WEEK-5:PIPELINE AUTOMATIONS WITH AZURE DEVOPS

1) Pre-requisites (quick checklist)

- Azure subscription + resource group (or access to one).
- Azure DevOps organization + project (create if needed).
- Repo reachable from Azure DevOps (Azure Repos or GitHub).
- A Service Principal (SP) for deployments (or let Azure DevOps create one automatically).
- Permissions: you must be Project Admin (to create service connections, variable groups, environments).

Steps to Create CI & CD Pipelines in Azure DevOps

Part 1: Create CI Pipeline (Continuous Integration)

This pipeline will **build, test, and generate artifacts** automatically when you push code to your Git repo.

Step 1 — Go to Azure DevOps

- Open **https://dev.azure.com**
- Select your **organization** and **project**.

Step 2 — Create a New Pipeline

1. Go to **Pipelines → Pipelines →** Click **New Pipeline**.
2. Choose **Azure Repos Git** (or **GitHub** if your repo is in GitHub).
3. Select your repository.

Step 3 — Choose Pipeline Type

- Click **Use the Classic Editor** (bottom of the page).
- Select **Empty Job**.

Step 4 — Configure the Pipeline

- **Pipeline Name** → Example: RetailSales-CI
- Choose your **Default branch** (e.g., main or dev).

Step 5 — Add Tasks

Inside the **Agent Job**:

1. **Add "Use Python Version" Task**
    - Select version **3.10** (or your version).
2. **Add "Command Line" Task → Install Dependencies**
3. pip install -r requirements.txt
4. **Add "Command Line" Task → Run Tests**
5. pytest
6. **Add "Publish Build Artifacts" Task**
    - **Path** → $(System.DefaultWorkingDirectory)/outputs
    - **Artifact Name** → etl_outputs.

Step 6 — Save & Run

- Click **Save & Queue** → Select your branch → Run.
- After completion, check **Artifacts** to confirm category_metrics.csv and other outputs.

Part 2: Create CD Pipeline (Continuous Deployment)

This pipeline **deploys your outputs** (CSV / Docker image / Notebooks) to **staging or production**.

Step 1 — Go to Releases

1. Go to **Pipelines → Releases**.
2. Click **New Pipeline**.
3. Select **Empty Job**.

Step 2 — Add Artifact

1. Click **Add Artifact**.
2. Choose the **CI Pipeline** you created earlier.
3. Select the **Latest version**.
4. Click **Add**.

Step 3 — Configure Stage

- Rename **Stage 1** → Staging (or Production later).
- Click **Tasks** on the stage.

Step 4 — Add Deployment Tasks

Inside your stage, choose based on deployment type:

Option A → Save outputs to Azure Blob Storage

1. Click + → Add **Azure File Copy** task.
2. Choose your **Azure Service Connection**.
3. Select:
    - **Source Folder** → $(System.DefaultWorkingDirectory)/etl_outputs
    - **Container Name** → retail-data
    - **Blob Prefix** → metrics.

Option B → Deploy Docker Image to Azure Web App

1. Add **Azure Web App for Containers** task.
2. Select:
    - Service Connection → Your Azure SP connection.
    - Web App Name → e.g. retail-sales-dashboard.
    - Image Source → Azure Container Registry.
    - Image Tag → $(Build.BuildId).

Step 5 — Enable Continuous Deployment Trigger

1. Go to **Pipeline → Artifacts → Lightning Icon**.

2. Turn on **Continuous Deployment Trigger**.
3. Now, whenever CI publishes a new artifact, CD will deploy automatically.

Step 6 — Add Production Stage (Optional)

- Clone the staging stage → Rename to **Production**.
- Add **Pre-deployment Approvals**:
    - Go to **Environments → Approvals and Checks** → Add an approval step before Production deploy.

Part 3: Verify CI/CD

- Push code → CI pipeline runs → Builds artifacts.
- Once CI succeeds → CD pipeline automatically deploys.
- Check deployment logs:
    - Go to **Releases → Latest Release**.
    - See real-time logs and confirm the deployment.

Final Setup Overview

Stage	Purpose	Trigger	Output
CI	Build, test, publish	On every code push	etl_outputs artifacts
CD	Deploy outputs	Auto after CI	Web app / Blob / Dashboard

.