***Building Automated Data Processing Pipelines with Airflow on AWS for Noirframes***

Priyanka Tandon | Komal Datta Sidgiddi

Incedo Inc.

[piyanka.tandon@incedoinc.com](mailto:piyanka.tandon@incedoinc.com) | [komaldatta.s@incedoinc.com](mailto:komaldatta.s@incedoinc.com)

### Abstract

The project focuses on developing a robust and scalable data processing ecosystem utilizing AWS services. This initiative aims to design and implement an automated data pipeline using Apache Airflow hosted on EC2, with data storage on EC2, VPC for secure network connectivity, and IAM for access control. The primary goal is to streamline data ingestion, transformation, and mutation workflows while ensuring cost-effectiveness and robust security. By leveraging Airflow’s Directed Acyclic Graphs (DAGs), the project will create reusable workflows that manage recurring data tasks efficiently. The outcome will be a secure, scalable, and well-documented system that adheres to best practices in data processing and cloud architecture.

### Introduction

**Context:**

* Noirframes requires a scalable, secure data processing system.
* The project will leverage AWS services to meet these needs.

**Project Components:**

* **Apache Airflow:** Deployed on EC2 to orchestrate data workflows.
* **Amazon EC2:** For storage of data.
* **Amazon VPC:** For private network connectivity.
* **AWS IAM:** For managing access and permissions.

**Objectives:**

* Automate and manage data ingestion, transformation, and mutation tasks.
* Design reusable workflows for recurring data formats.
* Optimize resource use to manage costs effectively.
* Implement best practices for security and reliability.

**Deliverables:**

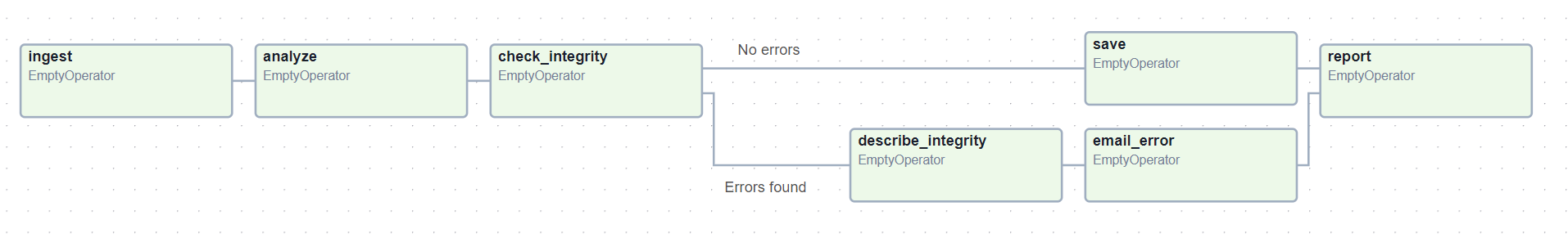
* A comprehensive, documented data processing ecosystem.
* Functioning Airflow DAGs and workflow automation.
* Cost estimation and performance validation.

**Scope:**

* The project will cover design, implementation, and testing of the data processing system tailored to Noirframes' needs.

### Architecture Overview:

Airflow is a platform that lets us build and run *workflows*. A workflow is represented as a [DAG](https://airflow.apache.org/docs/apache-airflow/stable/core-concepts/dags.html) (a Directed Acyclic Graph), and contains individual pieces of work called [Tasks](https://airflow.apache.org/docs/apache-airflow/stable/core-concepts/tasks.html), arranged with dependencies and data flows taken into account.



A DAG specifies the dependencies between tasks, which defines the order in which to execute the tasks. Tasks describe what to do, be it fetching data, running analysis, triggering other systems, or more.

**Required components**

A minimal Airflow installation consists of the following components:

* A [scheduler](https://airflow.apache.org/docs/apache-airflow/stable/administration-and-deployment/scheduler.html), which handles both triggering scheduled workflows, and submitting [Tasks](https://airflow.apache.org/docs/apache-airflow/stable/core-concepts/tasks.html) to the executor to run. The [executor](https://airflow.apache.org/docs/apache-airflow/stable/core-concepts/executor/index.html), is a configuration property of the *scheduler*, not a separate component and runs within the scheduler process. There are several executors available out of the box, and you can also write your own.
* A *webserver*, which presents a handy user interface to inspect, trigger and debug the behavior of DAGs and tasks.
* A folder of *DAG files*, which is read by the *scheduler* to figure out what tasks to run and when to run them.
* A *metadata database*, which airflow components use to store the state of workflows and tasks.

