Exercise 8: Exploring usage of AWS S3 with astronomer cloud

Introduction

In this lab, we will learn how to integrate Apache Airflow with AWS S3 using Astronomer Cloud, which is a managed service for Apache Airflow. You will create a DAG (Directed Acyclic Graph) that connects to AWS S3, waits for a file to arrive in a specific S3 bucket, reads that file (in CSV format), and prints the content. This exercise will help you understand how to utilize Airflow operators and sensors to interact with AWS S3 and automate data processing workflows.

Objectives

- Understand how to connect Apache Airflow to AWS S3.
- Use the S3KeySensor to wait for a file to arrive in an S3 bucket.
- Read the content of a CSV file stored in an S3 bucket using S3Hook.
- Automate the process of checking, retrieving, and processing files from AWS S3 using Airflow DAGs.
- Learn how to structure and define a DAG in Airflow to automate ETL tasks.

Preparation

Prerequisites

- Familiarity with Airflow DAGs and task dependencies.
- Basic understanding of AWS cloud and S3 storage..
- Docker installed and running.
- Python (3.7 or above) and pip installed.
- Active Astronomer Cloud account and workspace.
- Astro CLI installed on your local machine
    - Verify using bash command `astro version`

Create a aws account to access s3

- Go to AWS Sign-Up and click Create an AWS Account.
- Provide your personal or company information, including a valid email address.
- Choose an AWS Support plan (you can select the Basic plan for free).
- Enter your payment information (AWS provides a free tier, but some services may charge).
- Complete the identity verification (via phone).
- Once the account is created, sign in to the AWS Management Console at console.aws.amazon.com.

Let's Get Started

Part 1: Environment Setup

Step 1. Authenticate Astro CLI:

- Authenticate your CLI with Astronomer Cloud:
    - i. Use command `astro login`
    - ii. Press Enter

Step 2. Project setup

- Ensure you have an initialized Airflow project. If not, initialize a new project using:
    - i. Create a new directory and navigate into it:
        1. `mkdir astro_cloud`
        2. `cd astro_cloud`
        3. `astro dev init`

Part 2: Creating AWS Connection in Astronomer Cloud

Step 1. Set Up AWS Connection in Astronomer Cloud

- Log in to your Astronomer Cloud workspace.
- Navigate to your Deployment and go to the Environment Variables or Connections tab.
- Get aws connection details from aws console
    - i. Go to dashboard
        1. Click on your profile on top right most corner
        2. Select Security credentials
        3. Go to Access keys
            - a. Make sure to save that Access Key ID and Secret Access Key at some place for further use, otherwise it will not be available again.
        4. Use these parameters below while creating connection
- Add a new connection with the following details:
    - i. Conn Id: aws_s3
    - ii. Conn Type: Amazon Web Services.
    - iii. Access Key: Your AWS Access Key.
    - iv. Secret Key: Your AWS Secret Key.

Save the connection.

Part 3: Deploying to Astronomer Cloud

Step 1. Add a new Dag

- Navigate to the `dags/` directory and create a file named `s3_read_and_print_csv_with_wait`.py

```
import pandas as pd

from airflow import DAG

from airflow.providers.amazon.aws.hooks.s3 import S3Hook
```

```python
from airflow.providers.amazon.aws.sensors.s3 import S3KeySensor

from airflow.operators.python import PythonOperator

from airflow.utils.dates import days_ago

import io



# Default arguments for the DAG

default_args = {

   'owner': 'airflow',

   'start_date': days_ago(1),

   'retries': 1,

}



# Define the DAG

dag = DAG(

   's3_read_and_print_csv',

   default_args=default_args,

   description='DAG to read CSV from S3 and print the values',

   schedule_interval=None,  # Set to None for manual execution

)



# AWS S3 connection details

BUCKET_NAME = 'airflowbuckets3'

OBJECT_KEY = 'awsdata/user.csv'

#LOCAL_FILE_PATH = '/usr/local/airflow/include/'
```

```
# Function to read the CSV from S3 and print its values
def read_and_print_csv_from_s3():
   # Use S3Hook to interact with S3
   s3_hook = S3Hook(aws_conn_id='aws_s3')  # Use your AWS connection ID

   # Fetch the file from S3 as a byte stream
   file_obj = s3_hook.get_key(key=OBJECT_KEY, bucket_name=BUCKET_NAME)

   if file_obj:
       # Read the content into a pandas DataFrame
       csv_content = file_obj.get()["Body"].read()
       df = pd.read_csv(io.BytesIO(csv_content))  # Read the CSV from byte stream

       # Print the DataFrame content
       print(df)
   else:
       print(f"File {OBJECT_KEY} not found in bucket {BUCKET_NAME}")

wait_for_file = S3KeySensor(
   task_id='wait_for_s3_file',
   bucket_name=BUCKET_NAME,
   bucket_key=OBJECT_KEY,
   aws_conn_id='aws_s3',  # Use your AWS connection ID
   poke_interval=60,  # How often to check the S3 bucket (in seconds)

   timeout=600,  # How long to wait before giving up (in seconds)

   mode='poke',  # This mode will keep checking until the file is found

   dag=dag,

)



# Task to read and print CSV content

read_csv_task = PythonOperator(

   task_id='read_and_print_csv',

   python_callable=read_and_print_csv_from_s3,

   dag=dag,

)



wait_for_file >> read_csv_task
```

- Save the file and navigate to the project root.
- **Replace AWS S3 details**: In the code above, replace:
    i. `'airflowbuckets3'` with your actual S3 bucket name.
    ii. `'awsdata/user.csv'` with the path to the CSV file in your S3 bucket.
        1. You can use this csv file for verification

Step 2.Update requirements file

- Navigate to the `project` directory and update the file requirements.txt with below text

```
apache-airflow-providers-amazon

pandas
```

Step 3. Create deployment

- Use below command to create astor deployment
  1. astro deploy
  2. It will ask to create a deployment if no deployment is available as of now Provide details like
     1. name of deployment
     2. Cloud region where it need to be deployed
  3. if deployment is available then select that deployment and proceed with deploy

Step 4. Verify deployment

- Log in to the Astronomer Cloud UI and navigate to your deployment.
- Ensure your DAG appears under the DAGs tab.

Step 5. Monitor and Manage Workflows

- Trigger the DAG from the Astronomer Cloud UI.
- Monitor task execution: The `S3KeySensor` task will wait for the file to appear in S3. Once the file is found, it will trigger the next task (`read_and_print_csv`) to read and print the CSV content.
- Check the logs of the `read_and_print_csv` task to see the output of the CSV file printed to the Airflow logs.

Step 6. Update deployment

- Remove the csv file from the cloud location
- Trigger the dag again
- Wait for few mins till the time airflow output show log as similar to
    i. “Poking for key : s3://airflowbuckets3/awsdata/user.csv”
- Upload the csv file to the cloud location
- Again Check logs and task status on Airflow UI

Conclusion

In this lab, you:

- Use Airflow with Astronomer Cloud to automate workflows that interact with AWS S3.
- Utilize S3KeySensor to wait for the arrival of a file in an S3 bucket.
- Leverage S3Hook to read a CSV file from S3 and process it using Pandas.

By following this guide, you now have the skills to automate the process of interacting with AWS S3 within Airflow, making it easier to build data pipelines for tasks such as ETL (Extract, Transform, Load).