module 4: Azure Database Services

- Advanced management and scaling strategies for Azure SQL Database.
- Implementing Azure Cosmos DB for global distribution and multi-model database solutions.
- Overview of Azure Database for MySQL, PostgreSQL, and MariaDB for open-source database management.

Module 5: Security and Identity

- Implementing and managing Azure Active Directory (Azure AD) for advanced scenarios.
- Azure Key Vault for secrets management and application security.
- Advanced threat protection with Azure Sentinel and Azure Security Center.

Module 6: DevOps and CI/CD

- Implementing Azure DevOps for continuous integration and continuous deployment (CI/CD) pipelines.
- Utilizing Azure Artifacts and Azure Repos for source control and artifact management.
- Best practices for DevOps culture and automation in Azure.

Hands-on Projects

Project: Deploying a Multi-Container Application with AKS

Project Overview:

This project focuses on deploying a multi-container application using Azure Kubernetes Service (AKS). The application will be a microservices-based architecture, where each microservice runs in its own container. Students will learn to configure autoscaling to manage the application's load efficiently and set up monitoring to maintain high availability and performance.

Objective:

Deploy a scalable and monitored multi-container application on AKS, demonstrating proficiency in managing containerized applications and Kubernetes resources on Azure.

Task 1: Setting Up AKS

Objective: Create and configure an AKS cluster to host the multi-container application.

Activities:

- Use the Azure CLI or Azure portal to create an AKS cluster with the required configuration (node size, count, etc.).
- Configure kubectl to connect to your AKS cluster.

Task 2: Containerizing the Application

Objective: Prepare the application for deployment by containerizing the microservices.

Activities:

- Write Dockerfiles for each microservice component of the application.
- Build Docker images and push them to Azure Container Registry (ACR).

Task 3: Deploying the Application to AKS

Objective: Deploy the containerized application to the AKS cluster.

Activities:

- Create Kubernetes deployment and service YAML files for each microservice.
- Deploy the application to AKS using kubectl, ensuring each microservice is correctly exposed and can communicate with others within the cluster.

Task 4: Implementing Autoscaling

Objective: Configure horizontal pod autoscaler (HPA) and cluster autoscaler for the AKS cluster.

Activities:

- Set up HPA to automatically scale the number of pods in a deployment based on observed CPU utilization or other selected metrics.

- Configure the cluster autoscaler to adjust the number of nodes in the cluster based on the needs of the workloads.

Task 5: Setting Up Monitoring and Alerts

Objective: Implement monitoring for the AKS cluster and the deployed application to ensure its health and performance.

Activities:

Implement security groups and IAM policies to control access to AWS resources.Use AWS Shield and AWS WAF to protect cloud resources from DDoS attacks and web exploits. Conduct a security assessment to identify potential vulnerabilities and apply necessary patches or configuration changes.

Task 6: Monitoring and Optimization

Objective: Set up monitoring for the hybrid cloud environment and optimize for performance and cost.

Activities:

- Enable Azure Monitor for containers to collect metrics and logs from the AKS cluster and containers.
- Set up dashboards in Azure Monitor to visualize key performance metrics.
- Configure alerts for critical conditions like high CPU/memory usage, pod failures, or service downtime.

Deliverables:

- Detailed architecture diagrams and deployment plans.
- Configuration files, scripts, and commands used throughout the project tasks.
- A monitoring and alerting setup with customized dashboards and alert

==

Project: Hybrid Cloud Connectivity with Azure ExpressRoute

Project Overview :

In this project, students will establish hybrid cloud connectivity using Azure ExpressRoute, connecting an on-premises network to Azure. This setup will simulate a real-world scenario where businesses require a dedicated, private connection between their data center and the cloud for improved bandwidth, reliability, and security.

Objective:

Create a secure and reliable hybrid cloud environment by establishing connectivity between an on-premises network and Azure using ExpressRoute.

Task 1: Planning and Design

Objective: Design a hybrid network architecture that integrates on-premises infrastructure with Azure services.

Activities:

- Assess the on-premises network setup and requirements for cloud connectivity.

- Plan the ExpressRoute circuit, including peering locations and bandwidth.

Task 2: Provisioning ExpressRoute

Objective: Provision an ExpressRoute circuit and establish a connection to Azure.

Activities:

- Use the Azure portal to create an ExpressRoute circuit and select a connectivity provider.
- Work with the connectivity provider to provision the physical connection.
- Configure peering for the ExpressRoute circuit, including Azure private peering and Microsoft peering as required.

Task 3: Configuring On-Premises Connectivity

Objective: Set up the on-premises network to connect to the ExpressRoute circuit.

Activities:

- Configure the on-premises router for BGP routing with Azure.
- Set up VPN fallback for ExpressRoute for additional resilience.

Task 4: Integrating Azure Services

Objective: Connect Azure services to the ExpressRoute circuit for seamless hybrid operations.

Activities:

- Link Virtual Networks to the ExpressRoute circuit for direct access from on-premises.
- Configure Azure services such as Azure VMs, Storage Accounts, and SQL Databases to utilize the ExpressRoute connection.