A diagram of a software system

Description automatically generated

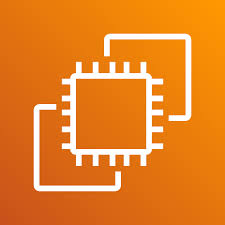
Architecture of our Project:

A diagram of a person's face

Description automatically generated

### Services used

#### Amazon EC2 (Elastic Compute Cloud)



Amazon Elastic Compute Cloud (Amazon EC2) is a web service provided by Amazon Web Services (AWS) that offers resizable compute capacity in the cloud. It is designed to make web-scale cloud computing easier for developers by providing a virtual computing environment that allows them to run applications without needing to invest in physical hardware. Here’s an overview of Amazon EC2, its features, benefits, and use cases:

Key Features of Amazon EC2:

1. Scalability:

- Auto Scaling: Automatically adjust the number of EC2 instances in response to demand to maintain performance and minimize costs.

- Elastic Load Balancing: Distribute incoming traffic across multiple EC2 instances to ensure reliability and availability.

2. Variety of Instance Types:

- Instance Families: Choose from a variety of instance types optimized for different use cases, such as compute-intensive, memory-intensive, storage-optimized, and general-purpose instances.

- On-Demand Instances: Pay for compute capacity by the hour or second with no long-term commitments.

3. Flexible Configuration:

- Customizable: Configure CPU, memory, storage, and networking capacity to meet the specific needs of your applications.

- Pre-configured AMIs: Use Amazon Machine Images (AMIs) to quickly deploy instances with pre-configured software, including operating systems, applications, and custom configurations.

4. Security and Compliance:

- Security Groups: Define rules that control the inbound and outbound traffic to instances, acting as virtual firewalls.

- IAM Roles: Use AWS Identity and Access Management (IAM) roles to manage permissions and control access to EC2 instances.

- Encryption: Secure data in transit and at rest with encryption features.

5. Networking:

- Virtual Private Cloud (VPC): Launch instances in a logically isolated virtual network that you define, providing complete control over network settings.

- Elastic IP Addresses: Assign static IP addresses to instances for consistent, reliable addressing.

6. Storage Options:

- Elastic Block Store (EBS): Persistent block storage volumes for use with EC2 instances, offering high availability and durability.

- Instance Store: Temporary storage that is physically attached to the host computer for high-performance storage needs.

- Amazon S3 Integration: Store and retrieve any amount of data using Amazon S3 for scalable, durable storage.

Benefits of Amazon EC2

1. Cost-Effectiveness:

- Pay-as-You-Go: Only pay for the compute capacity you use, with no upfront costs or long-term contracts.

- Cost Optimization: Save money with Reserved Instances and Spot Instances for predictable or flexible workloads.

2. Flexibility and Agility:

- Quick Deployment: Rapidly scale up or down based on demand, reducing the time to deploy new applications and services.

- Wide Range of Use Cases: Suitable for a variety of applications, from simple web hosting to high-performance computing and big data processing.

3. Global Reach:

- Multiple Regions and Availability Zones: Deploy applications globally with low latency by using AWS regions and availability zones.

4. Integration with AWS Services:

- Seamless Integration: Easily integrate EC2 instances with other AWS services like RDS, Lambda, S3, CloudFront, and more to build comprehensive cloud solutions.

#### AWS Virtual Private Cloud



Amazon Virtual Private Cloud (Amazon VPC) is a service that allows you to create a logically isolated virtual network in the AWS cloud. It provides full control over your virtual networking environment, including selection of your own IP address range, creation of subnets, and configuration of route tables and network gateways. Here’s an overview of Amazon VPC, its features, benefits, and use cases:

Key Features of Amazon VPC

1. Network Isolation and Security

* Private IP Space: Define a custom IP address range using IPv4 or IPv6.
* Subnet Segmentation: Create subnets within your VPC to segment different parts of your application, such as public-facing and internal services.
* Security Groups: Act as virtual firewalls for your instances to control inbound and outbound traffic at the instance level.
* Network ACLs: Provide stateless filtering to control traffic at the subnet level.

1. Internet and VPN Connectivity

* Internet Gateway: Enable communication between instances in your VPC and the internet.
* NAT Gateway: Allow instances in a private subnet to connect to the internet or other AWS services, but prevent the internet from initiating connections with those instances.
* VPN Gateway: Establish a secure connection between your VPC and your on-premises network or another VPC.
* Routing and Traffic Control
* Route Tables: Control the traffic routing between subnets and to the internet or other networks.
* VPC Peering: Connect one VPC to another privately using private IP addresses, enabling instances in either VPC to communicate with each other.

1. Advanced Networking

* Elastic IP Addresses: Assign static IP addresses to instances in your VPC.
* Elastic Network Interfaces (ENIs): Attach additional network interfaces to instances in your VPC for high availability and failover.
* VPC Flow Logs: Capture detailed information about the traffic going to and from network interfaces in your VPC.

1. Enhanced Security Features

* AWS PrivateLink: Securely access services hosted on AWS without using public IP addresses.
* Gateway Endpoints: Privately connect your VPC to supported AWS services, such as S3 and DynamoDB, without requiring an internet gateway, NAT device, VPN connection, or AWS Direct Connect.

