**Web Server logs processing and Analysis**

This project involves building a data pipeline for processing and analysing web server logs.

**### \*\*Use Case: Building a Web Server Log Analytics Pipeline\*\***

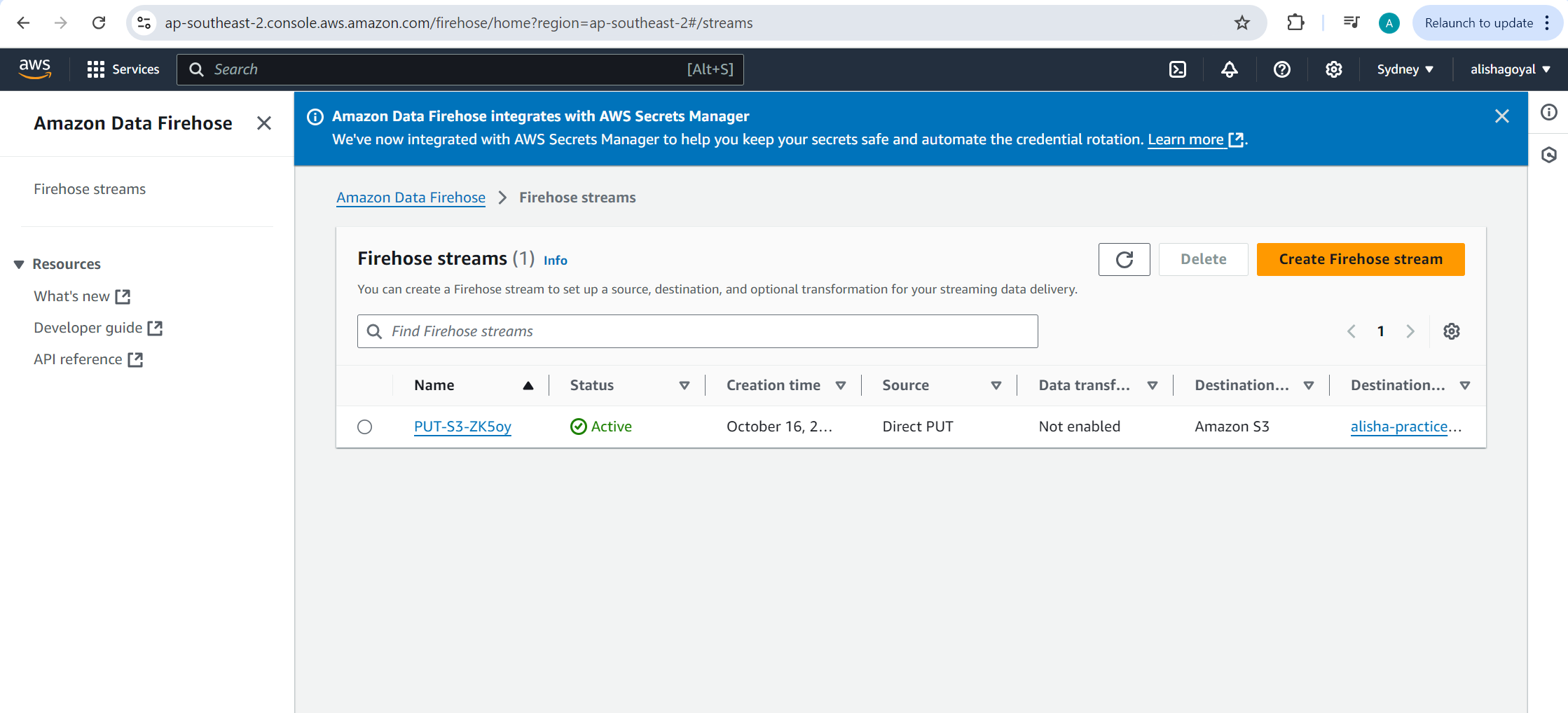
#### \*\*Problem Statement\*\*

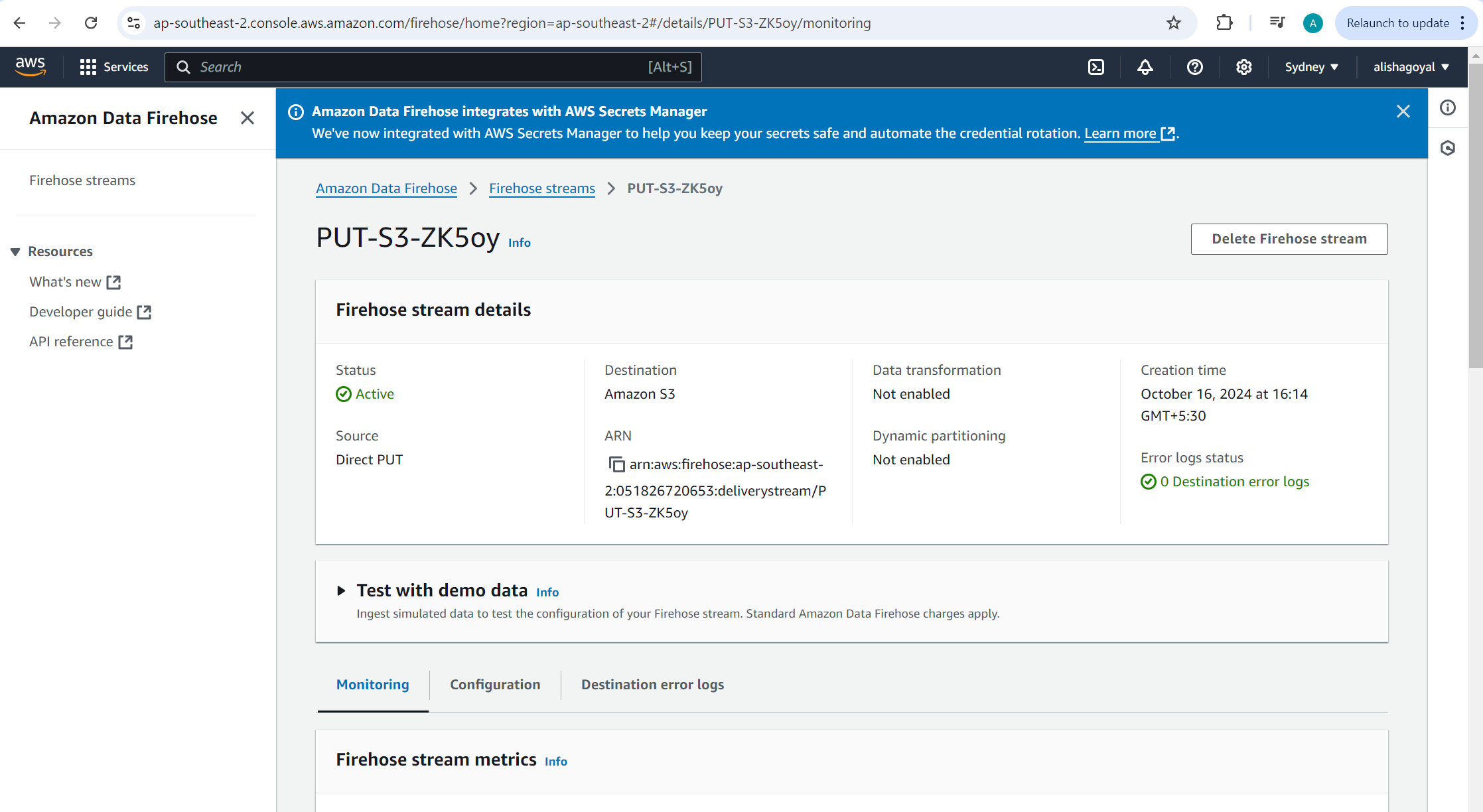
You need to analyse web server logs (e.g., from Apache or Nginx) to gain insights into user behaviour, detect anomalies, and generate reports on metrics such as page views, unique visitors, and error rates. The logs are generated continuously, and you want to process them in near-real-time and store the results for reporting and visualization.

**### 1. \*\*Data Ingestion\*\***

- \*\*Source\*\*: Web server logs (Apache or Nginx) stored as log files.

- \*\*AWS Kinesis Data Firehose\*\*: Set up a delivery stream to ingest streaming data from web servers in near-real-time. You can also use \*\*AWS Lambda\*\* to send the logs to Kinesis.





