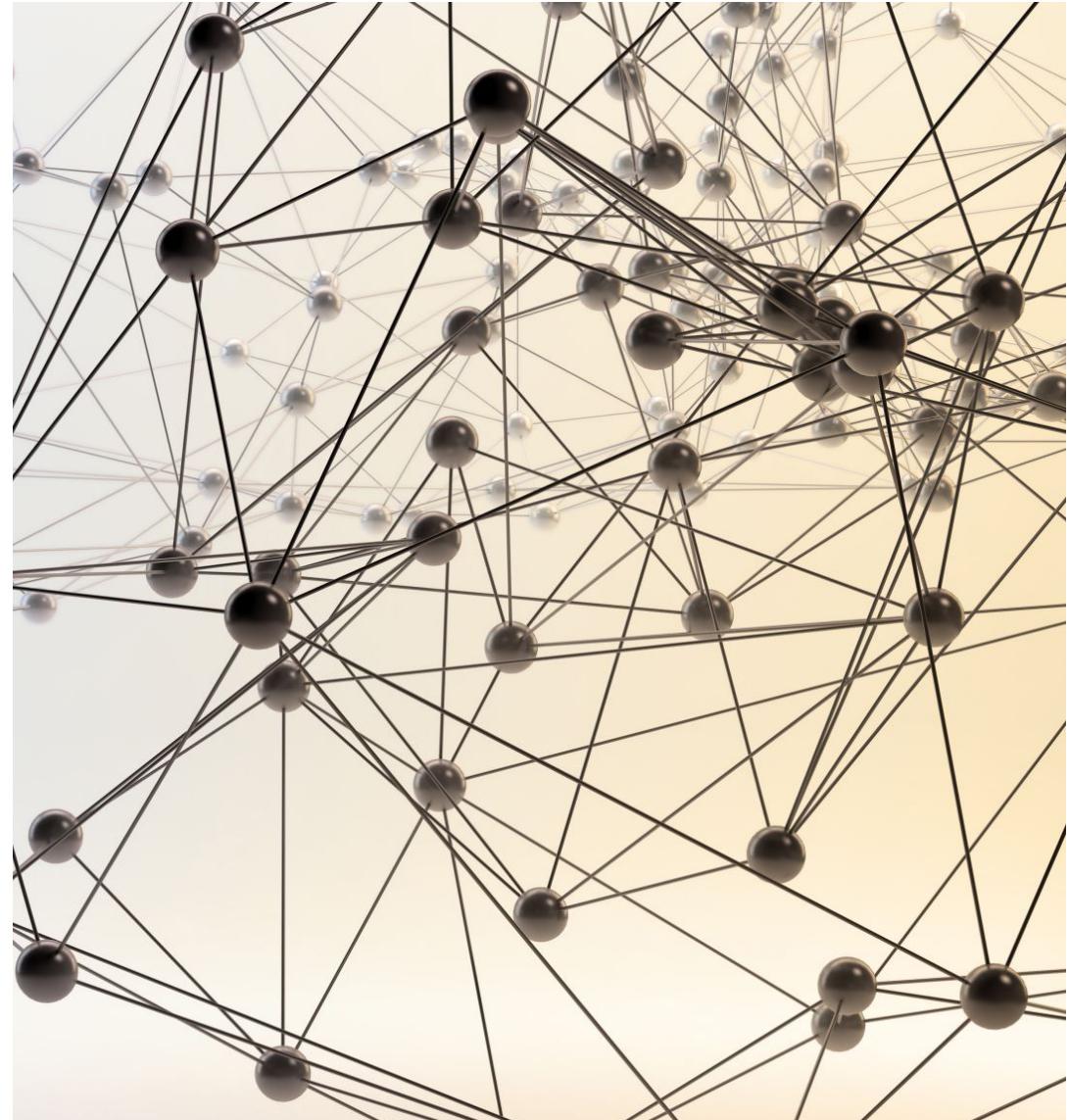




# Batch Load Data with Apache Airflow and Spark

by Andrea Zhang  
2022 Aug

---



# Project Motivation

---

**Objective:** The primary focus for this project is to build a batch processing application that partition a given file based on the given format and save into several distributed files in S3 bucket for analytics and visualization.