Benefits of Amazon VPC

1. Enhanced Security

* Isolation: Each VPC is isolated from other VPCs, ensuring your resources are secure and private.
* Fine-Grained Access Control: Use security groups and network ACLs to tightly control access to your resources.

1. Flexibility and Control

* Custom Network Configurations: Define your own network configurations, including IP address range, subnets, route tables, and gateways.
* Scalability: Easily scale your network infrastructure by adding subnets, instances, and services as needed.

1. Reliability

* High Availability: Deploy resources across multiple Availability Zones (AZs) for fault tolerance and high availability.
* Monitoring and Logging: Use VPC Flow Logs and CloudWatch to monitor and log network traffic for troubleshooting and compliance.

1. Cost-Effective

* Pay-As-You-Go: Only pay for the resources you use, such as NAT gateways, VPN connections, and data transfer.
* Optimized Costs: Reduce costs by designing efficient network architectures and using reserved instances where applicable.

#### AWS Identity and Access Management



AWS Identity and Access Management (IAM) is a web service that helps you securely control access to AWS resources. IAM enables you to manage permissions and access for users, groups, and roles, allowing you to define who can access what resources under which conditions. Here’s an overview of AWS IAM, its features, benefits, and use cases:

Key Features of AWS IAM

1. User Management

* Users: Create and manage individual IAM users with specific credentials for accessing AWS services.
* Groups: Organize IAM users into groups to manage permissions collectively. Users in a group inherit permissions assigned to the group.

1. Access Control

* Policies: Define policies that grant or deny permissions to AWS resources. Policies can be attached to users, groups, and roles.
* Policy Types: Support for identity-based policies (attached to IAM entities) and resource-based policies (attached to resources like S3 buckets).

1. Roles and Temporary Security Credentials

* Roles: Create IAM roles to delegate access to users, applications, or services that don't have a permanent AWS identity.
* Temporary Credentials: Issue temporary security credentials for roles, enabling secure short-term access.

1. Fine-Grained Permissions

* Least Privilege: Implement the principle of least privilege by granting only the permissions needed to perform a task.
* Condition-Based Policies: Apply conditions to policies for more granular control, such as allowing access only from specific IP addresses or during certain times.

1. Multi-Factor Authentication (MFA)

* MFA: Enhance security by requiring users to provide a second form of authentication in addition to their password.

1. Identity Federation

* Single Sign-On (SSO): Allow users to access AWS resources using existing corporate credentials with SSO.
* Federated Access: Integrate with identity providers (e.g., Active Directory, SAML) to provide federated access to AWS resources.

1. Audit and Compliance

* IAM Access Analyzer: Analyze access to your resources to identify unintended access and potential security risks.
* Logging and Monitoring: Use AWS CloudTrail to log IAM actions and CloudWatch to monitor IAM activity for auditing and compliance.

Benefits of AWS IAM

1. Enhanced Security

* Granular Access Control: Define precise permissions for users and resources to minimize the risk of unauthorized access.
* Multi-Factor Authentication: Strengthen security by requiring MFA for critical operations.

1. Flexibility and Scalability

* Customizable Policies: Create tailored policies to meet the specific security and access requirements of your organization.
* Scalable Management: Manage access for a large number of users, applications, and services across different AWS accounts and regions.

1. Centralized Management

* Unified Management Console: Manage all IAM users, groups, roles, and policies from a single console.
* Integration with AWS Services: Seamlessly integrate IAM with other AWS services for consistent access management.

1. Cost-Effective

* No Additional Cost: IAM is free to use. You only pay for the AWS resources you consume.

1. Compliance and Governance

* Audit Trails: Maintain comprehensive audit trails with AWS CloudTrail to support compliance and governance requirements.
* Access Analyzer: Continuously monitor and analyze access patterns to ensure compliance with security best practices.

#### Apache airflow



Apache Airflow is an open-source platform used to programmatically author, schedule, and monitor workflows. Developed initially by Airbnb, it is now an Apache Software Foundation project. Airflow allows you to define workflows as directed acyclic graphs (DAGs) using Python, making it highly flexible and scalable for complex workflows.

Key Features of Apache Airflow

1. Dynamic Workflow Definition

* Python-Based: Define workflows using Python code, enabling dynamic pipeline creation based on parameters and conditions.
* Directed Acyclic Graphs (DAGs): Represent workflows as DAGs, ensuring a clear and logical flow of tasks.

1. Scheduling and Execution

* Scheduler: Built-in scheduler to execute tasks based on specified schedules, supporting both cron expressions and custom intervals.
* Parallel Execution: Support for executing tasks in parallel using multiple worker nodes.

1. Task Dependencies and Management

* Task Dependencies: Define explicit dependencies between tasks, ensuring they are executed in the correct order.
* Retries and Alerts: Configure retries, timeouts, and alerting mechanisms for failed tasks.
* Extensibility
* Plugins and Hooks: Extend Airflow’s capabilities using custom plugins and hooks to integrate with various data sources and services.
* Operators: Pre-built operators for common tasks, such as BashOperator, PythonOperator, MySqlOperator, and many more.