**Direct Put command-**

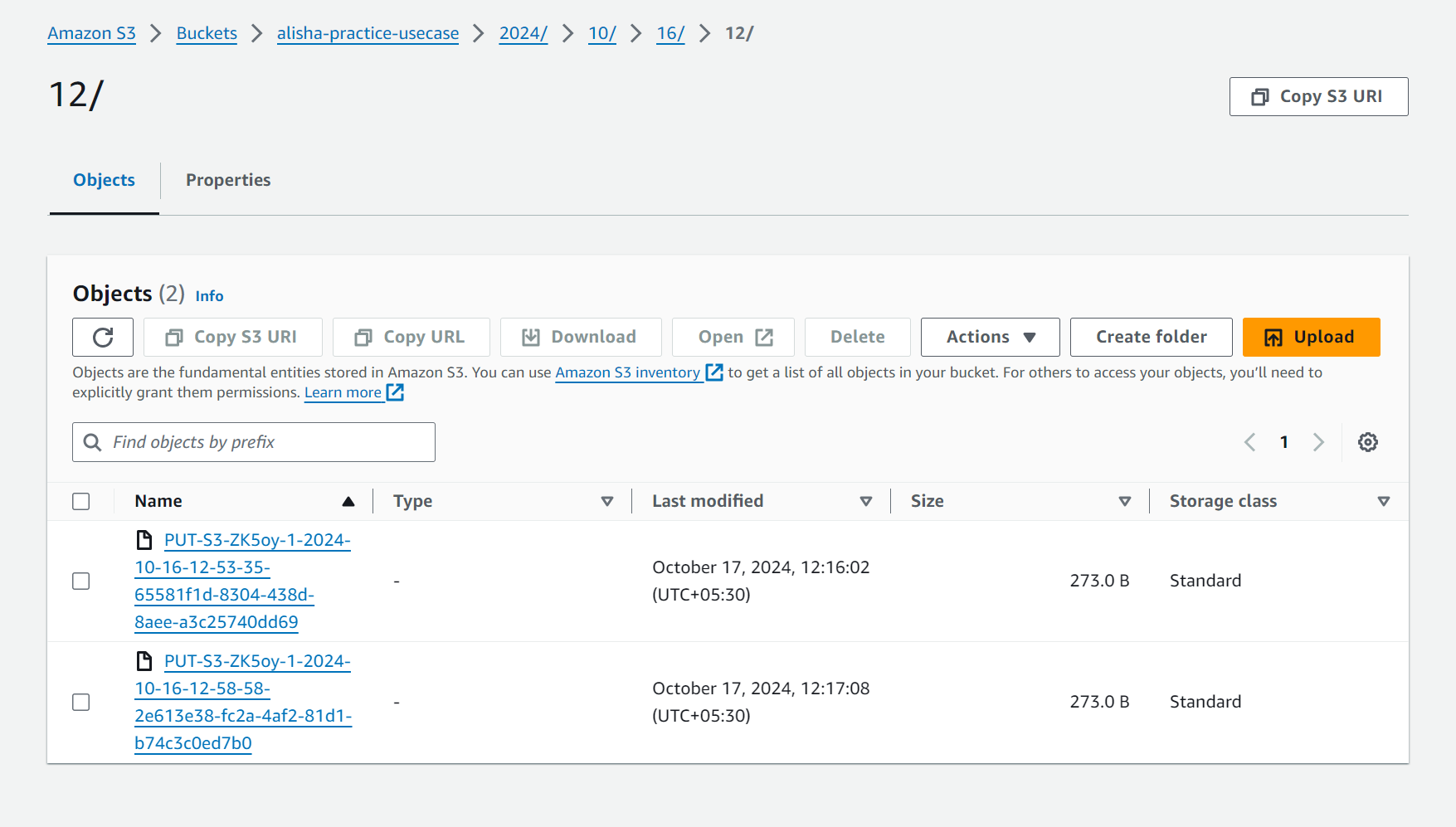
aws firehose put-record-batch --delivery-stream-name PUT-S3-ZK5oy --region ap-southeast-2 --records '[{"Data":"eyJ0aW1lc3RhbXAiOiIyMDI0LTEwLTE2VDEyOjM0OjU2WiIsImNsaWVudF9pcCI6IjE5Mi4xNjguMS4xIiwicmVxdWVzdCI6IkdFVCAvaW5kZXguaHRtbCBIVFRQLzEuMSIsInN0YXR1cyI6MjAwLCJyZXNwb25zZV90aW1lIjowLjEyM30K"}, {"Data":"eyJ0aW1lc3RhbXAiOiIyMDI0LTEwLTE2VDEyOjM1OjU2WiIsImNsaWVudF9pcCI6IjE5Mi4xNjguMS4yIiwicmVxdWVzdCI6IlBPU1QgL2FwaS91cGxvYWQgSFRUUC8xLjEiLCJzdGF0dXMiOjIwMSwicmVzcG9uc2VfdGltZSI6MC40NTZ9Cg=="}]'