For this project, multiple big data applications were used, including Docker, Airflow, Lambda function, Spark (EMR), AWS Glue, AWS Athena and visualization layer represented by Superset. The data source used in the project is public banking data.

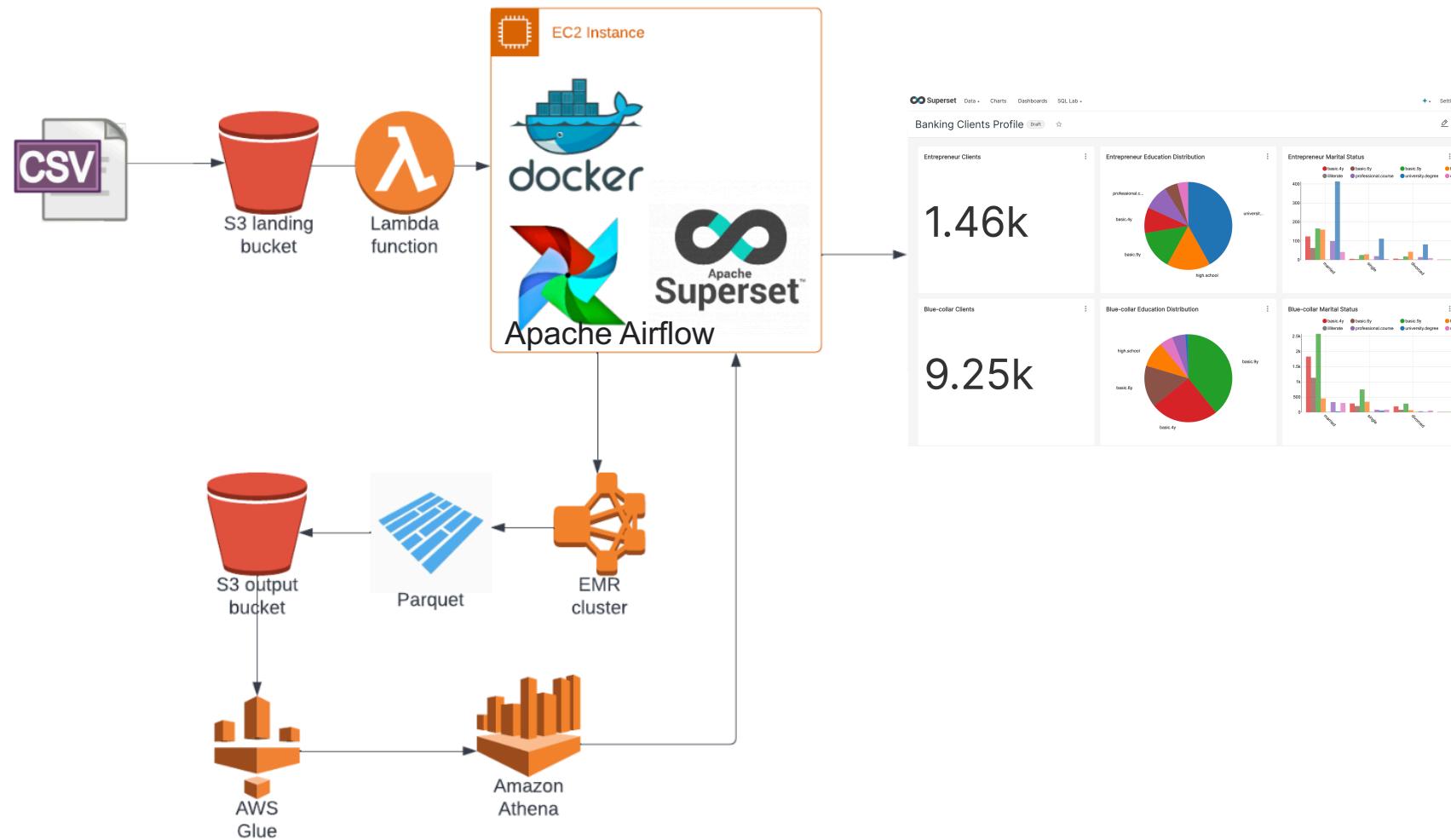
# Conceptual Data Streaming Architecture