Task 5: Monitoring and Troubleshooting

Objective: Implement monitoring for the ExpressRoute connection and troubleshoot any connectivity issues.

Activities:

- Use Azure Network Watcher and ExpressRoute monitoring features to monitor the health and performance of the connection.
- Troubleshoot common connectivity issues, ensuring stable and reliable hybrid network performance.

Deliverables:

- Detailed architecture diagrams and deployment plans.
- Configuration files, scripts, and commands used throughout the project tasks.
- A monitoring and alerting setup with customized dashboards and alert.

Project: Implementing a Highly Available Database with Azure SQL

Project Overview :

This project aims to set up a highly available database solution using Azure SQL Database. Students will learn how to configure a high-availability architecture by leveraging geo-replication and failover groups in Azure SQL, ensuring database resilience and minimizing downtime during planned or unplanned outages.

Objective:

Design and implement a highly available database system using Azure SQL Database features like geo-replication and failover groups to ensure data availability and business continuity.

Task 1: Azure SQL Database Setup

Objective: Provision and configure an Azure SQL Database for high availability.

Activities:

- Create an Azure SQL Database, selecting the appropriate service tier and performance level to meet the application's requirements.
- Configure SQL Database settings, including data retention, security, and performance tuning options.

Task 2: Implementing Geo-Replication

Objective: Set up active geo-replication for the Azure SQL Database to replicate data across Azure regions.

Activities:

- Select additional Azure regions for data replication, considering factors like proximity to users and regional compliance requirements.
- Configure active geo-replication in the Azure portal, establishing up to four readable secondary databases in different regions.
- Test the replication setup by performing read and write operations on the primary database and validating the synchronization with secondary databases.

Task 3: Configuring Failover Groups

Objective: Create and manage failover groups to automate the failover process between the primary and secondary databases.

Activities:

- Define a failover group that includes the primary database and its geo-replicated secondaries.
- Configure automatic failover policies based on predefined rules or metrics, such as database availability or performance thresholds.
- Test the failover mechanism by simulating scenarios like region outages or database corruption, ensuring the automatic failover and failback processes work as expected.

Task 4: Monitoring and Maintenance

Objective: Set up monitoring for the high-availability architecture and perform regular maintenance tasks.

Activities:

- Implement Azure Monitor and Azure SQL Analytics to track the health, performance, and availability of the databases.
- Schedule regular reviews of the failover group configuration and geo-replication health to ensure the high-availability setup remains optimal.
- Document maintenance procedures, including performance tuning, security patching, and failover testing.

Deliverables:

- Configuration details and setup instructions for the Azure SQL Database, including geo-replication and failover groups.
- A testing report detailing the failover and failback process, including any issues encountered and their resolutions.
- A monitoring setup with custom dashboards and alerts for tracking database health and performance.
- Maintenance guidelines for managing the high-availability database system.

==

Project: Securing Cloud Applications with Azure Active Directory and Key Vault

Project Overview :

This project focuses on securing cloud applications by integrating Azure Active Directory (AAD) for authentication and Azure Key Vault for managing secrets and encryption keys. Students will implement authentication, authorization, and secure storage mechanisms to protect sensitive information and ensure secure access to cloud applications.

Objective:

Enhance the security of cloud applications using Azure Active Directory for user authentication and Azure Key Vault for secrets and encryption keys management, demonstrating best practices in cloud application security.

Task 1: Integrating Azure Active Directory

Objective: Implement Azure AD authentication for a cloud application, enabling secure user sign-in and access control.

Activities:

- Register the application in Azure AD to obtain an Application (client) ID and configure authentication settings.
- Implement OAuth 2.0 or OpenID Connect in the application to support Azure AD sign-in.

- Configure user roles and permissions in Azure AD for fine-grained access control within the application.

Task 2: Managing Secrets with Azure Key Vault

Objective: Utilize Azure Key Vault for storing application secrets and encryption keys securely.

Activities:

- Create an Azure Key Vault and configure access policies to restrict access to authorized Azure AD identities only.
- Store application secrets (e.g., connection strings, API keys) and encryption keys in the Key Vault.
- Modify the application to retrieve secrets and keys from Key Vault programmatically, ensuring sensitive information is not hard-coded in the application code.

Task 3: Implementing Encryption and Data Protection

Objective: Apply encryption to protect data in transit and at rest, using Azure Key Vault for key management.

Activities:

- Implement SSL/TLS for the application to secure data in transit.
- Use Azure Key Vault keys to encrypt sensitive data at rest, such as database files or blob storage.
- Configure and enforce HTTPS-only communication with Azure services.

Task 4: Monitoring and Compliance

Objective: Set up monitoring for security-related events and ensure compliance with security standards.

Activities:

- Utilize Azure Monitor and Azure Security Center to track security events and potential vulnerabilities.
- Set up alerts for unauthorized access attempts, secret access, or configuration changes in Azure Key Vault.
- Review and document the application's compliance with relevant security standards and regulations.