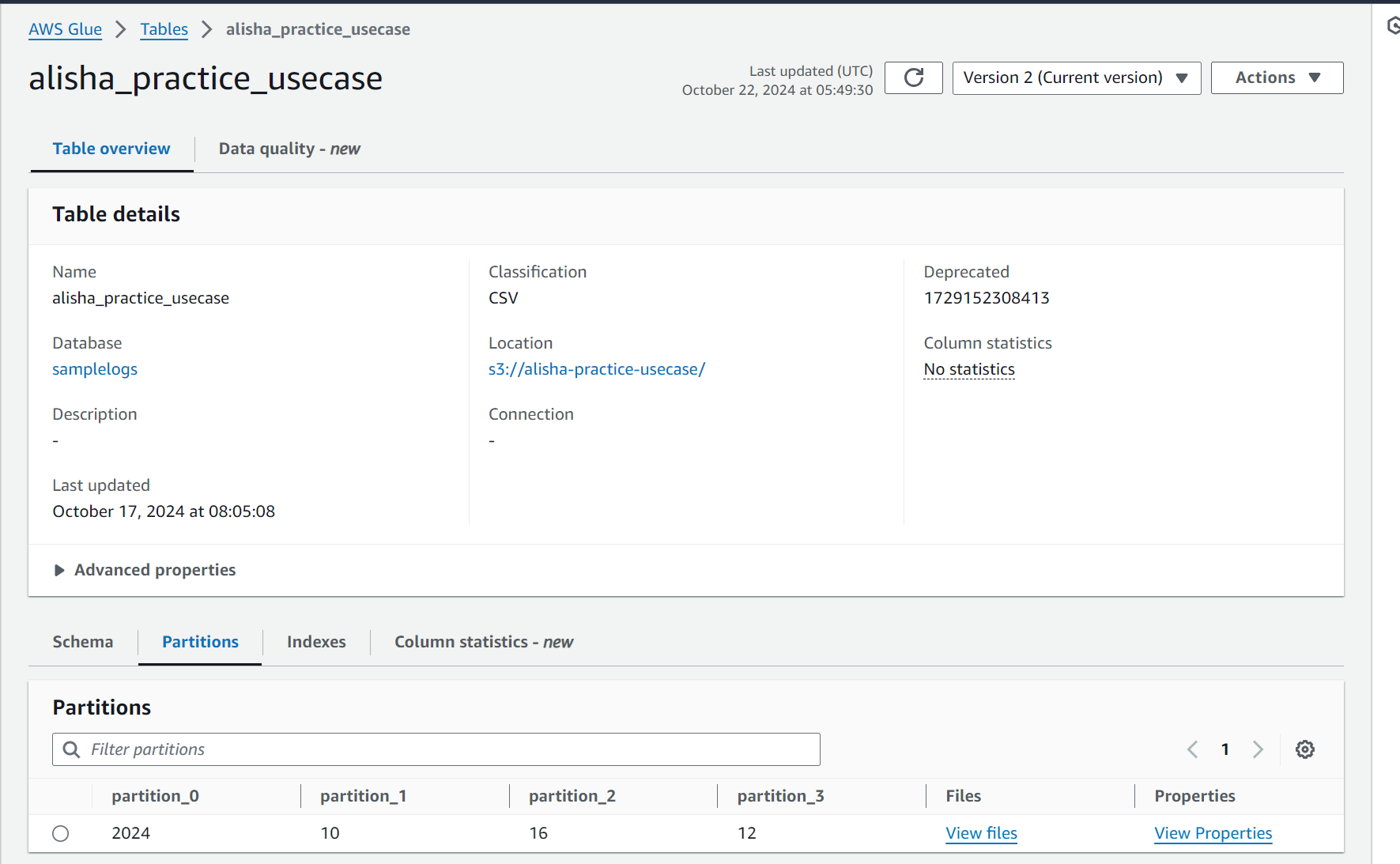
We need send data in encoded form in Kinesis Data Firehose and it will be decoded before delivering to destination.

**### 2. \*\*Data Storage\*\***

- \*\*Amazon S3\*\*: Store raw logs in S3 buckets for archival purposes.