Data Collection

Transformation

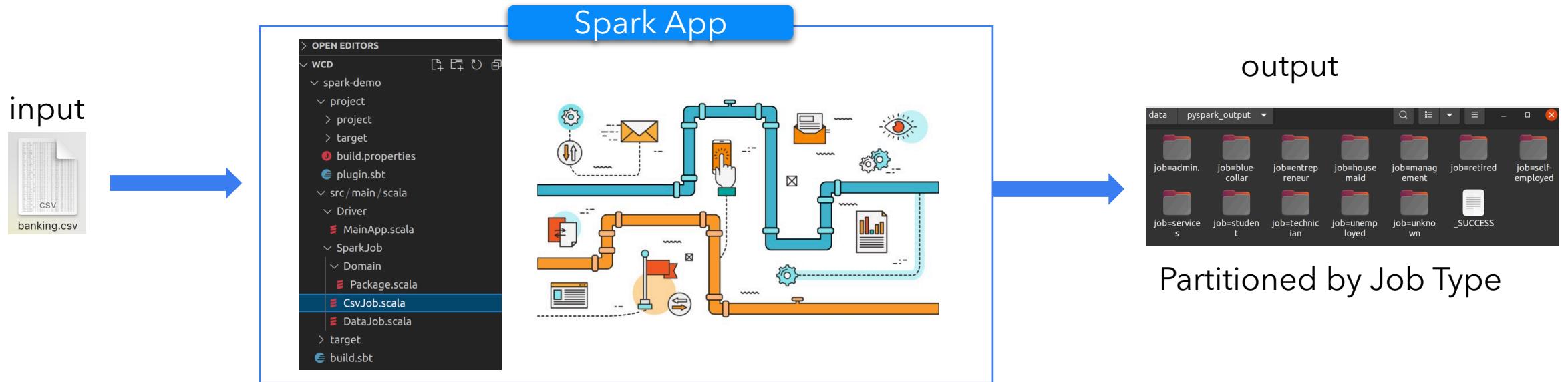
Storage

Visualization



# Data Transformation Spark

Compiling Spark Jobs Built in Scala. The Spark application transforms CSV files into multiple parquet files partitioned by the job type column, and export parquet files to the S3 bucket.



Deploy a Spark job:

- Get the latest project files
- Download dependencies
- Compile and Package Spark application
- Upload .jar file to an S3 bucket
- Spark-submit using the s3 file to the Spark (EMR cluster) master node