1. User Interface

* Web UI: A rich and intuitive web interface to monitor and manage workflows, view logs, and track the status of DAGs and tasks.
* Graphical Views: Visualize DAGs and task dependencies through graphical representations.

1. Scalability

* Distributed Execution: Scale horizontally by adding more worker nodes to handle large volumes of tasks.
* Task Queues: Use task queues to distribute tasks efficiently among workers.

1. Logging and Monitoring

* Centralized Logging: Collect and view logs centrally for easier debugging and monitoring.
* Metrics and Alerts: Integrate with monitoring tools to collect metrics and set up alerts for workflow performance and failures.

Benefits of Using Apache Airflow

1. Flexibility and Customization

* Dynamic Workflows: Create workflows dynamically using Python, adapting to complex and changing requirements.
* Custom Integrations: Easily integrate with various systems and services using custom operators and hooks.

1. Scalability and Reliability

* Distributed Architecture: Scale out by adding worker nodes to handle increased workload and ensure high availability.
* Retry Mechanism: Automatically retry failed tasks and configure alerting to ensure reliable execution.

1. Comprehensive Monitoring and Management

* User-Friendly Interface: Monitor and manage workflows through an intuitive web interface.
* Detailed Logging: Access detailed logs for troubleshooting and auditing.

1. Community and Ecosystem

* Active Community: Benefit from a large and active open-source community contributing to the development and improvement of Airflow.
* Ecosystem Integration: Integrate with a wide range of tools and services in the data ecosystem, including databases, cloud services, and big data platforms.

#### AWS Simple Storage Service (Future scope)



Amazon Simple Storage Service (Amazon S3) is an object storage service provided by Amazon Web Services (AWS). It offers highly scalable, durable, and secure storage for a wide range of data, from web applications and data archives to disaster recovery and big data analytics. Here’s an overview of Amazon S3, its features, benefits, and use cases:

Key Features of Amazon S3

1. Scalability and Performance

* Unlimited Storage: Store an unlimited amount of data in S3, with each object up to 5 terabytes in size.
* High Throughput: Handle high request rates and data transfer speeds for read/write operations.

1. Data Durability and Availability

* Durability: Designed for 99.999999999% (11 9s) durability, ensuring that your data is highly protected against failures.
* Availability: Offers 99.99% availability over a given year, ensuring your data is accessible when you need it.

1. Security and Compliance

* Encryption: Supports server-side encryption (SSE) and client-side encryption to protect data at rest.
* Access Control: Use IAM policies, bucket policies, and Access Control Lists (ACLs) to manage access to your data.
* Compliance: Meets compliance standards such as HIPAA, PCI-DSS, GDPR, and FedRAMP.

1. Storage Classes

* Standard: General-purpose storage for frequently accessed data.
* Intelligent-Tiering: Automatically moves data between two access tiers (frequent and infrequent) based on changing access patterns.
* Standard-IA (Infrequent Access): For data that is accessed less frequently but requires rapid access when needed.
* One Zone-IA: Lower-cost option for infrequently accessed data that does not require multiple availability zone resilience.
* Glacier: Low-cost storage for archival data with retrieval times ranging from minutes to hours.
* Glacier Deep Archive: Lowest-cost storage option for long-term archival with retrieval times of 12 hours.

1. Data Management

* Lifecycle Policies: Automate the transition of objects between different storage classes based on specified rules.
* Versioning: Keep multiple versions of an object to recover from accidental deletions or overwrites.
* Replication: Enable cross-region replication (CRR) and same-region replication (SRR) for disaster recovery and data redundancy.

1. Data Transfer and Integration

* Transfer Acceleration: Speed up uploads and downloads using Amazon CloudFront’s globally distributed edge locations.
* AWS Integration: Seamlessly integrate with other AWS services such as AWS Lambda, Amazon RDS, Amazon EMR, and AWS Glue.
* S3 Select: Retrieve a subset of data from an object using simple SQL expressions.

1. Event Notifications

* Trigger Events: Configure S3 to send notifications to AWS Lambda, Amazon SQS, or Amazon SNS when specific events occur (e.g., object creation, deletion).

Benefits of Amazon S3

1. Cost-Effective

* Pay-As-You-Go: Pay only for the storage you use, with no minimum fees.
* Cost Optimization: Use different storage classes and lifecycle policies to optimize costs based on access patterns.

1. High Durability and Availability

* Reliable Storage: Designed for high durability and availability, ensuring that your data is safe and accessible.

1. Secure

* Robust Security: Implement fine-grained access controls, encryption, and compliance measures to protect sensitive data.

1. Scalable

* Seamless Scaling: Automatically scale to accommodate growing data volumes without the need for manual intervention.

