Week 5: Real-Time Environmental Monitoring and Prediction System for Air Quality and Weather

Prerequisites:

1. Azure DevOps Account: Verify that an Azure DevOps project is established.

2. Azure Databricks Workspace: Go to the workspace on Azure Databricks where your clusters and notebooks are located.

3. For Azure Databricks, the Service Principal or Personal Access Token (PAT): In Databricks, create a PAT for authentication.

4. Installed and Configured Databricks CLI: To facilitate pipeline integration, install the Databricks CLI on your local computer or CI agent.

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:

URL of Databricks' host
Token: To authenticate, create a PAT in Databricks.

Step 2: Create an Azure DevOps pipeline:

1. Create a YAML Pipeline:

- Navigate to your Azure DevOps project.
- Go to Pipelines > New Pipeline.
- Select your repository and choose “YAML” to create a new pipeline.

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

Eg 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'

An Overview of the Pipeline

1. Trigger: When modifications are pushed to the main branch, the pipeline will start up immediately.

2. Pool: The build environment is based on the most recent Ubuntu image.

3. Install Python and the Databricks CLI: Python and the Databricks CLI are installed via the pipeline.

4. Configure the Databricks CLI: The environment variables DATABRICKS_HOST and DATABRICKS_TOKEN are used to configure the CLI.

5. Upload Notebook: In the /Shared/ directory of the Databricks workspace, the notebook (Environment_notebook.py) is posted.

6. Run Notebook: Using the configuration from the JSON file (run_config.json), the pipeline submits the notebook to be executed.

Step 4: Run Databricks Notebook with JSON Config File

The notebook parameters and cluster settings are defined in a JSON configuration file (such as run_config.json) before the notebook is executed.

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 notebook run's name.
• **new_cluster:** Configuration of the cluster.
• **notebook_task:** The path and any necessary parameters for the notebook in the Databricks workspace

Summary:

Step 1: Databricks CLI and Python are installed via the pipeline.

Step 2: Sets up the Databricks CLI by authenticating with the token and host URL.

Step 3: Upload Environment Notebook.py to the Databricks workspace in step three.

Step 4: Utilizing the settings and cluster specifications found in run_config.json, the uploaded notebook is executed utilizing the configuration.

Key Points:

Databricks CLI: This is the interface via which jobs and notebook uploads are done with Databricks.

Azure DevOps Variables: Store private data, such as tokens, as secrets or in Azure DevOps variable groups.

Run Configuration: The cluster information and notebook execution parameters are specified in the JSON file run_config.json.