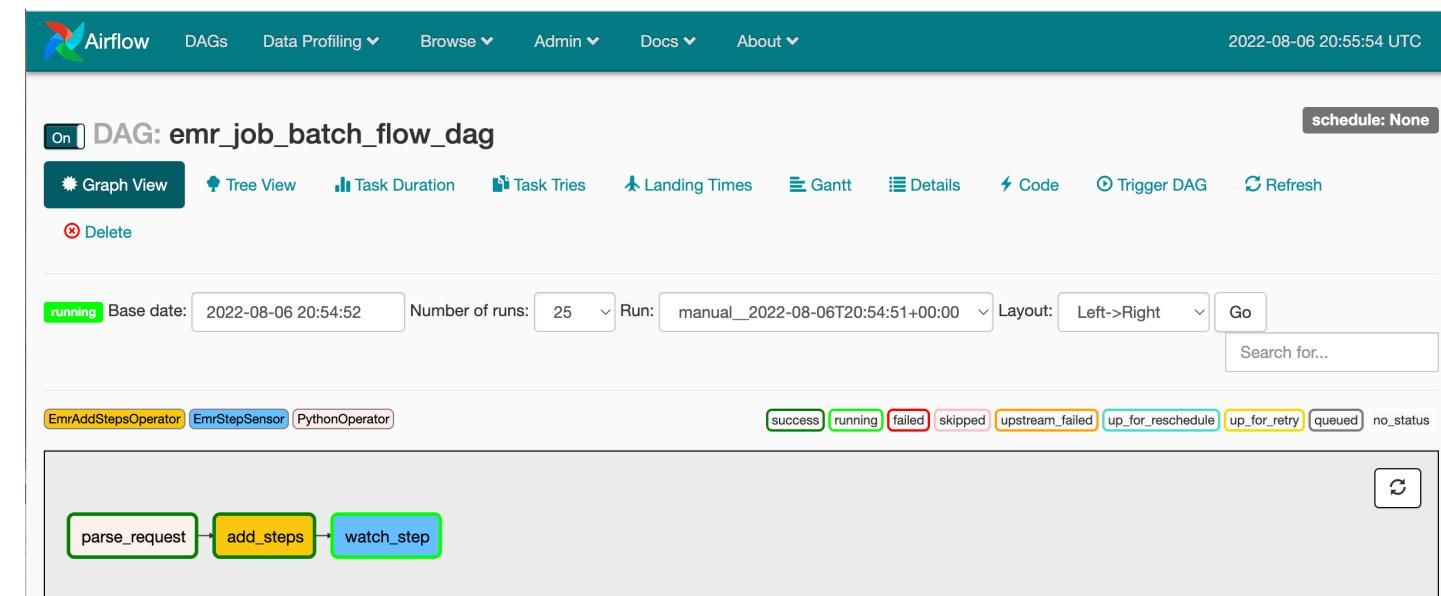
# Data Transformation - Apache Airflow

Airflow enable the execution of a collection of task in a more organized way using DAG.

- Define DAG in Python script and execute the script in a Docker container
- ✓ Schedule workflows
- ✓ Monitor workflows
- ✓ Scalability

**DAGs**

	<b>i</b>	<b>DAG</b>	<b>Schedule</b>	<b>Owner</b>
<input checked="" type="checkbox"/>	<input type="button" value="Off"/>	emr_job_batch_flow_dag	None	wcd_data_engineer
<input checked="" type="checkbox"/>	<input type="button" value="Off"/>	emr_job_flow_manual_steps_dag	None	wcd_data_engineer
<input checked="" type="checkbox"/>	<input type="button" value="Off"/>	tutorial	1 day, 0:00:00	airflow