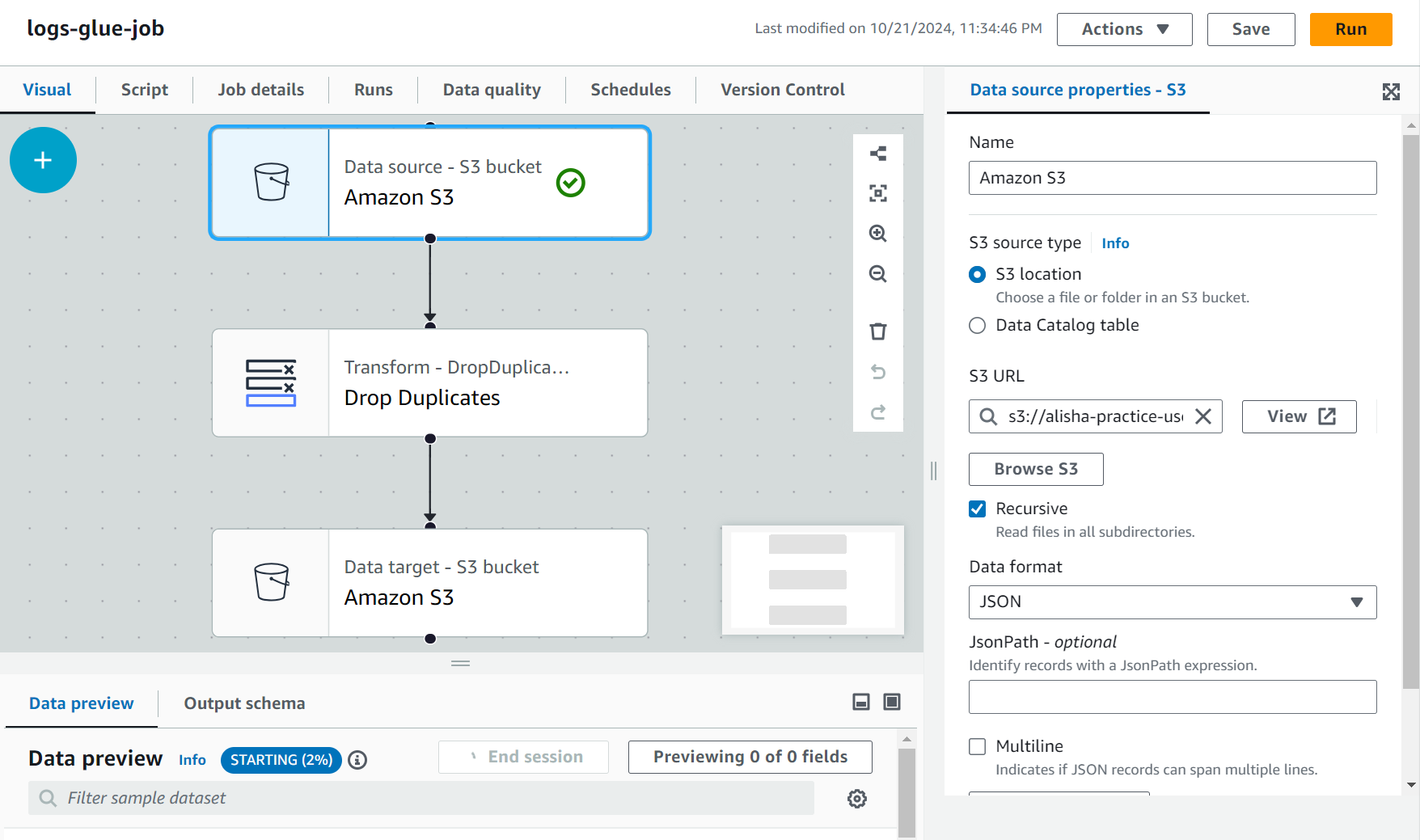
- \*\*AWS Glue Catalog\*\*: Use Glue to catalog the raw logs, creating a schema for easy querying.

