**SUMMARY-DAY23**

**Name:** Tejaswini Gokanakonda

**Roll no:**DE142

**Date:**13-12-2024

-------------------------------------------------------------------------------------------------------------------------------------

**Plan a Release Pipeline Using Azure Pipelines**

**Overview**

A release pipeline automates application deployment in stages, ensuring reliable, repeatable, and secure delivery. **Azure Pipelines** supports Continuous Delivery (CD) pipelines with robust features for deployment across environments like VMs, containers, or cloud platform

**1. Parts of a Basic CD Pipeline**

* **Trigger**: Initiates the pipeline (e.g., code commit, manual, schedule).
* **Stages**: Logical units containing one or more **jobs**.
  + Example:
    - Build stage (creates artifacts).
    - Deployment stage (deploys artifacts to environments).
* **Jobs**: Sequential tasks within a stage.
  + Deployment jobs offer enhanced tracking and audit trails.

**2. Understanding Key Components**

**Pipeline Stage**

* A logical grouping of tasks within a pipeline.
* Can be triggered by:
  + Success of a prior stage.
  + Manual intervention.
  + Time-based schedules.

**Environment**

* **Definition**: The target location for deployment.
  + Examples:
    - On-premises (VMs, physical servers).
    - Cloud-based (Azure App Service, Kubernetes).
    - Serverless (Azure Functions).
  + Tracks deployment history and facilitates controlled artifact promotion.
* **Azure Pipelines Environment**: Abstract representation of a deployment environment.

**Artifacts**

* The output from the build stage, such as .zip or .jar files.
* Deployed to environments in the pipeline.

**3. Choosing a Deployment Environment**

**Options:**

1. **Virtual Machines (VMs)**: Full control of the OS and environment. Best for legacy or specialized applications.
2. **Containers**: Lightweight and portable. Ideal for microservices or scalable solutions.
3. **Azure App Service**: A PaaS offering for web applications.
   * Simplifies infrastructure management.
   * Provides security features, automatic scaling, and load balancing.
4. **Serverless (Azure Functions)**: Best for event-driven, cost-efficient computing.

**Recommended Choice:**

**Azure App Service** for simplicity, scalability, and seamless integration with Azure DevOps.

**4. Deployment Workflow in Azure Pipelines**

**Authentication with Target Environment**

1. **Service Connection**: Secures access to Azure resources.
   * **Service Principal**: Limited-role identity for tasks.
   * **Managed Identities**: Simplifies authentication via Microsoft Entra ID.
2. **Azure Pipeline Configuration**:
   * Define target environment in YAML.
   * Tasks used:
     + DownloadPipelineArtifact: Fetches build artifacts.
     + AzureWebApp: Deploys applications to Azure App Service.

**Pipeline Stages Example**

* YAML Configuration:

- stage: 'DeployDev'

displayName: 'Deploy to dev environment'

dependsOn: Build

jobs:

- deployment: Deploy

pool:

vmImage: 'ubuntu-20.04'

environment: dev

strategy:

runOnce:

deploy:

steps:

- download: current

artifact: drop

- task: AzureWebApp@1

displayName: 'Azure App Service Deploy: website'

**5. Jobs and Strategies**

**Jobs**

* Sequential tasks in a stage.
* Can run:
  + In agent pools.
  + On containers.
  + Directly on Azure DevOps servers.
* Deployment jobs provide:
  + Audit trails.
  + Deployment status tracking.

**Strategies**

* Defines how applications are rolled out.
* Common strategies:
  + **RunOnce**: Executes deployment once (default for simple setups).
  + **Blue-Green**: Deploys to a secondary environment before switching traffic.
  + **Canary**: Gradual rollout to subsets of users.

**6. Steps to Plan a Deployment**

1. **Build Stage**:
   * Configures the pipeline to create artifacts.
2. **Deployment Stage**:
   * Downloads artifacts using DownloadPipelineArtifact.
   * Deploys them to the chosen environment (e.g., Azure App Service).
   * Uses service connections for authentication.

**7. Key Benefits of Azure Pipelines**

* **Scalability**: Supports parallel and sequential job execution.
* **Security**: Offers role-based access and secure service connections.
* **Flexibility**: Integrates with various deployment environments (cloud, on-premises).
* **Ease of Use**: YAML-based configuration simplifies complex workflows.

**8. Example Workflow**

1. **Input**: Build artifact from the existing build pipeline.
2. **Process**:
   * Download artifact in the deployment stage.
   * Authenticate with Azure App Service.
   * Deploy application using predefined tasks.
3. **Output**: Application deployed and ready for use.