# Spark Output - Parquet Files

Amazon S3 > Buckets > wcd-data-engineer-batch > output/

output/

**Objects** | **Properties**

**Objects (13)**

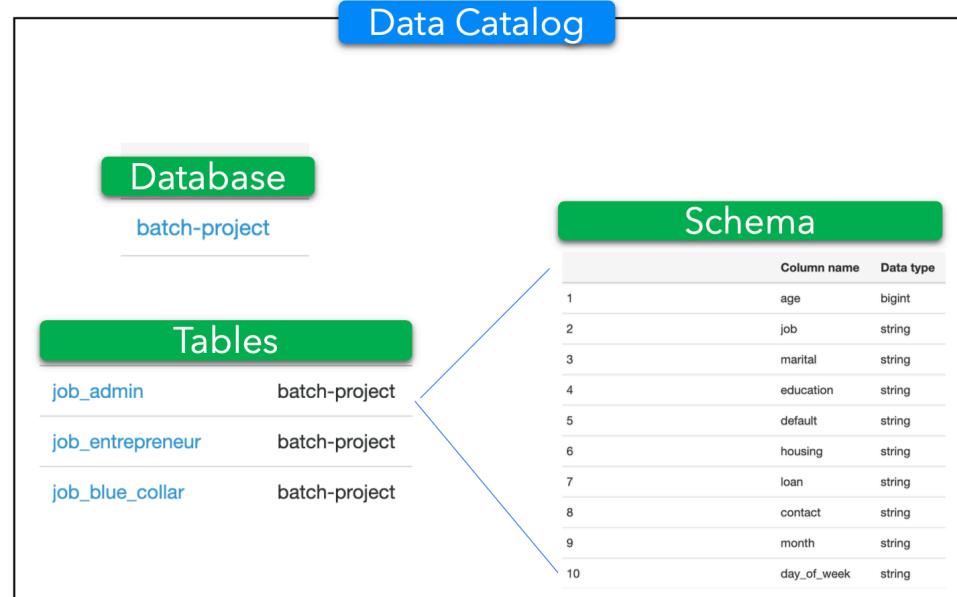
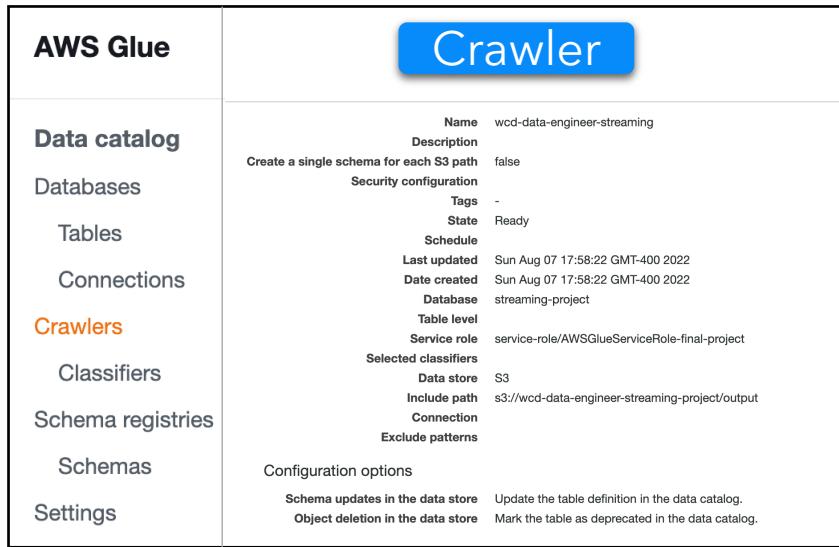
Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your ob

Copy S3 URI  Copy URL  Download  Open  Delete  Actions  Create fo

Find objects by prefix

<input type="checkbox"/>	Name	Type	Last modified
<input type="checkbox"/>	<a href="#">_SUCCESS</a>	-	August 7, 2022, 10:54:48 (UTC-04:00)
<input type="checkbox"/>	<a href="#">job=admin/</a>	Folder	-
<input type="checkbox"/>	<a href="#">job=blue-collar/</a>	Folder	-
<input type="checkbox"/>	<a href="#">job=entrepreneur/</a>	Folder	-
<input type="checkbox"/>	<a href="#">job=housemaid/</a>	Folder	-
<input type="checkbox"/>	<a href="#">job=management/</a>	Folder	-
<input type="checkbox"/>	<a href="#">job=retired/</a>	Folder	-
<input type="checkbox"/>	<a href="#">job=self-employed/</a>	Folder	-
<input type="checkbox"/>	<a href="#">job=services/</a>	Folder	-
<input type="checkbox"/>	<a href="#">job=student/</a>	Folder	-
<input type="checkbox"/>	<a href="#">job=technician/</a>	Folder	-
<input type="checkbox"/>	<a href="#">job=unemployed/</a>	Folder	-
<input type="checkbox"/>	<a href="#">job=unknown/</a>	Folder	-

# Data Integration - AWS Glue



**Glue provides serverless data integration that prepare and combine data for analytics.**

The Glue data Catalog serves as a central metadata repository.

- ✓ Discover and search across multiple data sets using Glue data Catalog.
- ✓ Once the data is cataloged, it is immediately available for search and query using Amazon Athena.

