# CI/CD Pipeline in Azure DevOps for Python Data Processing

## Introduction

This document explains the concept of a sample CI/CD pipeline built using Azure DevOps. The pipeline is designed to automate the process of fetching raw data, processing it, and publishing the results as artifacts. The pipeline ensures consistency, repeatability, and automation in data workflows.

## Project Structure

The project contains Python scripts and a pipeline definition file:  
  
azure-data-pipeline/  
├── data\_pipeline/  
│ ├── fetch\_data.py  
│ ├── process\_data.py  
│ └── requirements.txt  
└── azure-pipelines.yml  
  
- fetch\_data.py → Fetches raw data and saves it as raw\_data.json.  
- process\_data.py → Processes the raw data and generates processed\_data.json.  
- requirements.txt → Contains Python dependencies (e.g., pandas).  
- azure-pipelines.yml → Defines the Azure DevOps CI/CD pipeline steps.

## Pipeline Concept

The pipeline consists of several stages:  
  
1. Trigger → The pipeline is triggered when changes are pushed to the main branch.  
2. Checkout → Fetches the source code from the connected repository.  
3. Set up Python Environment → Installs Python 3.10 on the agent machine.  
4. Install Dependencies → Uses pip to install required libraries from requirements.txt.  
5. Run Data Fetcher → Executes fetch\_data.py, generating raw\_data.json.  
6. Run Data Processor → Executes process\_data.py, generating processed\_data.json.  
7. Publish Artifacts → Publishes processed\_data.json as a pipeline artifact.

## Azure Pipeline YAML Breakdown

The azure-pipelines.yml file defines the automation steps:  
  
- trigger → Runs the pipeline on main branch updates.  
- pool → Specifies the build agent (Ubuntu VM).  
- Checkout@1 → Downloads the repository.  
- UsePythonVersion@0 → Sets Python 3.10.  
- Install dependencies → Runs pip install -r requirements.txt.  
- Fetch raw data → Runs fetch\_data.py.  
- Process data → Runs process\_data.py.  
- PublishBuildArtifacts@1 → Uploads processed\_data.json.

## How It Works

1. Developer pushes code changes to the main branch.  
2. Pipeline triggers automatically in Azure DevOps.  
3. Repository is cloned on the build agent.  
4. Python environment and dependencies are set up.  
5. Scripts are executed step by step:  
 - Raw data is created.  
 - Processed data is generated.  
6. The final processed output is stored as a pipeline artifact.

## Benefits of the Pipeline

- Automation → No manual execution of scripts needed.  
- Consistency → Same steps are repeated every time.  
- Traceability → Artifacts and logs are stored for future reference.  
- Scalability → Can be extended for large datasets or additional steps.