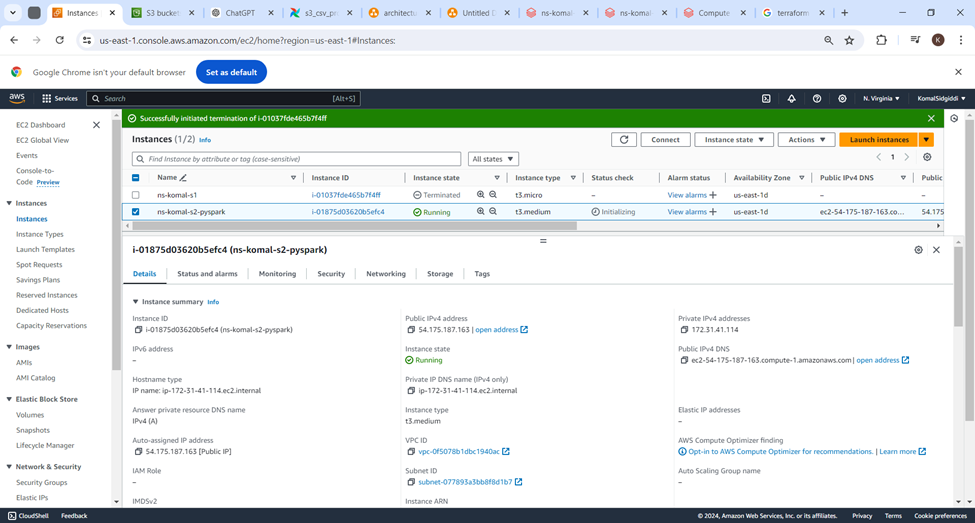
1. Flexible

* Wide Range of Use Cases: Suitable for various use cases, including backup and restore, archival, big data analytics, and content distribution.

### Implementation

#### Spark DAG

1. Create an EC2 instance hosted on the default VPC with Ubuntu 22.04 version and instance type as t3.medium.



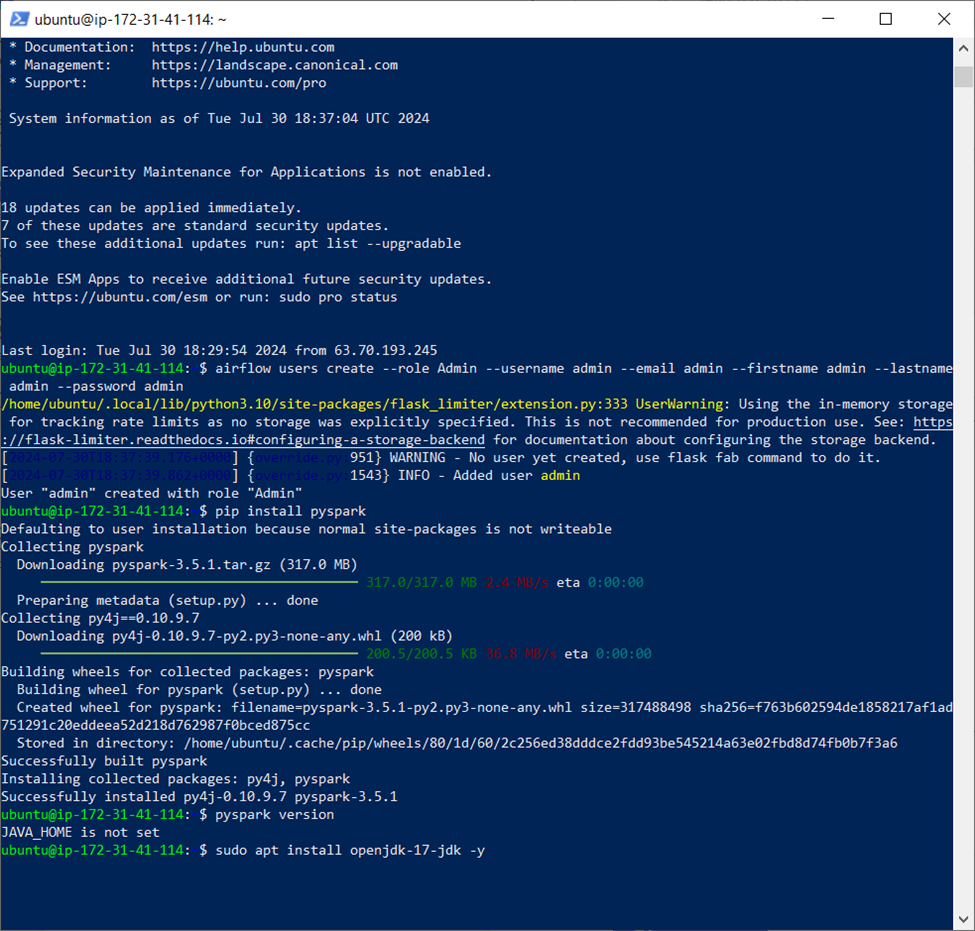
1. Include the default and custom security groups with added port support.
2. Open PowerShell terminal and connect to the EC2 instance.
3. Set up Apache Airflow and PySpark on this instance using the following commands

