Prerequisites

1. **Azure DevOps Account**: Set up a project in Azure DevOps.
2. **Azure Databricks Workspace**: Access to an Azure Databricks workspace.
3. **Service Principal or Personal Access Token (PAT)** for Azure Databricks.
4. **Databricks CLI Installed and Configured**: The Databricks CLI will be used to interact with the Databricks workspace.

Step 1: Set Up the Databricks CLI

1. Install the Databricks CLI locally or in the agent you are using.

   ```
   pip install databricks-cli
   ```

2. Configure the Databricks CLI with your workspace information and token:

   ```
   databricks configure --token
   ```

   You will be prompted to provide:
   - Databricks Host URL
   - Token (You can generate a PAT in Databricks)

Step 2: Create an Azure DevOps Pipeline

1. **Create a YAML Pipeline** in Azure DevOps.
2. **Add Variables**:
   - Add the following variables to the Azure DevOps pipeline for the Databricks configuration:
     - `DATABRICKS_HOST`: The URL of your Azure Databricks workspace.
     - `DATABRICKS_TOKEN`: The Personal Access Token.

Step 3: Azure DevOps YAML Pipeline Example

Here is an `azure-pipelines.yml` file to execute a Databricks notebook.

```
# Azure DevOps pipeline YAML for deploying the fraud detection system
trigger:
    - main
pool:
    vmImage: 'ubuntu-latest'
steps:
    - task: UsePythonVersion@0
      inputs:
          versionSpec: '3.x'

    - script: |
          pip install -r requirements.txt
          python deploy_fraud_detection.py
      displayName: 'Deploy Fraud Detection System'

    - task: AzureMonitorMetrics@0
      inputs:
          monitorName: 'FraudAlerts'
          alertCriteria: 'Suspicious Transaction Detected'
```

Explanation

1. **Trigger**: The pipeline triggers when changes are pushed to the `main` branch.
2. **Pool**: It uses the latest Ubuntu image.
3. **Install Python and Databricks CLI**: The pipeline installs Python and the Databricks CLI.
4. **Configure Databricks CLI**: It configures the CLI using the environment variables (`DATABRICKS_HOST` and `DATABRICKS_TOKEN`).
5. **Upload Notebook**: The notebook (`sample_notebook.py`) is uploaded to the Databricks workspace in the `/Shared/` directory.
6. **Run Notebook**:
   - A JSON file (`run_config.json`) is used to specify the job configuration for the notebook run.
   - The `run_id` is fetched, and the pipeline waits for the job to complete.

Sample JSON Config File (`run_config.json`)

This file defines the notebook parameters and cluster settings:

```
{
  "run_name": "Sample Notebook Run",
  "new_cluster": {
    "spark_version": "10.4.x-scala2.12",
    "node_type_id": "Standard_DS3_v2",
    "num_workers": 2
  },
  "notebook_task": {
    "notebook_path": "/Shared/sample_notebook",
    "base_parameters": {
      "param1": "value1",
      "param2": "value2"
      }
    }
  }
```

Summary

- **Step 1**: The pipeline installs Python and Databricks CLI.
- **Step 2**: Configures the Databricks CLI using the host and token.
- **Step 3**: Uploads the notebook to the Databricks workspace.
- **Step 4**: Runs the notebook in Azure Databricks using the configuration from `run_config.json`.

Key Points

- **Databricks CLI**: This is used to interact with Databricks for uploading notebooks and running jobs.
- **Azure DevOps Variables**: Keep sensitive information like tokens in the Azure DevOps variable groups or secrets.
- **Run Configuration**: The JSON file (`run_config.json`) contains the configuration details for running the notebook, including cluster details.