**Day 24 Notes**

**Name: Podutur Lahari - DE126**

**Date:14-12-2024**

**Azure Databricks Notebooks Version Control and Deployment Pipelines**

**Version Control with Azure DevOps**

* **Create a Git Repository**: Start by creating a Git repository in Azure DevOps to store all your Databricks notebooks, enabling version control and team collaboration.
* **Export Notebooks**: Export Databricks notebooks into compatible formats such as .dbc, .html, or .ipynb. These exported files can be version-controlled in Git, allowing you to track changes.
* **Push to Azure DevOps**: After exporting, commit and push the notebooks to the Azure DevOps repository. This enables easy tracking of updates and history.
* **Import Back to Databricks**: When needed, you can import notebooks back into Databricks from the repository, ensuring synchronization between Databricks and your version control system.

**Building and Automating Deployment Pipelines**

* **Build Pipeline**: Set up a build pipeline in Azure DevOps to automate the process of exporting notebooks, running tests (if applicable), and preparing them for deployment. The build pipeline triggers when changes are made to the repository.
* **Release Pipeline**: A release pipeline is used to deploy notebooks from the version-controlled repository to different Databricks workspaces (e.g., development, staging, or production). The release pipeline automates deployment tasks and ensures the latest changes are applied.
* **Triggers and Approvals**: Pipelines can be set to trigger automatically when changes occur, or you can configure manual approval steps before deploying to production environments.

**Best Practices for Version Control and Deployment**

* **Branching Strategy**: Implement a branching strategy like GitFlow, where features, bug fixes, and releases are managed in separate branches, enabling parallel work without conflicts.
* **Versioning**: Use semantic versioning by tagging releases or adding version numbers to notebooks to track significant updates and changes.
* **Modularize Notebooks**: Break down large notebooks into smaller, reusable modules or scripts to make them easier to manage and collaborate on.
* **Secure Sensitive Data**: Avoid storing sensitive information like API keys directly in notebooks. Instead, use Azure Key Vault or Databricks Secrets to securely manage credentials and secrets.

**Benefits of Version Control and CI/CD Pipelines**

* **Collaboration**: Version control allows multiple team members to work on the same project without overwriting each other's changes. It also provides a clear history of modifications.
* **Traceability**: You can easily trace the history of changes made to notebooks, understand who made the changes, and what was changed, improving debugging and troubleshooting.
* **Automation**: CI/CD pipelines automate the processes of exporting, testing, and deploying notebooks, reducing manual effort and minimizing the risk of errors.
* **Scalability**: As your project grows, automated pipelines help scale deployments across different environments without adding significant manual effort.

**Conclusion**

By integrating Azure Databricks notebooks with Azure DevOps for version control and deploying them through automated CI/CD pipelines, you streamline collaboration, ensure traceability, and automate deployment processes. This approach improves project management, scalability, and overall efficiency in managing Databricks notebooks in a production environment.