**AWS EMR Spark Job for Processing** [**Age and sex by ethnic group (grouped total responses), for census usually resident population counts, 2006, 2013, and 2018 Censuses**](https://www3.stats.govt.nz/2018census/Age-sex-by-ethnic-group-grouped-total-responses-census-usually-resident-population-counts-2006-2013-2018-Censuses-RC-TA-SA2-DHB.zip) **Dataset**

**Project Overview**

This document outlines the complete process of running an Apache Spark job on AWS EMR using a dataset (Data8277.csv) stored in Amazon S3. The purpose of the job was to process demographic data and aggregate the total count per Area.

**1. Data Description**

**Source:** ***s3://aws-emr-large-data/Data8277.csv***

**Columns:**

* **Year:** Year of data collection
* **Age:** Age group code
* **Ethnic:** Ethnic group code
* **Sex:** Sex code
* **Area:** Geographic area code
* **count:** Observed count for the demographic slice

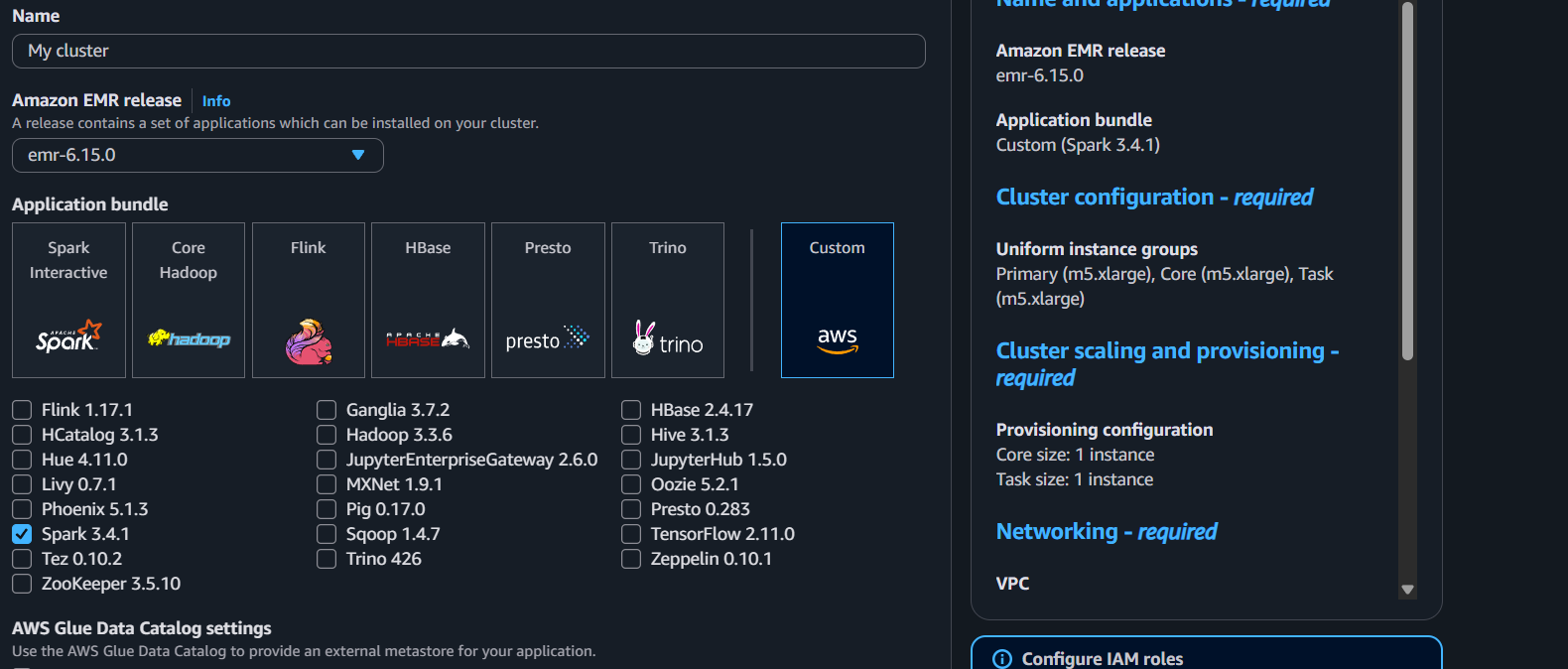
**2. Prerequisites**

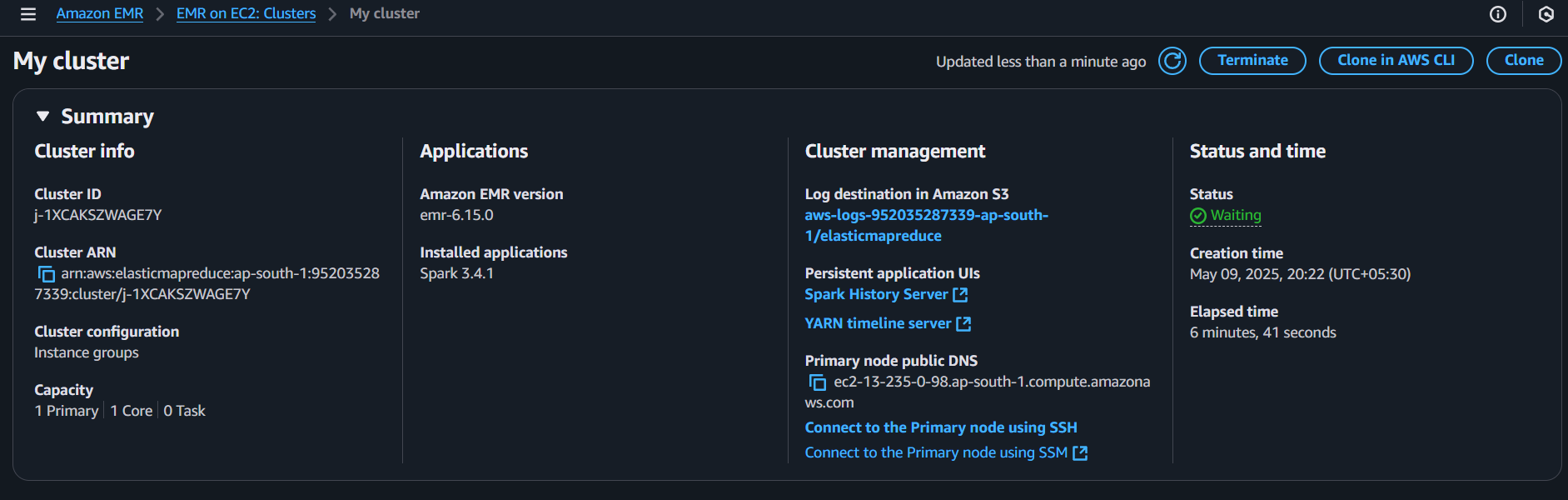
* AWS account with appropriate permissions
* S3 bucket: aws-emr-large-data
* File uploaded: Data8277.csv
* EC2 Key Pair (for cluster access/debugging
* IAM roles: EMR\_DefaultRole, EMR\_EC2\_DefaultRole

**3. EMR Cluster Setup**

**Cluster Configuration:**

* Release Version : EMR 6.12.0
* Applications: Spark
* Instance Type (Master) : m5.xlarge (This is the most budget friendly as aws does not give instances for free.)
* Cluster Name: My cluster





**4. Spark Script**

* **Script Name:** processing.py
* **Location:** s3://aws-emr-large-data/scripts/processing.py

**Functionality:**

* Reads the dataset from S3
* Casts relevant fields to integers
* Drops invalid or null records
* Groups by Area
* Aggregates total count per area
* Writes the result back to S3 in CSV format

