**Final DevOps Coding Assessment(12-01-2024)**

**Name:** Aparna Kadiyala

**Email:** kadiyalaaparna2001@gmail.com

**Task -1 :** Explain process of Azure DevOps CI/CD  pipeline

**Task -2:** Explain azure SQL server pool integration with azure synapse.

**Task-1: Explain process of Azure DevOps CI/CD  pipeline**

**Azure DevOps pipeline**

* Basically Azure DevOps has a special tool called pipeline
* Pipeline is a helps automate the steps when making and releasing software. In that we have CI/CD pipelines
* CI stands for Continuous Integration and CD stands for Continuous Delivery.

**CI/CD pipeline**

The main purpose of Azure DevOps CI/CD pipelines is to automate the software development lifecycle, from code changes to deployment, in a repeatable and reliable way also. There are two main goals.

**Speed up software delivery:**

* Actually the pipelines would automate build, test, and deployment tasks, allowing us to release new features and updates to users faster and more frequently.
* This translates to quicker time to market, improved responsiveness to user needs, and increased competitive advantage.

**Improve software quality:**

* These automated testing embedded within the pipeline catches bugs and potential problems early in the development process, preventing them from reaching production and impacting users.
* Perhaps, this leads to more reliable and stable applications, with fewer bug fixes needed later.

Now let us know **what is actually CI/CD process**. Here we have common steps that what we actually do with the CI/CD pipeline.

**1. Code Commit:**

Developers push their code changes to a shared repository, like Azure Repos. This triggers the CI/CD pipeline.

**2. Continuous Integration (CI):**

*Build*: The pipeline builds the code, meaning it compiles it into its final executable form. This ensures the code compiles correctly and catches any syntax errors early.

*Test:* Automated tests are run against the built code. These tests check for functional and non-functional aspects like performance and security.

*Package:* If all tests pass, the pipeline prepares the build outputs for deployment that only known as artifacts.

**3. Continuous Delivery (CD):**

*Deployment:* Artifacts are automatically deployed to chosen environments, typically starting with testing environments and progressing to production environments.

*Monitoring:* The deployed application is monitored for stability and performance issues.

Feedback: Feedback from monitoring and user testing feeds back into the CI/CD pipeline, enabling continuous improvement.

**Pipeline Stages:**

*CI Pipelines:* The pipelines will run frequently with every code change, focusing on building, testing, and packaging.

*Release Pipelines*: Those would triggered manually or by specific conditions, handle deploying artifacts to different environments in controlled stages and continuous Integration(CI).

**Now let us discuss the implementation process**

**1. Plan and Define:**

*Identify goals:* we need to deefine what we want to achieve with CI/CD, such as faster releases, improved quality, or better collaboration.

*Map out the process:* Then outline the steps involved in your development and deployment workflow, including build, test, and deployment phases.

*Choose tools*: Then decide on the CI/CD tools you'll use, such as Azure Pipelines, Azure Repos, and Azure Artifacts.

Set up infrastructure: Provision necessary resources like build agents, deployment targets, and testing environments.

**2. Create CI Pipeline:**

*Connect to repository:* We have to link your Azure DevOps project to our code repository.

*Define build steps:* Then specify how to compile and build our code, including dependencies and tools required for the process.

*Add test tasks:* Then integrate automated tests to run against the built code, covering unit, integration, and other types of tests.

*Package artifacts:* Finally ,prepare the build outputs for deployment, creating packages or container images.

**3. Create CD Pipeline:**

*Choose deployment environments:* we need to define the environments where we will deploy your application, such as testing, staging, and production.

*Set up release triggers:* Then determine when to trigger deployments, either manually or automatically based on conditions.

*Define deployment tasks:* Specify the steps to deploy artifacts to different environments, including configuration changes and infrastructure provisioning.

*Add approval gates:* Implement manual approval steps if required for certain environments or releases.

**4. Integrate with Other Tools:**

*Connect testing tools:* Integrate with testing tools for more comprehensive testing, performance analysis, and security scans.