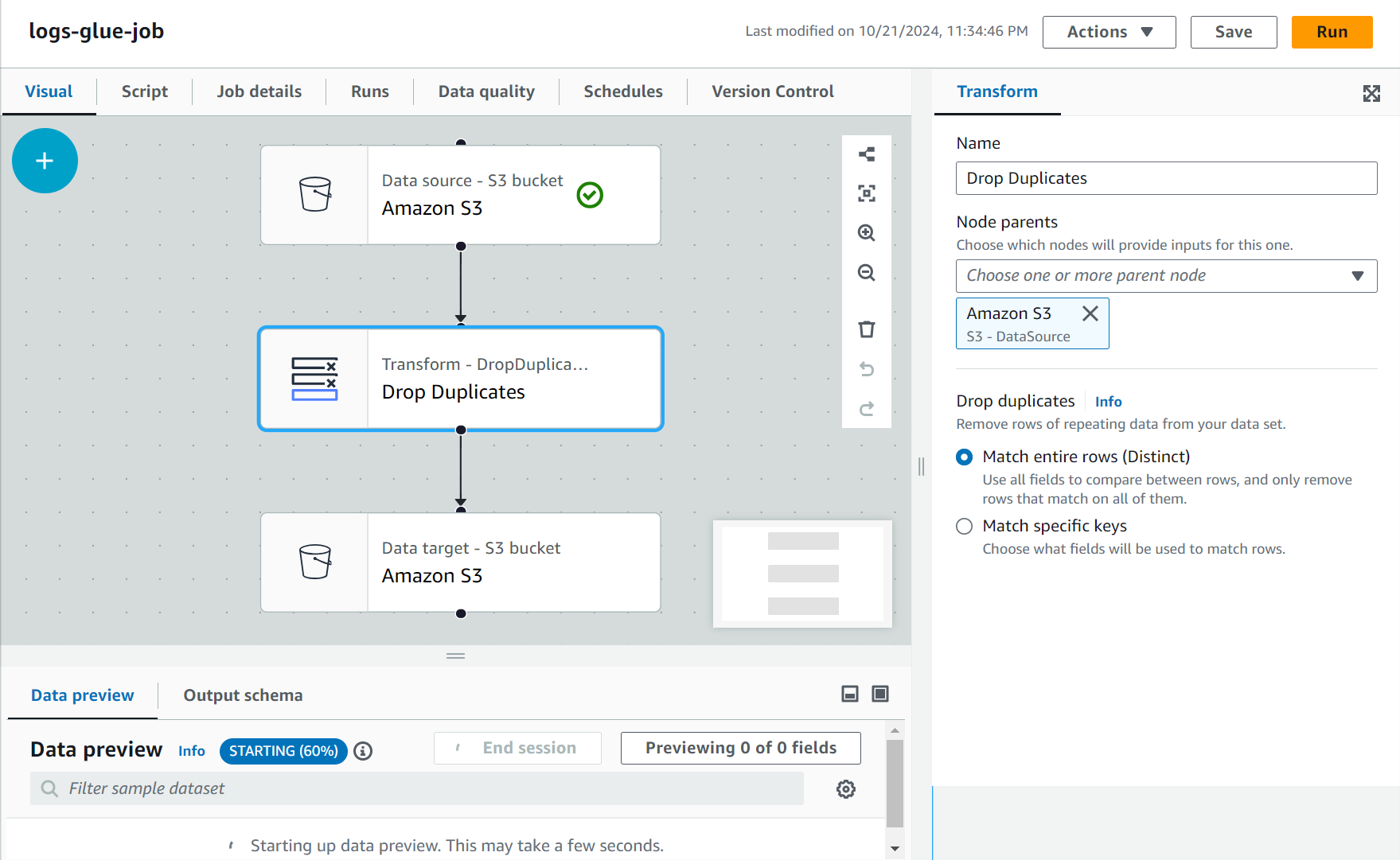


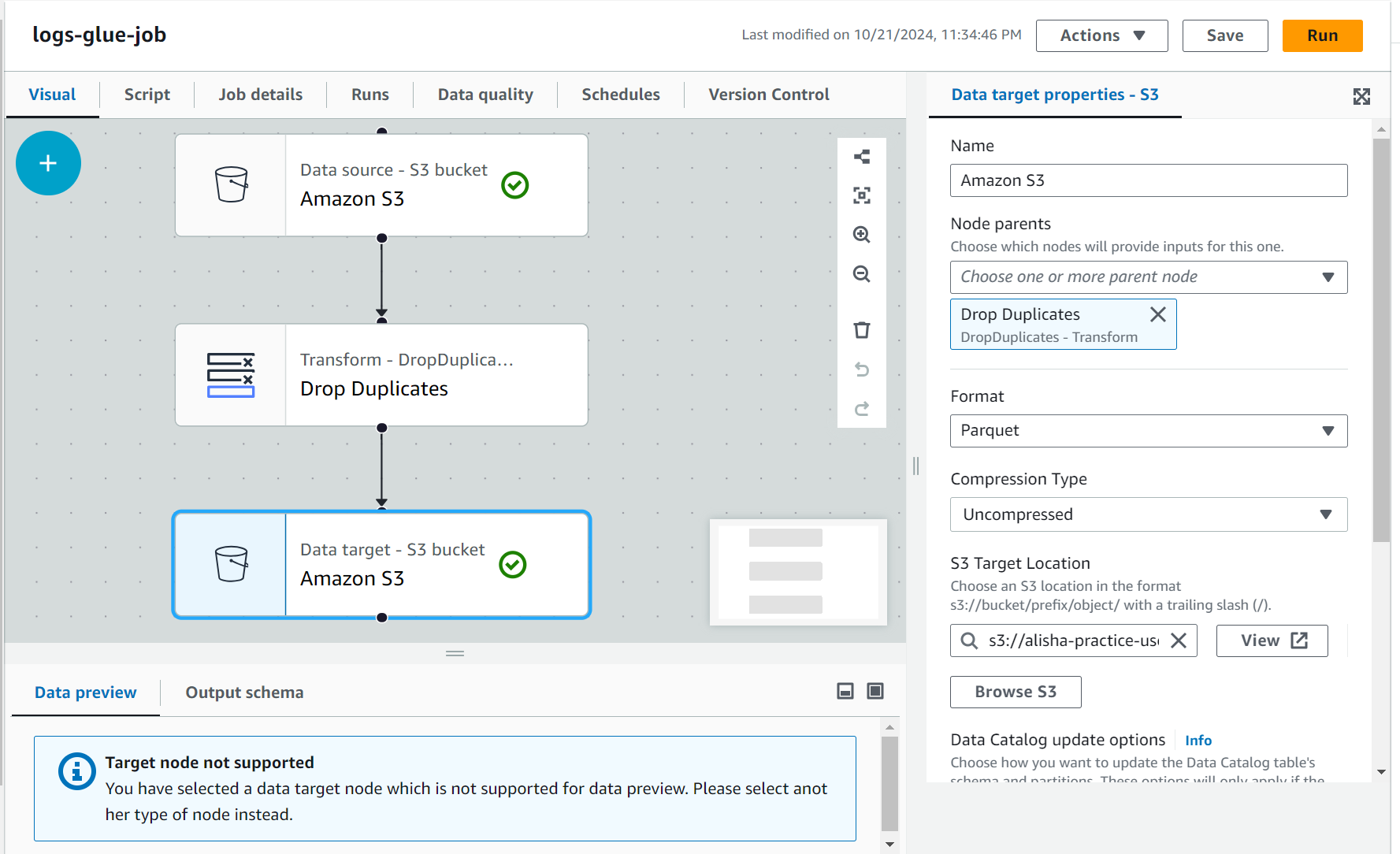


**### 3. \*\*Data Transformation\*\***

- \*\*AWS Glue ETL Jobs\*\*: Set up Glue jobs to transform the log data (e.g., parse IP addresses, timestamps, HTTP status codes) and convert the data into structured formats like Parquet or ORC for efficient querying.







**Glue Job Script-**

import sys

from awsglue.transforms import \*

from awsglue.utils import getResolvedOptions

from pyspark.context import SparkContext

from awsglue.context import GlueContext

from awsglue.job import Job

from awsglue.dynamicframe import DynamicFrame

from pyspark.sql import functions as SqlFuncs

args = getResolvedOptions(sys.argv, ['JOB\_NAME'])

sc = SparkContext()

glueContext = GlueContext(sc)

spark = glueContext.spark\_session

job = Job(glueContext)

job.init(args['JOB\_NAME'], args)

# Script generated for node Amazon S3

AmazonS3\_node1729147662024 = glueContext.create\_dynamic\_frame.from\_options(format\_options={"multiLine": "false"}, connection\_type="s3", format="json", connection\_options={"paths": ["s3://alisha-practice-usecase"], "recurse": True}, transformation\_ctx="AmazonS3\_node1729147662024")

# Script generated for node Drop Duplicates

DropDuplicates\_node1729148140482 = DynamicFrame.fromDF(AmazonS3\_node1729147662024.toDF().dropDuplicates(), glueContext, "DropDuplicates\_node1729148140482")