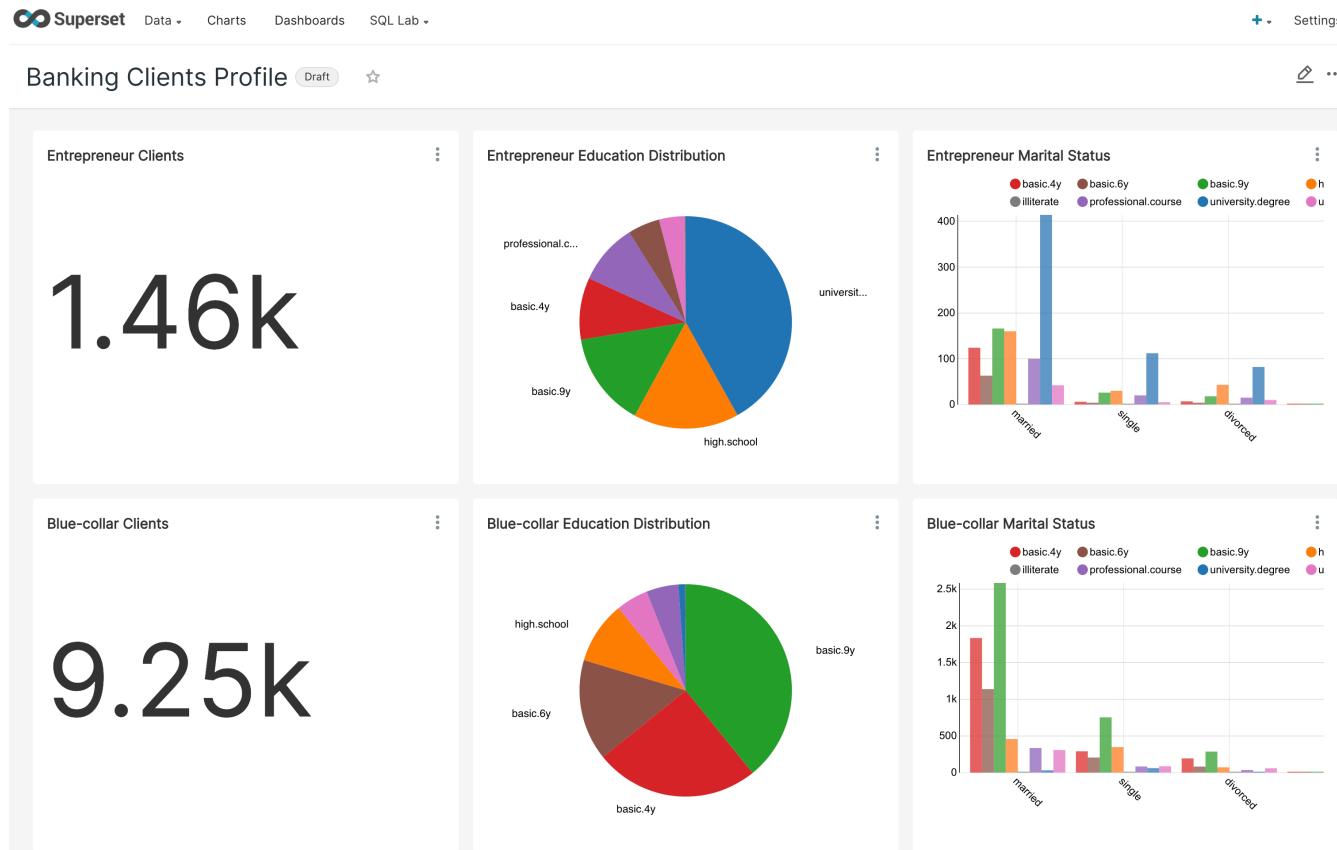
# Data Analytics - AWS Athena

Connect Glue Data Catalog and point to the streaming project database and enable querying the bus data instantly.