***from pyspark.sql import SparkSession***

***from pyspark.sql.functions import col***

***from pyspark.sql.types import IntegerType***

***spark = SparkSession.builder.appName("Data8277Processing").getOrCreate()***

***input\_path = "s3://aws-emr-large-data/Data8277.csv"***

***output\_path = "s3://aws-emr-large-data/output/area\_count\_summary.csv"***

***df = spark.read.option("header", "true").csv(input\_path)***

***df = df.withColumn("Year", col("Year").cast(IntegerType())) \***

***.withColumn("Age", col("Age").cast(IntegerType())) \***

***.withColumn("Ethnic", col("Ethnic").cast(IntegerType())) \***

***.withColumn("Sex", col("Sex").cast(IntegerType())) \***

***.withColumn("Area", col("Area").cast(IntegerType())) \***

***.withColumn("count", col("count").cast(IntegerType()))***

***df\_clean = df.dropna(subset=["Area", "count"])***

***result = df\_clean.groupBy("Area").sum("count").withColumnRenamed("sum(count)", "total\_count")***

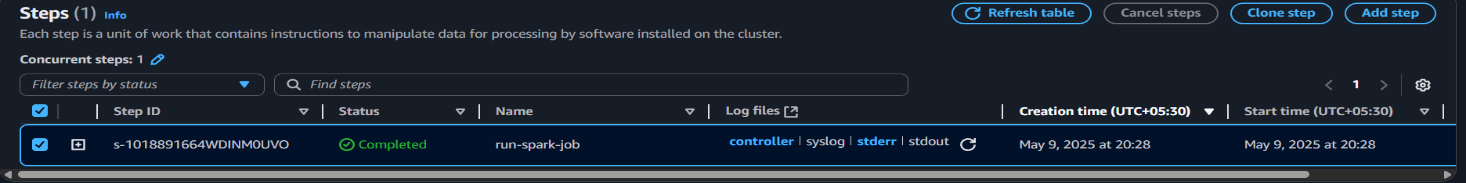
***result.write.mode("overwrite").option("header", "true").csv(output\_path)***

***spark.stop()***

**5. Step Submission**

Once the cluster was in a Waiting state:

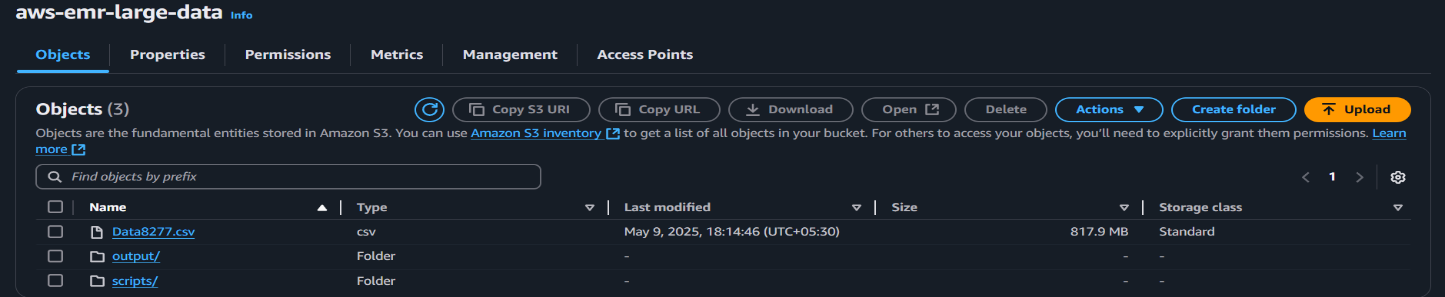
* Step Type: Spark
* Script Location: s3://aws-emr-large-data/scripts/processing.py
* Arguments: s3://aws-emr-large-data/Data8277.csv
* Deploy Mode: Client