>> sudo apt update

>> sudo apt install python3-pip

>>pip install pyspark

>> sudo apt install openjdk-17-jdk –y



1. Add Path variable to the Last line in .bashrc

>> nano .bashrc

export JAVA\_HOME=/usr/lib/jvm/java-17-openjdk-amd64  
export PATH=$PATH:$JAVA\_HOME/bin

>> sudo pip install apache-airflow

>> airflow version

1. Ensure inbound rules in the security group have a rule for port 8950. Add if required.
2. In a second PowerShell terminal, connect to instance and run below commands

>> airflow webserver -p 8950

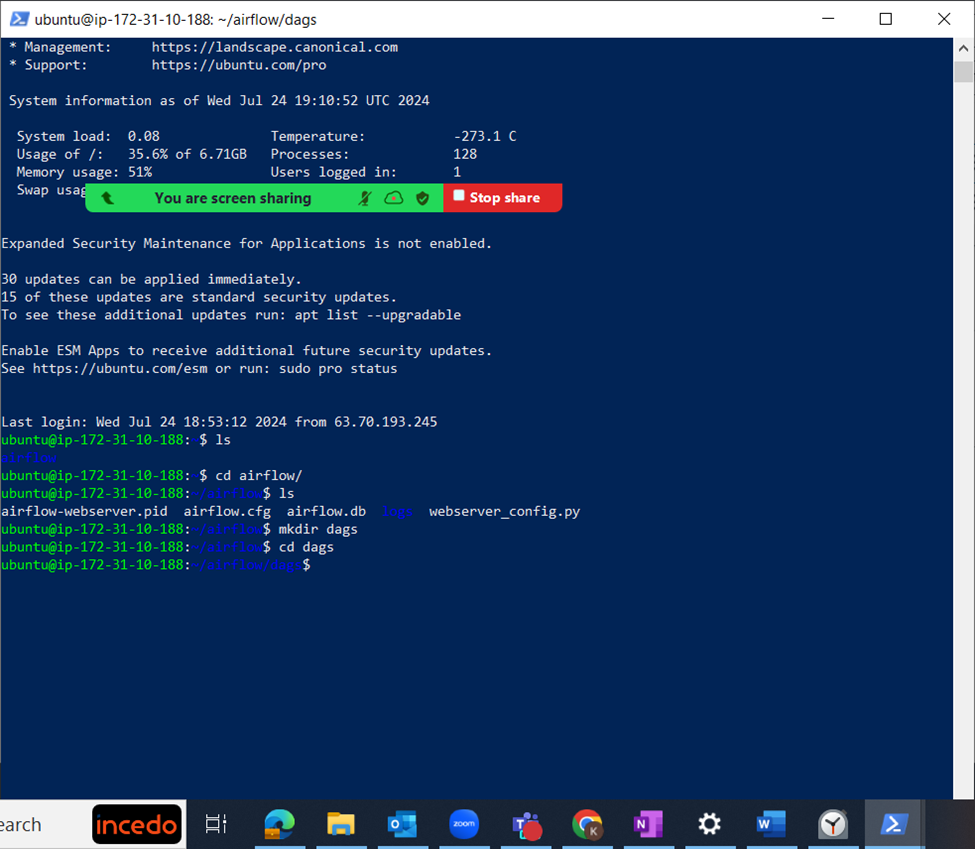
>> airflow users create --role Admin --username admin --email admin --firstname admin --lastname admin --password admin

>> airflow scheduler

1. Once the scheduler is running, open the Airflow webserver page with the publicIPAddress:portNumber, in our case:

54.146.144.3:8950

1. This takes us to the login page. Enter the admin credentials we created using above command.
2. Add new directories for dags, input and output csv files in the airflow folder



1. Get the .csv data files from S3 using the “wget s3\_bucket\_url” command.
2. Move input file to input folder and the blank output file to output folder.
3. Create a .py file in the dag directory you just created and add the following PySpark code

from airflow import DAG

from airflow.operators.python\_operator import PythonOperator

from datetime import datetime, timedelta

from pyspark.sql import SparkSession

from pyspark.sql.functions import col

import pandas as pd

default\_args = {

'owner': 'airflow',

'depends\_on\_past': False,

'start\_date': datetime(2023, 1, 1),

'email\_on\_failure': False,

'email\_on\_retry': False,

'retries': 1,

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

}

dag = DAG(

'pyspark\_csv\_processing\_dag',

default\_args=default\_args,

description='DAG to read, process, and write CSV using PySpark',

schedule\_interval=timedelta(days=1),

)

def create\_spark\_session():

spark = SparkSession.builder \

.appName("PySpark DAG Example") \

.getOrCreate()

return spark

def read\_csv\_with\_pyspark(\*\*kwargs):

spark = create\_spark\_session()

df = spark.read.csv("/home/ubuntu/airflow/airflow\_input/customers\_table.csv", header=True, inferSchema=True)

