**Implementing Continuous Integration and Continuous Delivery – Azure DevOps**

1. Created an Organization and Project in Azure DevOps:

A screenshot of a computer

AI-generated content may be incorrect.

1. Replicated Realtime scenario by Three more environments as in Resource Groups Dev, Test and Production.

Dev:

A screenshot of a computer

AI-generated content may be incorrect.

Test:

A screenshot of a computer

AI-generated content may be incorrect.

Prod:

A screenshot of a computer

AI-generated content may be incorrect.

1. Mounted Azure DevOps on dev data factory, resulted in creating two branches i.e., Main or Collaborative branch and ADF publish branch upon first publish.

A screenshot of a computer

AI-generated content may be incorrect.

**Build pipeline:**

1. Created a DevOps build pipeline by using a YAML script by Microsoft to automatically download a NPM package.
2. Once the NPM package is downloaded, it initiates series of action that publish the changes to dev environment and generates the ARM templates as artifacts.
3. These artifacts then moved to the adf\_publish branch automatically.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

**Release pipeline:**

1. Once the ARM template has been released to the ADF publish branch, the release pipeline automatically fetches the ARM.
2. These ARM templates are then deployed continuously to test and production environments without an intervention.
3. The release has two artifacts:

**A screenshot of a computer

AI-generated content may be incorrect.**

The first artifact points to the build pipeline which generates ARM templates based on the commits to the main branch. This trigger has been automated by enabling continuous delivery option.

A screenshot of a computer

AI-generated content may be incorrect.

The second artifact points to the main branch which commits the changes to main branch.

**Stages in Release pipeline:**

**A screenshot of a computer

AI-generated content may be incorrect.**

A screenshot of a computer

AI-generated content may be incorrect.

The pipeline variables are declared at the pipeline level which holds values like location, resource group name and datalake name.

These variables are then dynamically invoked in each activities within each activity of the respective stages.

**Dev/Test/Prod:**

**A screenshot of a computer program

AI-generated content may be incorrect.**

1. Each stage within each stages such dev, test and prod have three Agent job / Activities.

**First Job:**

**Azure PowerShell Script: Pre-deployment stage:**

1. One of the drawbacks of continuous integration and continuous delivery is that ADF does not allow alter or update triggers that are active. Resulting in failures.
2. To overcome this failure, Microsoft offers predefined shell script which turn offs any active triggers and updates it and reactivate the trigger after deployment.
3. The pre-deployment script follows the same. It deactivates any active triggers before deployment in each stage.

A screenshot of a computer

AI-generated content may be incorrect.

1. Script arguments have to passed properly pointing to respective environments resource group name and ADF name, this is where the variables are used effectively.

**Second Job:**

**Arm deployment Job:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

1. This stage is responsible for deploying ARM templates which was published by ADF publish branch by Build pipeline to take care.
2. The template and template parameters are pointed properly which is in the main branch.
3. “Override template parameters” - -factoryName "$(dataFactoryName)" -ls\_storageAccount\_Conn\_properties\_typeProperties\_url <https://$(dataLakename).dfs.core.windows.net/>
4. The above option needs to be mandatorily changed and update the environment config within the ARM template using variables
5. We also used managed identity and IAM access control to create a service principal to make it work across each environment.

**Third Job:**

**Post Deployment Job:**

1. This job responsible to activate any triggers which are deactivated by the pre-deployment script.

A screenshot of a computer

AI-generated content may be incorrect.