**CI/CD Pipeline Strategy for Microsoft Fabric Deployment**

**Overview**

We have adopted a structured, scalable, and secure approach for Continuous Integration and Continuous Deployment (CI/CD) using Git and Azure DevOps, specifically tailored for Microsoft Fabric deployments. This strategy facilitates the automated deployment of Data Pipelines, Notebooks, Copy Jobs, Lakehouses, and Shortcuts.  
  
**Version Control with Git (Azure Repos)**

* **Centralized Repositories:** Organized and version-controlled in Azure Repos with clear separation:

A screen shot of a computer

AI-generated content may be incorrect.**Branching Strategy:**

* + Main branch for stable releases
  + Feature branches for development

Visual Representation:  
  
A diagram of a software development process

AI-generated content may be incorrect.

**CI/CD Pipeline Architecture  
  
A diagram of a workflow

AI-generated content may be incorrect.**

* Commit / Merge – Developer merges a pull request into develop; Azure DevOps instantly queues the pipeline.
* CI Validation – Pipeline spins up a clean agent, installs fabric‑cicd, and runs lightweight syntax/standards checks. *Goal:* fail fast if code is malformed.
* Deploy to Test – Using secrets stored in Azure DevOps, the pipeline authenticates as a Service Principal and publishes every artifact to the Test workspace. Parameter file swaps Dev IDs for Test IDs automatically.
* Automated Tests (optional) – A sample notebook or data pipeline executes to confirm the new content runs successfully in Test. Results posted back to the pipeline log.
* Manual Approval – Release is paused until an approver (e.g. Team Lead or Release Mgr) clicks *Approve* in the Azure DevOps environment gate.
* Deploy to Prod – Pipeline re‑runs the publish step, this time targeting the Production workspace and injecting Prod‑specific parameters. Same artifacts, different config.
* Notify & Monitor – Success/failure message sent to Teams; Azure DevOps dashboard shows run history for audit.