*Integrate monitoring tools:* Add monitoring tools to track application health and performance in production.

*Integrate with infrastructure management tools*: Connect with tools for provisioning and managing infrastructure resources.

**5. Test and Refine:**

*Test the pipeline:* Run the pipeline with test code changes to ensure it works as expected and identify any issues.

*Review and refine:* Iterate on the pipeline configuration to optimize performance, add new tests, or adjust deployment strategies based on feedback and results.

*Monitor and maintain:* Continuously monitor pipeline performance and make adjustments as needed to ensure it remains reliable and efficient.

**Task -2: Explain azure SQL server pool integration with azure synapse.**

**Azure SQL server pool :** Azure SQL Server pool is like a high-performance swimming pool for your data, where you can store, manage, and access it with power and flexibility.

**Azure Synapse:** It is defined as the Azure Synapse is a limitless analytics service that brings together data integration, data warehousing, and big data analytics in a unified platform.

**Use of Integration of Azure SQL server pool with Azure Synapse:**

**Analyse operational data in real-time or near-real-time:** we need to access and analyse data from our Azure SQL databases directly within Azure Synapse, without the need for complex data movement or any ETL processes.

**Run analytics on both structured and unstructured data:** Then we need tocombine structured data from Azure SQL with unstructured data from other sources (like data lakes) for comprehensive insights.

**Build end-to-end analytics solutions:** Integrate data preparation, cleaning, transformation, and analysis tasks within a single platform, streamlining your analytics workflow.

**Key integration methods:**

**1.Azure Synapse Link for SQL:**

* Seamless integration between Azure SQL Database or SQL Server 2022 with Azure Synapse Analytics dedicated SQL pools.
* Automatically replicates data changes with minimal impact on source databases.
* Enables near-real-time analytics on operational data.

**2.Linked Services:**

* Connect Azure Synapse to external Azure SQL databases or SQL Server instances using linked services.
* Query and process data directly within Synapse pipelines and notebooks.
* Supports various authentication methods for secure access.

**3.Copy Activities in Azure Data Factory:**

* Move data between Azure SQL databases and Azure Synapse using data pipelines in Data Factory.
* Orchestrate complex data flows and transformations.
* Integrate with other data sources and destinations as needed.

***Now let us know how can we implement them as together.***

**Integration Purpose:** Analyse operational data in real-time or near-real-time, blend structured and unstructured data, and build end-to-end analytics solutions within a unified platform.

We already know about the **Key Methods** as follows:

1. Azure Synapse Link for SQL that means near-real-time replication for operational analytics.
2. Linked Services just like connect to external Azure SQL databases.
3. Copy Activities in Azure Data Factory means move data for complex transformations.

**Implementation Steps:**

**Step-1: Define Requirements**

* Workload types (we have different types of workloads like transactional, data warehouse, mixed)
* Performance needs (query response time, throughput)
* Scalability expectations
* Data security and compliance requirements

**Steo-2 : Choose Pool Type**

* General Purpose (balanced performance for most workloads it will be based on situation )
* Business Critical (highest performance and availability)
* Hyperscale (extremely large datasets and data warehousing)

**Step-3: Create Pool in Azure Portal or CLI/API**

* Specify pool type, region, performance tier
* Define storage space and automatic scaling settings
* Configure security features (logins, roles, firewall rules)

**Step-4: Migrate Databases**

* Use built-in tools for simple migrations from on-premises SQL Server or Azure SQL Database
* Use Azure Data Factory pipelines for complex data movement and transformations

**Step-5: Connect and Use the Pool**

* Access and manage databases using familiar SQL Server tools and applications
* Integrate with Azure Synapse Analytics for analytics and data warehousing
* Monitor pool performance and adjust configurations

**Step-6: Integrate with Azure Synapse**

* Choose the integration method (Synapse Link, Linked Services, Data Factory) based on needs
* Configure connections and authentication
* Query and process data directly within Synapse pipelines and notebooks

The above mentioned steps we will use when we are integrating with the azure SQL server pool integration with azure synapse.