pd\_df = df.toPandas() # Convert to Pandas DataFrame for XCom serialization

kwargs['ti'].xcom\_push(key='dataframe', value=pd\_df.to\_dict())

spark.stop()

def process\_dataframe\_with\_pyspark(\*\*kwargs):

ti = kwargs['ti']

df\_dict = ti.xcom\_pull(key='dataframe', task\_ids='read\_csv\_with\_pyspark')

pd\_df = pd.DataFrame(df\_dict)

spark = create\_spark\_session()

df = spark.createDataFrame(pd\_df)

filtered\_df = df.filter(col("Country") == 'Germany')

pd\_filtered\_df = filtered\_df.toPandas()

kwargs['ti'].xcom\_push(key='processed\_dataframe', value=pd\_filtered\_df.to\_dict())

spark.stop()

def write\_csv\_with\_pyspark(\*\*kwargs):

ti = kwargs['ti']

df\_dict = ti.xcom\_pull(key='processed\_dataframe', task\_ids='process\_dataframe\_with\_pyspark')

pd\_df = pd.DataFrame(df\_dict)

spark = create\_spark\_session()

df = spark.createDataFrame(pd\_df)

df.write.csv("/home/ubuntu/airflow/airflow\_output/new\_customers\_table.csv", header=True, mode="overwrite")

spark.stop()

# Define the tasks

task1 = PythonOperator(

task\_id='read\_csv\_with\_pyspark',

python\_callable=read\_csv\_with\_pyspark,

provide\_context=True,

dag=dag,

)

task2 = PythonOperator(

task\_id='process\_dataframe\_with\_pyspark',

python\_callable=process\_dataframe\_with\_pyspark,

provide\_context=True,

dag=dag,

)

task3 = PythonOperator(

task\_id='write\_csv\_with\_pyspark',

python\_callable=write\_csv\_with\_pyspark,

provide\_context=True,

dag=dag,

)