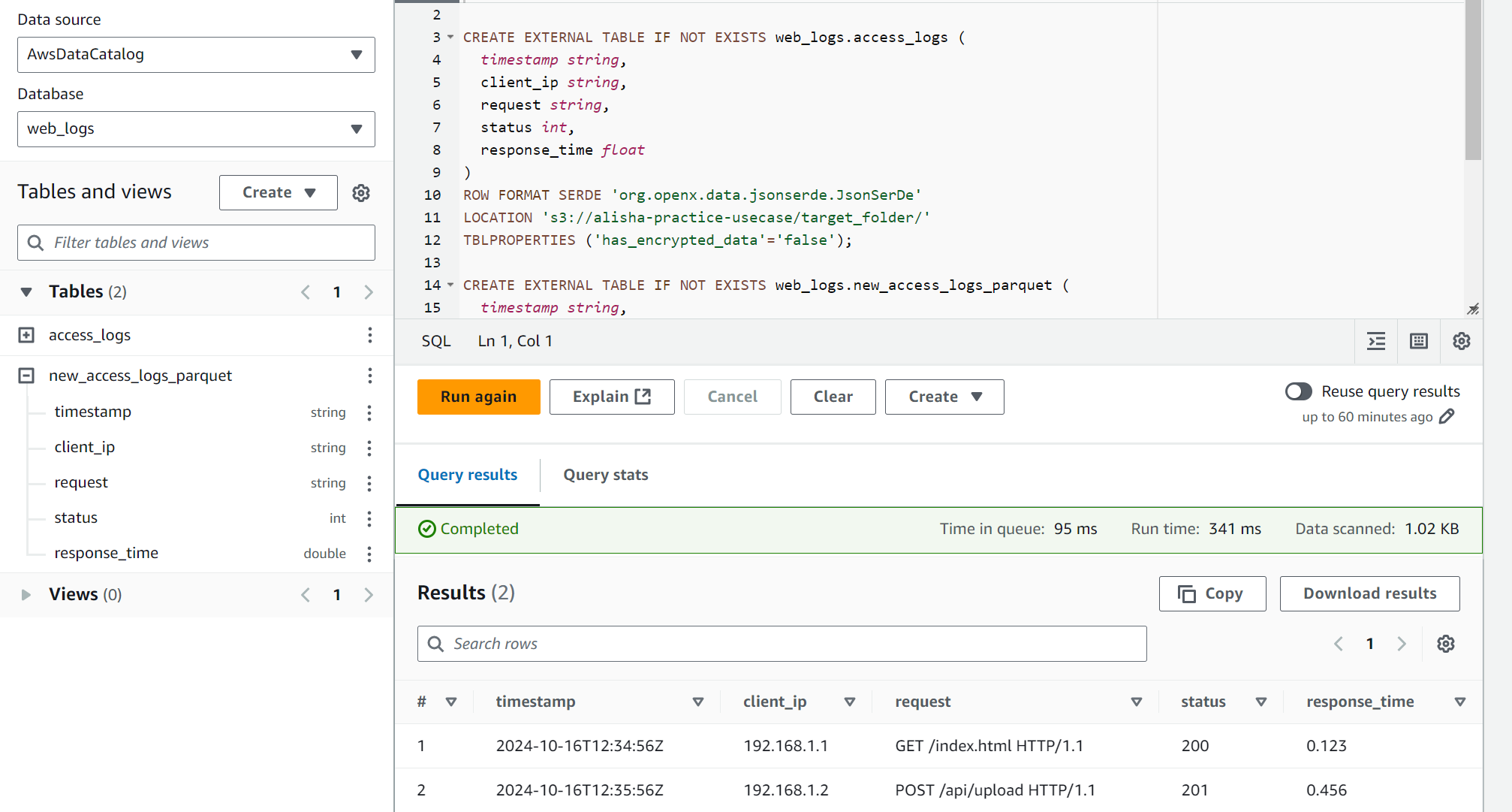
# Script generated for node Amazon S3

AmazonS3\_node1729148147003 = glueContext.write\_dynamic\_frame.from\_options(frame=DropDuplicates\_node1729148140482, connection\_type="s3", format="glueparquet", connection\_options={"path": "s3://alisha-practice-usecase/target\_folder/", "partitionKeys": []}, format\_options={"compression": "uncompressed"}, transformation\_ctx="AmazonS3\_node1729148147003")

job.commit()

**### 4. \*\*Data Processing\*\***

- \*\*Amazon Athena\*\*: Use Athena to run SQL queries on the transformed log data stored in S3.



**Athena Queries-**

CREATE DATABASE web\_logs;

CREATE EXTERNAL TABLE IF NOT EXISTS web\_logs.access\_logs (

timestamp string,

client\_ip string,

request string,

status int,

response\_time float

)

ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'

LOCATION 's3://alisha-practice-usecase/target\_folder/'

TBLPROPERTIES ('has\_encrypted\_data'='false');

CREATE EXTERNAL TABLE IF NOT EXISTS web\_logs.new\_access\_logs\_parquet (

timestamp string,

client\_ip string,

request string,

status int,

response\_time double

)

STORED AS PARQUET

LOCATION 's3://alisha-practice-usecase/target\_folder/'

TBLPROPERTIES ('parquet.compression'='SNAPPY');

Select \* from new\_access\_logs\_parquet;

DROP TABLE IF EXISTS web\_logs.access\_logs\_parquet;

#### \*\*Technologies Used\*\*:

- \*\*AWS Kinesis\*\* for data ingestion.

- \*\*Amazon S3\*\* for log storage.

- \*\*AWS Glue\*\* for ETL and data cataloging.

- \*\*Amazon Athena\*\* for querying and analysis.

#### \*\*End Goal\*\*:

- A fully automated pipeline that ingests web server logs, stores and processes them, and provides a real-time data to further dashboard to monitor user traffic, error rates, and performance metrics.

---