Assignment(Azure Devops)

Name-Ajay Chaudhary
Batch-Data Engineering(Batch-1)

What is CI/CD?

A CI/CD pipeline is a concept central to software. It spans a whole field of processes, testing methods, and tooling, all facilitated by the Git code versioning process.
Since the terms “CI/CD Pipeline” and “Data Pipeline” are confusing, we will simply refer here to the CI/CD process.

***Continuous integration*:** Every time you add a new track piece, you immediately test it by running the data through it. This ensures that your new addition didn't introduce any problems. If there's an issue, you know instantly and can fix it before it becomes a bigger problem.

- Continuous Integration is the practice of frequently integrating code changes into a shared repository.

- In Azure DevOps, Azure Pipelines facilitates CI by automatically building and testing code every time a change is committed to the repository.

***Continuous deployment*:** Once you've confirmed that your new piece fits and the data runs smoothly, you don't wait to show it off. You immediately let everyone see and use the updated track. In other words, as soon as your changes are verified, they're made live and functional in the main track (production environment).

- Continuous Deployment is the practice of automatically deploying code changes to production or other environments after passing the CI process.

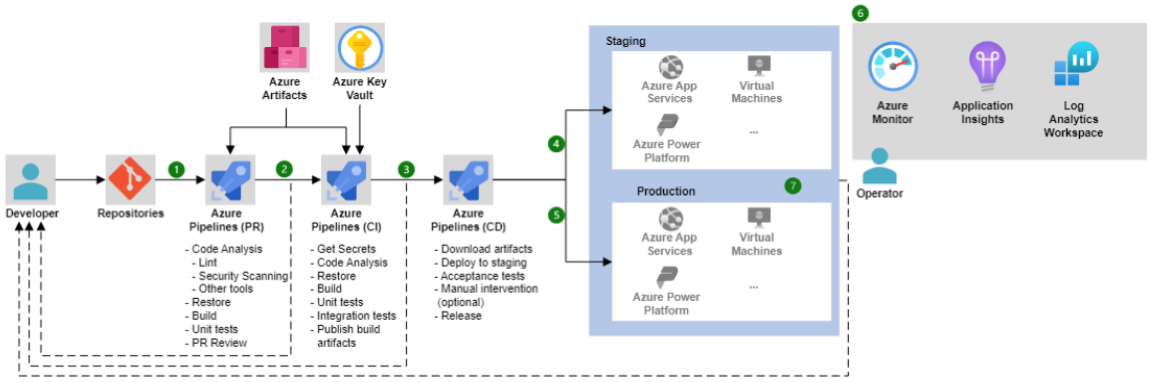
- Azure DevOps supports CD through Azure Pipelines, enabling automated deployment of data engineering artifacts, such as ETL jobs, SQL scripts, or machine learning models.

Azure Pipelines:

- Azure Pipelines is a cloud-based service for building, testing, and deploying code across different platforms and languages.

- It supports both CI and CD workflows and allows you to define pipelines using YAML or the visual designer.

- Pipelines can include multiple stages, jobs, and tasks to automate various aspects of the development process, including data engineering tasks like data validation, transformation, and deployment to target data stores.



Key Concepts in Azure Pipelines:

- Pipeline: Defines the entire CI/CD process, including stages, jobs, and tasks.

- Stage: Represents a logical boundary within the pipeline, such as Build, Test, or Deploy.

- Job: Defines a set of tasks that run sequentially or in parallel within a stage.

- Task: Represents a single action within a job, such as executing a script, running a test suite, or deploying an artifact.

Integration with Data Engineering Tools:

- Azure Pipelines integrates seamlessly with various data engineering tools and technologies commonly used in Azure ecosystem, such as Azure Data Factory, Azure Databricks, Azure Synapse Analytics, and Azure SQL Database.

- Integration may involve running scripts, executing commands, deploying packages, or triggering workflows in these services as part of the CI/CD process.

Monitoring and Reporting:

- Azure DevOps provides monitoring and reporting capabilities to track the progress and health of CI/CD pipelines.

- You can monitor pipeline runs, view build and release logs, analyze test results, and generate reports to identify issues and optimize performance.

Security and Compliance:

- Azure DevOps includes features for ensuring security and compliance in CI/CD processes, such as role-based access control (RBAC), encryption, audit logs, and compliance certifications (e.g., SOC, ISO).

- It also supports integration with Azure Key Vault for securely managing secrets and credentials used in pipelines.

By leveraging Azure DevOps for CI/CD in data engineering projects, teams can automate the deployment of data pipelines, maintain consistency across environments, and accelerate the delivery of data-driven solutions while ensuring reliability and quality.

ETL pipelines

ETL (Extract, Transform, Load) pipelines are at the heart of data engineering. They're the processes that pull data from sources (databases, APIs, etc.), transform it into a usable format, and then load it into a destination, like databases or a data warehouse. When you deploy an ETL script to Git, you're not just saving the code—you could be triggering a series of events:

1. **Testing**: Automated tests are first run to ensure the new code doesn't break anything
2. **Deployment**: Once tests pass, the ETL process can be automatically deployed to a staging or production environment
3. **Notifications**: If any part of the process fails, or if it's successfully completed, notifications can be sent out