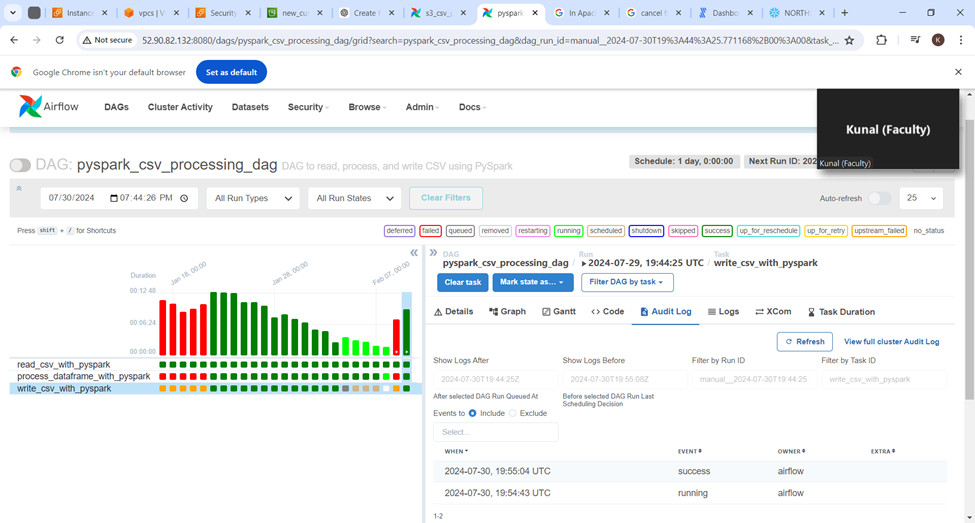
# Set task dependencies

task1 >> task2 >> task3

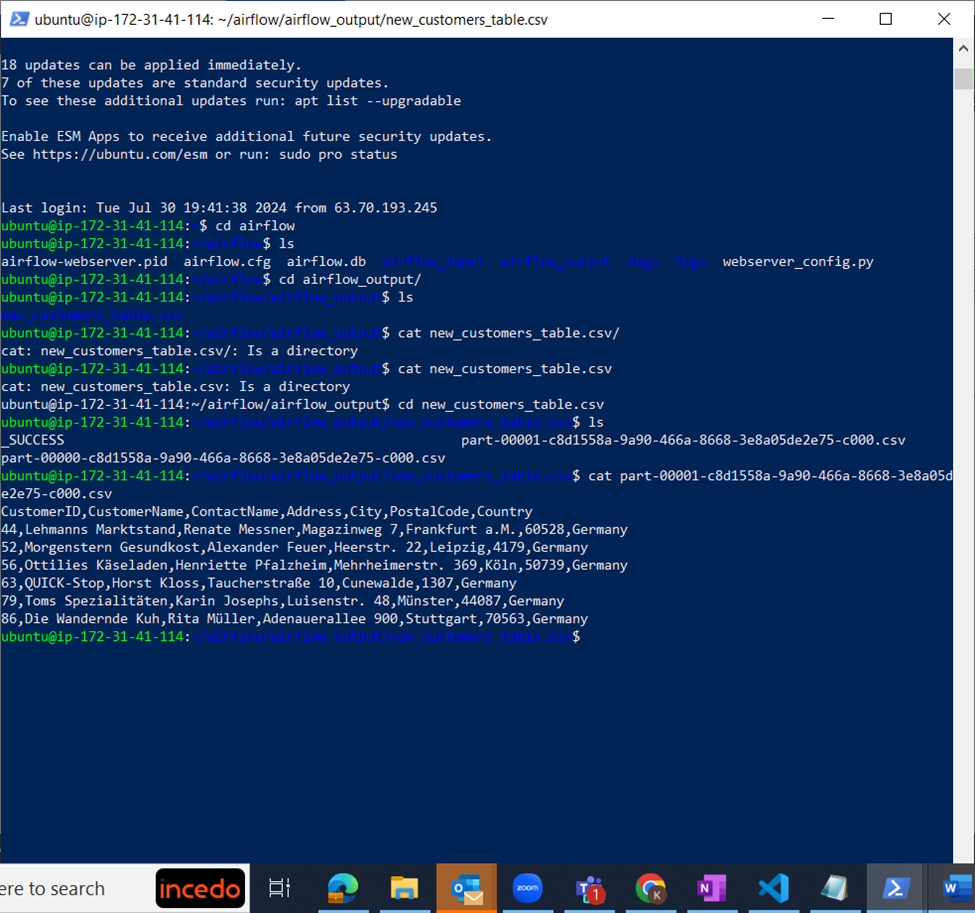
1. Save the file and check Apache Airflow webserver for this dag.
2. In case the dag is not visible, run

>> airflow db init

1. Locate the dag and trigger it



1. On successful DAG completion, the output file looks like this



#### Pandas DAG

1. For the airflow to run successfully on EC2, create instance with below configurations:

OS- UBUNTU

AMI- Ubuntu Server 22.04 LTS

Instance type- t3 small

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. Select the VPC created and default and custom security group.

A screenshot of a computer

Description automatically generated

Instance is created

A screenshot of a computer

Description automatically generated

1. Connect the created EC2 instance using SSH

A screenshot of a computer program

Description automatically generated

1. Update the server using command:

>>sudo apt update

A screen shot of a computer screen

Description automatically generated

1. Install python on the instance using command:

>>sudo apt install python3-pip

A blue screen with white text

Description automatically generated

1. Install pandas as we have to read data from csv and create a dataframe, use command-

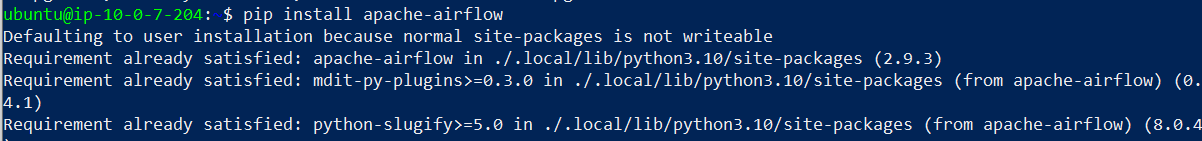
>>pip3 install pandas

A blue screen with white text

Description automatically generated

1. Install apache airflow using command:

>>pip install apache-airflow



1. Check the version of airflow using command:

>>airflow version

A blue screen with white text

Description automatically generated

1. Now, run the airflow webserver on any port, we are using port 8950. Add this port in inbound rules under the custom security group.

>>airflow webserver -p 8950

A blue screen with white text

Description automatically generated

1. Go to a browser and open the link: <http://public> ip of instance:8950. Airflow window should open here.

A screenshot of a computer

Description automatically generated

1. Go to another powershell, connect the EC2 instance and run below command to create username and password to connect airflow console:

>>airflow users create --role Admin --username admin --email admin --firstname admin --lastname admin --password admin

A blue screen with white text

Description automatically generated

1. Under the instance go to airflow directory and create dags folder, airflow\_input folder and airflow\_output folder.

dags folder: this will hold python files

airflow\_input folder: this will hold the csv file which the dag will read from the instance

airflow\_output folder: this will hold the csv file where dag output will be stored

A computer screen shot of a blue screen

Description automatically generated

1. Created dag2.py file which holds the code to read data from csv file located on EC2 instance, create a dataframe, filter the country column and return the output to csv file on instance.

A screenshot of a computer program

Description automatically generated

1. After creating .py file run the airflow scheduler using command:

>>airflow scheduler

This will push the .py file on the airflow console.

A computer screen shot of a blue screen

Description automatically generated

1. Go to airflow console, login using the username and password and open the new DAG created. Run the DAG. We can see on the screen that the DAG ran successfully.

A screenshot of a computer

Description automatically generated

1. Check the graph which shows the tasks connectivity.

A screenshot of a computer

Description automatically generated

1. DAG output will be saved in the output file on EC2 instance.

A blue screen with white text

Description automatically generated