.NET Microservices – Azure DevOps and AKS

Section 12: Azure - Notes

Introduction to Cloud Computing

Cloud Computing is a technology that allows users to access and use computing resources over the internet. Instead of owning and maintaining physical hardware and software, users can leverage cloud services provided by cloud service providers (CSPs). This model offers flexibility, scalability, and cost efficiency.

Key Concepts:

1. **On-Demand Self-Service**:
   - Users can provision computing capabilities like server time and network storage as needed without requiring human interaction with each service provider.
2. **Broad Network Access**:
   - Cloud services are accessible over the network from various devices (e.g., smartphones, tablets, laptops) using standard protocols.
3. **Resource Pooling**:
   - Cloud providers pool resources to serve multiple customers, using a multi-tenant model where physical and virtual resources are assigned and reassigned according to demand.
4. **Rapid Elasticity**:
   - Cloud resources can be scaled up or down quickly based on demand. This elasticity allows for handling varying workloads efficiently.
5. **Measured Service**:
   - Cloud systems automatically control and optimize resource usage by leveraging metering capabilities. This ensures resources are paid for based on usage.

Types of Cloud Computing:

1. **Public Cloud**:
   - Services offered over the public internet. Examples: AWS, Azure, Google Cloud Platform.
   - Benefits: Cost-effective, scalable, and accessible from anywhere.

2. **Private Cloud**:
    - Cloud infrastructure is operated solely for one organization. Can be managed internally or by a third party.
    - Benefits: Enhanced security and control, customizable to organizational needs.
3. **Hybrid Cloud**:
    - Combines public and private clouds, allowing data and applications to be shared between them.
    - Benefits: Flexibility, cost efficiency, and enhanced security.
4. **Community Cloud**:
    - Shared infrastructure for a specific community of organizations with common concerns (e.g., security, compliance).
    - Benefits: Cost savings through shared resources.

Cloud Service Models

1. **Infrastructure as a Service (IaaS)**:
    - Provides virtualized computing resources over the internet.
    - Example: Azure Virtual Machines, AWS EC2.
    - Benefits: Scalability, flexible pricing, control over operating systems and applications.
2. **Platform as a Service (PaaS)**:
    - Provides a platform allowing customers to develop, run, and manage applications without dealing with the underlying infrastructure.
    - Example: Azure App Services, Google App Engine.
    - Benefits: Focus on application development, automatic updates, and scaling.
3. **Software as a Service (SaaS)**:
    - Delivers software applications over the internet, on a subscription basis.
    - Example: Office 365, Salesforce.
    - Benefits: No need for installation or maintenance, accessible from any device, automatic updates.

Introduction to Azure

Azure is Microsoft's cloud computing platform that provides a range of services, including those for computing, analytics, storage, and networking. It offers a robust set of tools and services for building, deploying, and managing applications through a global network of data centers.

Key Features of Azure:

1. **Compute Services**:
   - Virtual Machines, Azure App Services, Azure Kubernetes Service (AKS).
2. **Storage Services**:
   - Blob Storage, Table Storage, Disk Storage.
3. **Networking Services**:
   - Virtual Networks, Load Balancer, VPN Gateway.
4. **Database Services**:
   - Azure SQL Database, Cosmos DB, Azure Database for MySQL/PostgreSQL.
5. **Analytics and AI**:
   - Azure Machine Learning, Azure Synapse Analytics, Azure Data Lake.
6. **Management and Security**:
   - Azure Security Center, Azure Monitor, Azure Policy.

Problems / Limitations with Azure

1. **Complexity**:
   - The vast array of services and features can be overwhelming. It requires a steep learning curve to master and optimize usage.
2. **Cost Management**:
   - Pricing models can be complex, and costs can escalate if resources are not managed properly. Requires careful monitoring and budgeting.
3. **Service Limits**:
   - Some services have limits on resources or performance that can impact large-scale applications. Understanding these limits is crucial.
4. **Downtime and Reliability**:
   - Although Azure offers high availability, there can be occasional outages or disruptions. Relying on multiple regions and services can mitigate this.

5. **Compliance and Security**:
    - Ensuring compliance with regulations and managing security in a cloud environment requires diligent configuration and monitoring.

Azure CLI

Azure Command-Line Interface (CLI) is a cross-platform command-line tool that allows you to manage Azure resources from the command line. It provides a powerful and flexible way to interact with Azure services.

Key Commands:

1. **Login to Azure**:

```
az login
```

- Opens a browser window for authentication.

2. **List Resource Groups**:

```
az group list
```

- Displays all resource groups in the subscription.

3. **Create a Resource Group**:

```
az group create --name MyResourceGroup --location eastus
```

- Creates a new resource group named "MyResourceGroup" in the East US region.

4. **Deploy a Resource**:

```
az deployment group create --resource-group MyResourceGroup --template-file myTemplate.json
```

- Deploys resources defined in the specified template file to a resource group.

5. **List Containers in Azure Container Registry**:

```
az acr repository list --name MyRegistry --output table
```

- Lists all repositories in the specified Azure Container Registry.

Azure Resource Groups (using CLI)

Resource Groups in Azure are containers that hold related resources for an application. They allow for resource management, access control, and billing.

Common CLI Commands:

1. **Create a Resource Group**:

```
az group create --name MyResourceGroup --location eastus
```

   - Creates a resource group named "MyResourceGroup" in the East US region.

2. **Delete a Resource Group**:

```
az group delete --name MyResourceGroup --yes --no-wait
```

   - Deletes the resource group "MyResourceGroup" without confirmation.