Deliverables:

- A detailed implementation guide for integrating Azure Active Directory and Azure Key Vault with a cloud application.
- Source code snippets or templates demonstrating the authentication flow and secure secrets retrieval.
- A security assessment report outlining encryption implementations, access controls, and compliance checks.
- Monitoring and alerting configuration details for tracking security events and ensuring the application's security posture.

Project: Implementing an Application with Azure AD Authentication and Key Vault for Secrets Management

Project Overview :

This project involves developing a cloud-based application that leverages Azure Active Directory (Azure AD) for secure user authentication and Azure Key Vault for managing application secrets. The goal is to demonstrate how to enhance application security by integrating Azure's identity management services and secrets management capabilities.

Objective:

Develop a secure application architecture that uses Azure AD for authentication and Azure Key Vault to store and access sensitive configuration information like database connection strings or API keys.

Task 1: Setting Up Azure AD for Application Authentication

Objective: Configure Azure AD to manage user identities and authentication for the application.

Activities:

- Create a new application registration in Azure AD to represent the application within the directory.
- Configure authentication settings, redirect URIs, and permissions required by the application.
- Implement OAuth 2.0 authentication flow in the application code to enable users to sign in using their Azure AD credentials.

Task 2: Securing Application Secrets with Azure Key Vault

Objective: Utilize Azure Key Vault for storing and accessing application secrets securely.

Activities:

- Provision a new Key Vault instance and configure access policies to allow the application to retrieve secrets.
- Store application secrets such as database connection strings, API keys, and certificates in Key Vault.
- Modify the application to retrieve these secrets from Key Vault at runtime rather than storing them in configuration files or source code.

Task 3: Integrating Azure AD and Key Vault with the Application

Objective: Ensure the application securely authenticates users with Azure AD and accesses secrets stored in Key Vault without exposing sensitive information.

Activities:

- Use Azure SDKs or REST APIs to integrate Azure AD authentication into the application's sign-in process.
- Implement code to authenticate with Azure Key Vault and securely retrieve secrets as needed during application runtime.
- Test the application to ensure that authentication flows work as expected and that secrets are accessed securely from Key Vault.

Task 4: Documentation and Best Practices

Objective: Document the setup process and highlight best practices for implementing security with Azure AD and Key Vault.

Activities:

- Create a detailed guide on setting up Azure AD authentication and Key Vault integration for applications.
- Document the steps taken to secure application secrets and manage user authentication.
- Outline best practices for managing identities and secrets in Azure to maintain a high security posture.

Deliverables:

- Application code integrating Azure AD for user authentication and Azure Key Vault for secrets management.
- ARM template or Azure CLI scripts used for provisioning Azure resources.
- Comprehensive documentation covering the setup of Azure AD and Key Vault, application integration steps, and security best practices.

==

Project: Automating Infrastructure Deployment with ARM Templates

Project Overview :

This project focuses on using Azure Resource Manager (ARM) templates to automate the deployment of a fully configured cloud environment in Azure. Students will gain hands-on experience in writing, deploying, and managing infrastructure as code, streamlining the provisioning process for complex environments.

Objective:

Master the creation and deployment of ARM templates to automate the setup of Azure resources, ensuring consistent and repeatable deployments across development, testing, and production environments.

Task 1: Plan the cloud infrastructure architecture that will be deployed using ARM templates.

Objective: Configure Amazon EKS to host the microservices application, ensuring it is secure, scalable, and highly available.

Activities:

- Identify the Azure resources required for the project (e.g., virtual networks, VMs, storage accounts, web apps).
- Define the resource relationships and dependencies to ensure proper deployment order.
- Sketch an architecture diagram to visualize the deployment.

Task 2: Writing ARM Templates

Objective: Write ARM templates to define the infrastructure as code.

Activities:

- Create ARM templates for each set of resources, utilizing parameters, variables, and resources blocks to define the infrastructure.
- Use linked templates for complex deployments, separating resources into modular templates for better manageability.
- Incorporate best practices for template development, including the use of parameters for environment-specific values.

Task 3: Deploying the Infrastructure

Objective: Deploy the infrastructure to Azure using the ARM templates.

Activities:

- Use Azure CLI, PowerShell, or the Azure portal to deploy the ARM templates to Azure.
- Validate the deployment in the Azure portal, ensuring all resources are correctly provisioned and configured.
- Test the connectivity and functionality of the deployed resources (e.g., accessing a deployed web application).

Task 4: Managing and Updating Deployments

Objective: Update and manage the infrastructure using ARM templates.

Activities:

- Make changes to the ARM templates to modify or add resources and redeploy.
- Implement deployment slots for web applications to manage updates and rollbacks.
- Use Azure Resource Manager to track deployment history and manage resource states.

Deliverables:

- A set of ARM templates that represent the infrastructure architecture.
- Scripts or commands used for deploying the templates.
- A detailed guide documenting the template creation process, deployment steps, and management practices.
- An architecture diagram representing the deployed infrastructure.