Week 5: Automating the Environmental Monitoring System with Azure DevOps

Prerequisites:

1. **Azure DevOps Account**: Ensure you have a project set up in Azure DevOps.
2. **Azure Databricks Workspace**: Access an Azure Databricks workspace where your notebooks and clusters are hosted.
3. **Service Principal or Personal Access Token (PAT) for Azure Databricks**: Generate a PAT in Databricks for authentication.
4. **Databricks CLI Installed and Configured**: Install the Databricks CLI on your local machine or CI agent for pipeline integration

Step 1: Set up the databricks CLI:

1. **Install Databricks CLI:**

   Pip install databricks-cli

2. **Configure the Databricks CLI**:

   databricks configure --token

It provides the following:

Databricks Host URL

Token: Generate a PAT in Databricks for authentication.

Step 2: Create an Azure DevOps pipeline:

1. **Create a YAML Pipeline**:
   - Go to **Pipelines → Create Pipeline** in Azure DevOps.
   - Select your repository where the project code is stored.
   - Choose to configure your pipeline using a YAML file.

2. **Add Variables**: In Azure DevOps, navigate to **Pipelines > Library** and add the following variables for Databricks configuration:

   DATABRICKS_HOST: The URL of your Azure Databricks workspace.
   DATABRICKS_TOKEN: The Personal Access Token.

Step 3: Azure DevOps YAML Pipeline Example

Example for azure-pipelines.yml file:

```
trigger:
  branches:
    include:
      - main

pool:
  vmImage: 'ubuntu-latest'

variables:
  DATABRICKS_HOST: 'https://<databricks-instance>.azuredatabricks.net'
  DATABRICKS_TOKEN: $(databricksToken)

steps:
  # Step 1: Install Python and Databricks CLI
  - task: UsePythonVersion@0
    inputs:
      versionSpec: '3.x'
      addToPath: true
  - script: |
      pip install databricks-cli
    displayName: 'Install Databricks CLI'

  # Step 2: Configure Databricks CLI
  - script: |
      databricks configure --host $(DATABRICKS_HOST) --token $(DATABRICKS_TOKEN)
    displayName: 'Configure Databricks CLI'
    env:
      DATABRICKS_HOST: $(DATABRICKS_HOST)
      DATABRICKS_TOKEN: $(DATABRICKS_TOKEN)
```

```
# Step 3: Upload Notebook to Databricks Workspace
 - script: |
     databricks workspace import ./notebooks/Environment_notebook.py
      /Shared/Environment_notebook -l PYTHON
   displayName: 'Upload Notebook to Databricks Workspace'

 # Step 4: Run Databricks Notebook
 - script: |
     JOB_ID=$(databricks runs submit --json-file run_config.json | jq -r '.run_id')
     echo "Job ID: $JOB_ID"
     databricks runs wait --run-id $JOB_ID
   displayName: 'Run Databricks Notebook'
```

Explanation of the Pipeline

1. **Trigger**: The pipeline will trigger automatically when changes are pushed to the main branch.
2. **Pool**: It uses the latest Ubuntu-latest image for the build environment.
3. **Install Python and Databricks CLI**: The pipeline installs Python and the Databricks CLI.
4. **Configure Databricks CLI**: It configures the CLI using the DATABRICKS_HOST and DATABRICKS_TOKEN environment variables.
5. **Upload Notebook**: The notebook (Environment_notebook.py) is uploaded to the Databricks workspace in the /Shared/ directory.
6. **Run Notebook**: The pipeline submits the notebook to be executed using the configuration from the JSON file (run_config.json).

Step 4: Run Databricks Notebook with JSON Config File

A **JSON configuration file** (e.g., run_config.json) defines the notebook parameters and cluster settings for running the notebook.

Sample JSON config file(run_config.json):

```
{
 "run_name": "Environment 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/Environment_notebook",
  "base_parameters": {
   "param1": "value1",
   "param2": "value2"
  }
 }
}
```

- **run_name**: The name of the notebook run.
- **new_cluster**: Cluster configuration.
- **notebook_task**: The path to the notebook in the Databricks workspace and any parameters it requires

Summary of the Pipeline:

Step 1: The pipeline installs Python and Databricks CLI.
Step 2: Configures the Databricks CLI using the host URL and token for authentication.
Step 3: Uploads the Environment_notebook.py to the Databricks workspace.
Step 4: Runs the uploaded notebook using the configuration in run_config.json, which includes cluster specifications and parameters for the notebook.

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 Azure DevOps variable groups or as secrets.

Run Configuration: The JSON file (`run_config.json`) specifies the cluster details and parameters for the notebook execution.