The screenshot shows the Amazon Athena Query editor interface. On the left, the sidebar displays the 'Data' section with 'AwsDataCatalog' selected as the data source and 'streaming-project' as the database. Under 'Tables and views', there is one table named 'output'. The main area shows two queries: 'Query 2' (completed) and 'Query 3' (in progress). The completed query's SQL statement is: `SELECT * FROM "streaming-project"."output" limit 10;`. The results table shows 10 rows of data from the 'output' table. The columns are: #, record\_id, id, routeid, directionid, predictablene, seccssincereport, kp, and heading.

#	record_id	id	routeid	directionid	predictablene	seccssincereport	kp	heading
1	1	8156	7	7_1_7	1	28	18	348
2	2	8178	7	7_0_7	1	28	0	168
3	3	8132	7	7_1_7	1	28	33	73
4	4	8138	7	7_1_7	1	28	28	349
5	5	8358	7	7_1_7	1	28	24	344
6	6	8389	7	7_0_7	1	28	0	181
7	7	8130	7	7_0_7	1	28	0	181
8	8	8384	7	7_1_7	1	28	0	230
9	9	8150	7	7_1_7	1	28	0	342

# Data Visualization - Superset



## Ingesting Data From Kafka with Spark Streaming

- Create an AWS EMR and enable big data tools such as Apache spark, Hive, and Hudi.
- Write a Spark job using Scala and assembly into an application jar file
- ✓ Scalable
- ✓ Fault-tolerant
- ✓ Micro batches of data coming from the Kafka topic

# Appendix

# AWS - Set Up Spark Cluster (EMR)

The screenshot shows the AWS EMR console interface. At the top, there's a navigation bar with the AWS logo, a services menu, a search bar containing "Search for services, features, blogs, docs, [Option+S]", and account information for "N. Virginia". Below the navigation bar, there are links for EC2, EMR, S3, IAM, and Lambda.

The main area displays the "Amazon EMR" section. On the left, a sidebar lists "EMR Studio", "EMR Serverless [New]", and "EMR on EC2" (with "Clusters" selected). Other options in the sidebar include "Notebooks", "Git repositories", "Security configurations", "Block public access", "VPC subnets", and "Events".

The main content area shows a cluster named "Cluster: spark\_batch\_cluster" in the "Starting" state. There are three buttons at the top right: "Clone", "Terminate", and "AWS CLI export". Below the cluster name, there are tabs for "Summary", "Application user interfaces", "Monitoring", "Hardware", and "Configuration".

The "Summary" tab is active, displaying the following details:

- ID: j-3GN1BY9MFLTFJ
- Creation date: 2022-08-06 14:46 (UTC-4)
- Elapsed time: 1 minute
- After last step completes: Cluster waits
- Termination protection: On [Change](#)
- Tags: -- [View All / Edit](#)
- Master public DNS: ec2-3-225-221-178.compute-1.amazonaws.com [Connect to the Master Node Using SSH](#)

The "Configuration details" tab is also visible, listing the following configuration parameters:

- Release label: emr-6.5.0
- Hadoop distribution: Amazon 3.2.1
- Applications: Hive 3.1.2, Spark 3.1.2
- Log URI: s3://aws-logs-426788688562-us-east-1/elasticmapreduce/ [File](#)
- EMRFS consistent view: Disabled
- Custom AMI ID: --

# AWS - Set Up Airflow Connection

---

Click "Next: Add Tags"

## Step 5: Add Tags

A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name

A copy of a tag can be applied to volumes, instances or both.

Tags will be applied to all instances and volumes. [Learn more](#) about tagging your Amazon EC2 resources

Key	(128 characters maximum)	Value	(256 characters maximum)
<input type="text" value="USE_CASE"/>		<input type="text" value="Airflow_Demo"/>	
<a href="#">Add another tag</a> (Up to 50 tags maximum)			



# AWS - Configure Network (Key Pair)

Select an existing key pair or create a new key pair X

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance. Amazon EC2 supports ED25519 and RSA key pair types.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about [removing existing key pairs from a public AMI](#).