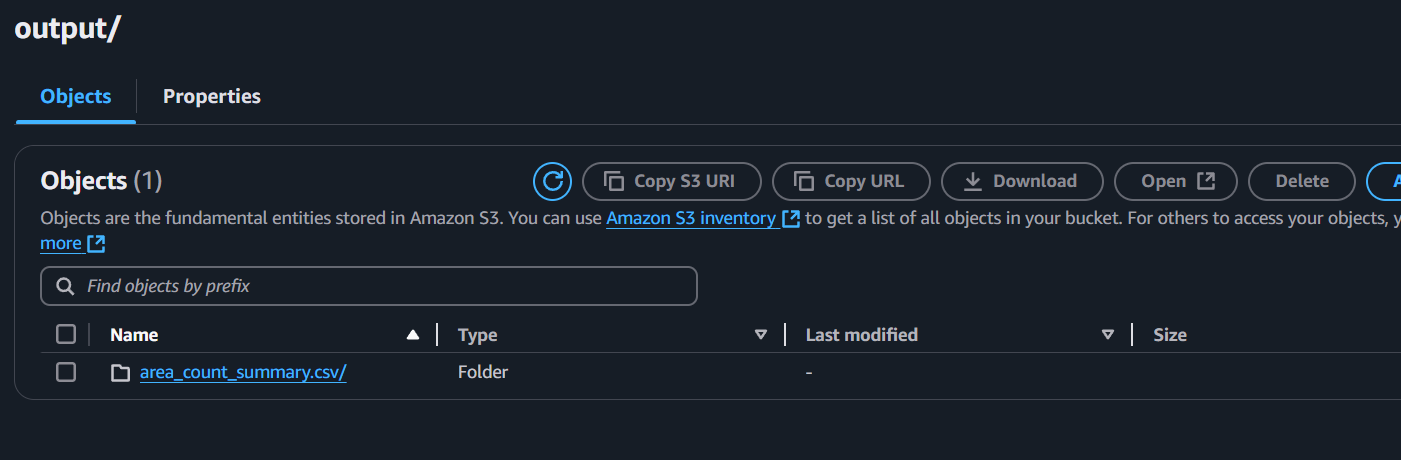


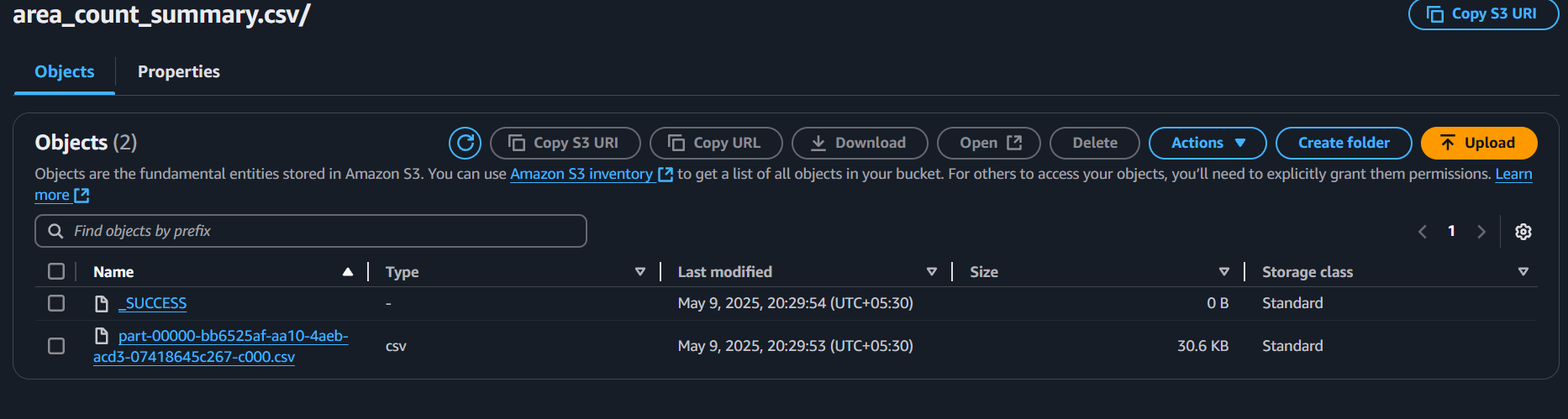
**6. Output**

**Output Path:**

***s3://aws-emr-large-data/output/area\_count\_summary.csv/***







The output consists of one or more CSV part-xxxxx files, each containing summarized data:

Area total\_count

1 795

2 5067

... ...

**7. Cleanup**

* To stay within the Free Tier and avoid charges:
* Terminated EMR cluster after job completion.
* Verified S3 storage usage and cleaned temporary files if needed.

**Conclusion**

* Deploy a minimal-cost Spark cluster on AWS EMR
* Run a data transformation job using PySpark
* Work with real-world structured data from Amazon S3