HOW TO INTEGRATE AMAZON AIRFLOW WITH DATABARICKS

Prerequisites

1. **Amazon MWAA Environment**: A running MWAA environment. You should have its ARN and know how to update its requirements and connections.
2. **Databricks Workspace**: An active Databricks workspace URL (e.g., https://dbc-a1b2345c-d6e7.cloud.databricks.com).
3. **Databricks Access Token**: A personal access token generated from your Databricks user account (User Settings -> Developer -> Access Tokens) or a service account. This token is what Airflow will use to authenticate.
4. **IAM Permissions**: Ensure the MWAA environment's execution role has necessary permissions (e.g., access to any S3 buckets your DAGs might need).

Step-by-Step Integration Guide

Step 1: Install the Databricks Provider Package in MWAA

MWAA requires you to specify Python dependencies in a requirements.txt file uploaded to your environment's S3 bucket.

1. Create a file named requirements.txt.
2. Add the following line to it:

txt

apache-airflow-providers-databricks>=5.0.0

1. Upload this file to the requirements folder in the S3 bucket associated with your MWAA environment.
2. In the AWS MWAA console, navigate to your environment and go to the **DAG code** section in the **Configuration** tab.
3. Point the **Requirements file** option to your uploaded requirements.txt file (e.g., s3://your-mwaa-bucket/requirements.txt).
4. **Trigger an update** of the environment. This will install the provider package, which includes the necessary operators, hooks, and connections.
5. Step 2: Configure the Airflow Connection to Databricks
6. You need to securely store your Databricks credentials in Airflow. The best practice is to use an **Airflow Connection**.

**Using the AWS Console (Web UI)**

1. Open your MWAA environment and click on **Open Airflow UI**.
2. Go to **Admin -> Connections**.
3. Click the **+** button to add a new connection.
4. Fill out the form as follows:
   * **Connection Id**: databricks\_default (This is the default ID the operators look for. You can use a custom ID if you prefer).
   * **Connection Type**: Select **Databricks** from the dropdown.
   * **Host**: Your Databricks workspace URL (e.g., https://dbc-a1b2345c-d6e7.cloud.databricks.com). *Do not include the /api/ part.*
   * **Extra**: Paste a JSON blob with your personal access token:

json

{"token": "dapi1234567890abcdefghijklmnopqrstuv"}

Step 3: Write Your DAG USING DatabricksSubmitRunOperator

create an Airflow DAG that uses the DatabricksSubmitRunOperator to submit a one-time job run, or the DatabricksRunNowOperator to trigger an existing job definition.

Create a Python file (e.g., databricks\_etl\_dag.py) and upload it to the dags folder in your MWAA S3 bucket.

**Using**DatabricksSubmitRunOperator

from datetime import datetime, timedelta

from airflow import DAG

from airflow.providers.databricks.operators.databricks import DatabricksSubmitRunOperator

default\_args = {

'owner': 'airflow',

'depends\_on\_past': False,

'email\_on\_failure': False,

'email\_on\_retry': False,

'retries': 1,

'retry\_delay': timedelta(minutes=5)

}

with DAG('databricks\_submit\_example',

start\_date=datetime(2023, 10, 1),

schedule\_interval='@daily',

default\_args=default\_args,

catchup=False) as dag:

# Define the JSON structure for a new cluster

new\_cluster = {

'spark\_version': '11.3.x-scala2.12',

'node\_type\_id': 'i3.xlarge',

'num\_workers': 2,

'aws\_attributes': {

'availability': 'ON\_DEMAND'

}

}

# Define the Notebook task

notebook\_task = {

'notebook\_path': '/Users/your.email@company.com/My\_Notebook',

}

# Define the Spark JAR task (alternative to notebook)

# spark\_jar\_task = {

# 'main\_class\_name': 'com.company.YourMainClass',

# 'parameters': ['--input-path', 's3://input-bucket/path', '--output-path', 's3://output-bucket/path']

# }

# Submit the job run

submit\_run = DatabricksSubmitRunOperator(

task\_id='submit\_databricks\_job',

databricks\_conn\_id='databricks\_default', # This matches the connection ID we set up

new\_cluster=new\_cluster,

notebook\_task=notebook\_task,

# spark\_jar\_task=spark\_jar\_task, # Use either notebook\_task OR spark\_jar\_task

do\_xcom\_push=True # Pushes the run\_id and run\_page\_url to XCom, useful for logging

)

submit\_run

Step 4: Monitor Your Jobs

1. **In Airflow UI**: The DAG runs will show the status (success, failure, running). You can see logs from the Airflow task itself, which will include the run\_id and a link to the Databricks run.
2. **In Databricks UI**: Go to your Databricks workspace -> **Workflows** -> **Jobs**. You will see all the runs triggered by Airflow. This is where you find detailed Spark logs, cluster metrics, and task output.