Azure Databricks CI/CD Pipeline - Step by Step Runbook

1) Create Azure resource groups

Make two RGs: one for Dev and one for Prod (unique, meaningful names).

2) Create Databricks workspaces

In each RG, create an Azure Databricks workspace (pick “Standard” tier). Wait for deploy and open the resource.

3) In Dev Databricks: prep compute & notebooks

Launch workspace → Compute → create a (cost-saving) single-node cluster.

Upload your notebooks (Bronze, Silver, Gold) that you want to move through CI/CD.

4) Wire up source control & branch policy

Clone/link your Azure Repos repo in Databricks Repos.

Protect main: try a direct push to main to confirm policy blocks it, then work via a feature branch → PR → approve → complete merge.

5) Create CI/CD folder structure in the repo

Under the repo, add a CICD/ folder with:

- scripts/DatabricksToken.ps1
- templates/deploy-notebooks.yml
- cicd-pipelines.yml

Import/upload those files into the repo (from local).

6) Configure the token script (Dev workspace)

Open the Dev Databricks workspace in Azure portal → Properties → Managed Identity → copy Object ID (Principal ID).

Put that value into DatabricksToken.ps1 so the pipeline can create a bearer token.

7) Build the CI pipeline (Dev)

In Azure DevOps → Pipelines → Library → create a variable group for Dev with:

- environmentName = $(dev-environment-name)
- resourceGroupName = $(dev-resource-group-name)
- serviceConnection = $(dev-service-connection-name)

Create the Dev environment (Pipelines → Environments) and use its exact name as the value for environmentName.

Create a Service Connection (Project Settings → Service connections → Azure Resource Manager; credentials: Workload Identity; scope it to the Dev RG). Use the SC name as the value for serviceConnection.

8) Grant the CI pipeline permissions

Give the pipeline permission to use: the variable group, the environment, and the service connection (Library/Env/Service connection → Security/Pipeline permissions → add your CI pipeline).

9) Author the CI YAML

In cicd-pipelines.yml, set trigger on main, reference the Dev variable group, and call the templates/deploy-notebooks.yml.

The CI run creates a /live folder in the Dev workspace and uploads the latest notebooks there (single source of truth for deployments).

10) Restructure/push via feature branch

Commit the new folders/files (CICD, templates, scripts) using a feature branch, then create a PR and merge to main. This merge will trigger CI.

11) Build the CD (Prod) layer

Create a Prod variable group with:

- environmentName = $(prod-environment-name)

- resourceGroupName = $(prod-resource-group-name)

- serviceConnection = $(prod-service-connection-name)

Create the Prod environment and a Prod service connection (same steps as Dev, but scoped to Prod). Give the CI pipeline permissions to use these as well.

12) Update pipeline for Prod deployment

In cicd-pipelines.yml, add the Prod variable group (e.g., - group: dbw-cicd-prod) and a new stage like Deploy_to_Prod_Environment that depends on the CI stage.

This makes CD auto-trigger after a successful CI run and deploy the notebooks from Dev /live to the Prod workspace.

13) Validate E2E

Make a small change in a feature branch → PR → merge to main.

Confirm CI creates/updates /live in Dev, then CD pushes those notebooks to Prod automatically.