3. **List Resource Groups**:

```
az group list --output table
```

   - Lists all resource groups in the subscription in a tabular format.

4. **Show Resource Group Details**:

```
az group show --name MyResourceGroup
```

   - Displays details of the specified resource group.

5. **Update Resource Group Tags**:

```
az group update --name MyResourceGroup --tags env=production
```

   - Adds or updates tags for the specified resource group.

Azure Container Registry (using CLI)

Azure Container Registry (ACR) is a private Docker registry service in Azure that allows you to store and manage Docker container images and artifacts.

Common CLI Commands:

1. **Create a Container Registry**:

```
az acr create --resource-group MyResourceGroup --name MyRegistry --sku Basic
```

   - Creates a new container registry named "MyRegistry" with Basic SKU in the specified resource group.

2. **List Container Registries**:

```
az acr list --resource-group MyResourceGroup --output table
```

   - Lists all container registries in the specified resource group in a tabular format.

3. **Show Container Registry Details**:

```
az acr show --name MyRegistry
```

   - Displays details of the specified container registry.

4. **Delete a Container Registry**:

```
az acr delete --name MyRegistry --resource-group MyResourceGroup --yes
```

   - Deletes the specified container registry from the resource group.

5. **Login to Container Registry**:

```
az acr login --name MyRegistry
```

   - Logs in to the specified container registry to push or pull images.

Limitations of AppService and Azure Container Apps to Host Complex Microservices

Azure App Service and **Azure Container Apps** are effective for many scenarios but may not be ideal for complex microservices applications. Here's why:

1. **Scaling Limitations**:
   - App Services may struggle with scaling to handle very high loads or complex workloads compared to AKS.

2. **Limited Customization**:
    - App Services offer limited control over the underlying infrastructure and configurations, which might be restrictive for complex applications.
3. **Performance Constraints**:
    - App Services and Azure Container Apps may not provide the performance and fine-grained control required for high-performance microservices architectures.
4. **Resource Constraints**:
    - Resource limits on App Services and Azure Container Apps might not be sufficient for large-scale or resource-intensive applications.
5. **Complex Orchestration Needs**:
    - For applications requiring sophisticated orchestration, service discovery, and distributed data management, Azure Kubernetes Service (AKS) offers more robust solutions.

AppService (using CLI - with Docker Container)

Azure App Service is a fully managed platform for building, deploying, and scaling web apps. It supports Docker containers and integrates with various deployment options.

Common CLI Commands:

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

```
az appservice plan create --name MyAppServicePlan --resource-group MyResourceGroup --sku B1 --is-linux
```

- Creates a new App Service Plan named "MyAppServicePlan" with Basic pricing tier and Linux environment.

2. **Create a Web App**:

```
az webapp create --resource-group MyResourceGroup --plan MyAppServicePlan --name MyWebApp --deployment-container
```

Azure Container Apps (using CLI)

Azure Container Apps is a serverless container service that enables you to deploy and manage containerized applications in a fully managed environment. It is designed for ease of use, providing automatic scaling and built-in support for microservices architectures.

Common CLI Commands:

1. **Create an Azure Container App Environment**:

```
az containerapp env create --name MyContainerAppEnv --resource-group MyResourceGroup --location eastus
```

   - Creates a new environment for Azure Container Apps named "MyContainerAppEnv" in the specified resource group and region.

2. **Create an Azure Container App**:

```
az containerapp create --name MyContainerApp --resource-group MyResourceGroup --environment MyContainerAppEnv --image mycontainerimage:v1 --cpu 0.5 --memory 1.0Gi --target-port 80
```

   - Deploys a container app named "MyContainerApp" using the specified container image, with allocated CPU and memory resources.

3. **Update an Azure Container App**:

```
az containerapp update --name MyContainerApp --resource-group MyResourceGroup --image mycontainerimage:v2
```

   - Updates the container image for an existing container app.

4. **List Azure Container Apps**:

```
az containerapp list --resource-group MyResourceGroup --output table
```

   - Lists all container apps in the specified resource group in a tabular format.

5. **Show Azure Container App Details**:

```
az containerapp show --name MyContainerApp --resource-group MyResourceGroup
```

   - Displays details of the specified container app.

6. **Delete an Azure Container App**:

```
az containerapp delete --name MyContainerApp --resource-group MyResourceGroup --yes
```

   - Deletes the specified container app from the resource group.

Key Points to Remember (for Interview Preparation)

1. **Cloud Computing Concepts**:
   - Understand the fundamental characteristics of cloud computing (on-demand self-service, broad network access, resource pooling, rapid elasticity, measured service).
2. **Cloud Service Models**:
   - Know the differences between IaaS, PaaS, and SaaS, and when to use each model based on project requirements.
3. **Azure Features and Benefits**:
   - Familiarize yourself with core Azure services like Virtual Machines, App Services, AKS, and Azure Container Registry. Understand their benefits and use cases.
4. **Azure CLI Basics**:
   - Be comfortable with basic Azure CLI commands for resource management, including creating, listing, and deleting resources.
5. **Azure Resource Management**:
   - Understand how to work with Azure Resource Groups for organizing and managing resources.
6. **Container Management**:
   - Know the basic commands for managing Azure Container Registry and deploying containers using Azure App Service and Azure Container Apps.
7. **App Service vs. AKS**:
   - Be able to articulate the limitations of Azure App Service and Azure Container Apps in hosting complex microservices and why AKS might be a better choice for large-scale projects.
8. **Limitations and Challenges**:
   - Be aware of common problems and limitations associated with Azure services, such as complexity, cost management, and resource constraints.