Create a new key pair ▼

**Key pair type**

RSA  ED25519

**Key pair name**

aws\_batchl\_demo\_key

**Download Key Pair**

 You have to download the **private key file** (\*.pem file) before you can continue. **Store it in a secure and accessible location.** You will not be able to download the file again after it's created.

**Cancel** **Launch Instances**

# AWS - Set Up EC2 Instance

EC2 > Instances > i-006757aa3bc2769c4

## Instance summary for i-006757aa3bc2769c4 [Info](#)

Updated less than a minute ago

Updated less than a minute ago

C [Connect](#) [Instance state ▾](#) [Actions ▾](#)

Instance ID	<a href="#">i-006757aa3bc2769c4</a>	Public IPv4 address	<a href="#">34.235.139.172   open address ↗</a>
IPv6 address	-	Instance state	 Running
Hostname type	IP name: ip-172-31-30-146.ec2.internal	Private IP DNS name (IPv4 only)	<a href="#">ip-172-31-30-146.ec2.internal</a>
Answer private resource DNS name	IPv4 (A)	Instance type	t2.medium

# Update Network Inbound Rules

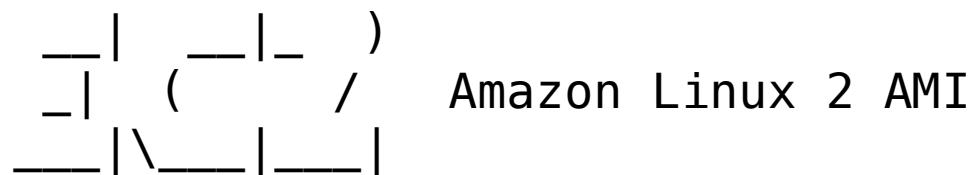
In EC2, white list my IP address and open port 8080. The airflow image will be connected using port 8080.

Inbound rules <a href="#">Info</a>				
Security group rule ID	Type <a href="#">Info</a>	Protocol <a href="#">Info</a>	Port range <a href="#">Info</a>	Source <a href="#">Info</a>
sgr-05495e0d1e98cbdb8	SSH	TCP	22	Custom ▾
-	Custom TCP	TCP	8080	My IP ▾

# SSH to EC2 Using Command Line

---

```
(base) #####@yings-iMac ~ % mv ~/Downloads/aws_batch_demo_key.pem ~/.ssh  
(base) #####@yings-iMac ~ % chmod 400 ~/ssh/aws_batch_demo_key.pem  
chmod: /Users/#####/ssh/aws_batch_demo_key.pem: No such file or directory  
(base) #####@yings-iMac ~ % chmod 400 ~/.ssh/aws_batch_demo_key.pem  
(base) #####@yings-iMac ~ % ssh -i ~/.ssh/aws_batch_demo_key.pem ec2-  
user@34.235.139.172  
The authenticity of host '34.235.139.172 (34.235.139.172)' can't be established.  
ED25519 key fingerprint is SHA256:mbdnrLT6zmkRuAdG3a6LExeJW7zwU9jrACSxIL3D3XE.  
This key is not known by any other names  
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes  
Warning: Permanently added '34.235.139.172' (ED25519) to the list of known hosts.
```



<https://aws.amazon.com/amazon-linux-2/>  
[ec2-user@ip-172-31-30-146 ~]\$

# Airflow DAG Runs

Airflow   DAGs   Data Profiling ▾   Browse ▾   Admin ▾   Docs ▾   About ▾   2022-08-06 20:55:54 UTC

On DAG: emr\_job\_batch\_flow\_dag schedule: None

Graph View Tree View Task Duration Task Tries Landing Times Gantt Details Code Trigger DAG Refresh

✖ Delete

running Base date: 2022-08-06 20:54:52 Number of runs: 25 Run: manual\_2022-08-06T20:54:51+00:00 Layout: Left->Right Go Search for...

EmrAddStepsOperator EmrStepSensor PythonOperator success running failed skipped upstream\_failed up\_for\_reschedule up\_for\_retry queued no\_status

```
graph LR; A[parse_request] --> B[add_steps]; B --> C[watch_step]
```