Building a CI/CD Pipeline in Azure DevOps for Python Data Processing

1. Introduction

In this project, we will set up a **CI/CD pipeline** in **Azure DevOps** that automates the following tasks:

1. Clone a repository from **GitHub** or **Azure Repos**.
2. Install the required dependencies.
3. Run Python scripts to **fetch and process data**.
4. Publish the processed data file as an **artifact** for later use.

2. Project Structure

Our project directory is organized as follows:

```
azure-data-pipeline/
├── data_pipeline/
│   ├── fetch_data.py        # Script to fetch and store raw data
│   ├── process_data.py       # Script to process and filter data
│   └── requirements.txt      # Dependencies required for the project
└── azure-pipelines.yml       # Azure DevOps pipeline configuration
```

3. Step 1: Python Scripts

We will use **two Python scripts**:

- One to fetch raw data and store it in a JSON file.
- Another to process the data and create a cleaned output file.

3.1 fetch_data.py

This script generates a dataset of students and writes it into a JSON file called **raw_data.json**.

```
import json
```

```
def fetch():
    data = {"students": [
        {"id": 1, "name": "Abhinav", "marks": 78},
        {"id": 2, "name": "Priya", "marks": 85},
        {"id": 3, "name": "Rahul", "marks": 92},
    ]}
    with open("raw_data.json", "w") as f:
        json.dump(data, f)
    print("Raw data fetched and saved to raw_data.json")


if __name__ == "__main__":
    fetch()
```

Explanation:

- We create a sample dataset with student names and marks.
- The dataset is saved into a file called **raw_data.json**.
- This file will be used later by the processing script.

3.2 process_data.py

This script reads the raw data, filters out students with marks **greater than 80**, and writes the results into a new file called **processed_data.json**.

```
import json


def process():
    with open("raw_data.json", "r") as f:
        data = json.load(f)
    high_scorers = [s for s in data["students"] if s["marks"] > 80]
    with open("processed_data.json", "w") as f:
        json.dump(high_scorers, f)
    print("Processed data saved to processed_data.json")


if __name__ == "__main__":
    process()
```

Explanation:

- Loads **raw_data.json**.
- Filters students who scored **above 80**.
- Saves the result into **processed_data.json**.
- This processed file will be published as an artifact.

3.3 requirements.txt

This file lists all Python dependencies required for the project.

- pandas

Explanation:
Even though our current code does not use **pandas**, we include it here to demonstrate how to manage dependencies. If your script uses any other libraries, add them here.

4. Step 2: Creating the Azure Pipeline

We now configure **Azure Pipelines** using the azure-pipelines.yml file.
This file defines **all the tasks** our CI/CD pipeline should perform.

4.1 azure-pipelines.yml

```
trigger:
- main    # Run the pipeline whenever code is pushed to the main branch


pool:
  vmImage: ubuntu-latest  # Use an Ubuntu VM to run the pipeline


steps:
# Step 1: Checkout the repository
- task: Checkout@1
```

```
# Step 2: Set up Python
- task: UsePythonVersion@0
  inputs:
    versionSpec: '3.10'
    addToPath: true

# Step 3: Install dependencies
- script: |
    python -m pip install --upgrade pip
    pip install -r data_pipeline/requirements.txt
  displayName: 'Install dependencies'

# Step 4: Run the data fetcher
- script: |
    cd data_pipeline
    python fetch_data.py
  displayName: 'Fetch raw data'

# Step 5: Process the data
- script: |
    cd data_pipeline
    python process_data.py
  displayName: 'Process data'

# Step 6: Publish the processed file as an artifact
- task: PublishBuildArtifacts@1
  inputs:
    PathtoPublish: 'data_pipeline/processed_data.json'
    ArtifactName: 'ProcessedData'
    publishLocation: 'Container'
```

4.2 Explanation of Each Section

Section	Purpose
trigger	Specifies when the pipeline runs. Here, it runs on **main** branch updates.
pool	Uses a Microsoft-hosted Ubuntu agent to run tasks.
Checkout@1	Clones your repository into the pipeline environment.
UsePythonVersion	Sets up Python **3.10** for the build agent.
Install dependencies	Installs the required Python libraries from requirements.txt.
Fetch raw data	Executes fetch_data.py to generate raw_data.json.
Process data	Runs process_data.py to generate processed_data.json.
PublishBuildArtifacts	Saves the processed file as an artifact so it can be downloaded later.

5. Step 3: Running the Pipeline

Follow these steps to execute the pipeline in **Azure DevOps**:

1. **Go to Azure DevOps Portal** → https://dev.azure.com
2. Navigate to **Pipelines** → Click **New Pipeline**.
3. Select your repository (**GitHub** or **Azure Repos**).
4. Choose **YAML pipeline**.
5. Point it to your **azure-pipelines.yml** file.
6. Click **Save & Run**.
7. The pipeline will:
    - Clone your repo.
    - Install dependencies.
    - Run Python scripts.
    - Publish processed_data.json as an artifact.

6. Expected Outputs

After the pipeline runs successfully:

File	Description	Location
raw_data.json	Generated by fetch_data.py	Inside data_pipeline/
processed_data.json	Filtered data from process_data.py	Published as an **artifact**

7. Advantages of Using Azure DevOps for CI/CD

- **Automation** → Eliminates manual execution of scripts.
- **Consistency** → Ensures the same steps run every time.
- **Artifact Management** → Easily stores and retrieves processed files.
- **Scalability** → Can extend pipelines for testing, deployments, or ML workflows.

8. Conclusion

In this project, we built a complete **CI/CD pipeline** in **Azure DevOps** for a Python-based data processing task.
We automated:

- Code checkout
- Dependency installation
- Data fetching & processing
- Artifact publishing