AWS PROJECT

KARIVENA VINAY KUMAR

LOG AGGREGATION AND ANALYSIS

FINAL PROJECT

2278506

2024

kvk9110788277@gmail.com

TABLE OF CONTENTS

GITHUB_REPO !

THE INTRODUCTION :-

+701-373-1005

In today's digital landscape, the generation of vast amounts of log data from various systems and applications is inevitable. These logs contain valuable information about system health, user interactions, errors, and performance metrics, which can provide crucial insights for troubleshooting, optimization, and decision-making processes. However, efficiently managing, analyzing, and deriving actionable insights from this log data can be challenging without a robust system in place.

The "Log Aggregation and Analysis" project aims to address this challenge by developing a comprehensive system for collecting, analyzing, and storing log data using AWS services. Leveraging the scalability, flexibility, and ease of use of AWS cloud services, this project establishes a reliable pipeline for log data ingestion, real-time analysis, and historical querying.

THE CORE COMPONENTS OF THE SYSTEM INCLUDE :-

Data Source: A dataset sourced from Kaggle, representing simulated log data, stored in Amazon S3.

Data Preprocessing: Utilizing an EC2 instance running a Python script to preprocess the raw log data and convert it into a structured format.

Real-Time Analysis: Employing Amazon Kinesis Data Analytics to perform real-time analysis on streaming log data, enabling immediate insights into system behavior and performance.

2024

Data Storage: Leveraging Amazon S3 as a scalable and durable storage solution for both streaming and historical log data.

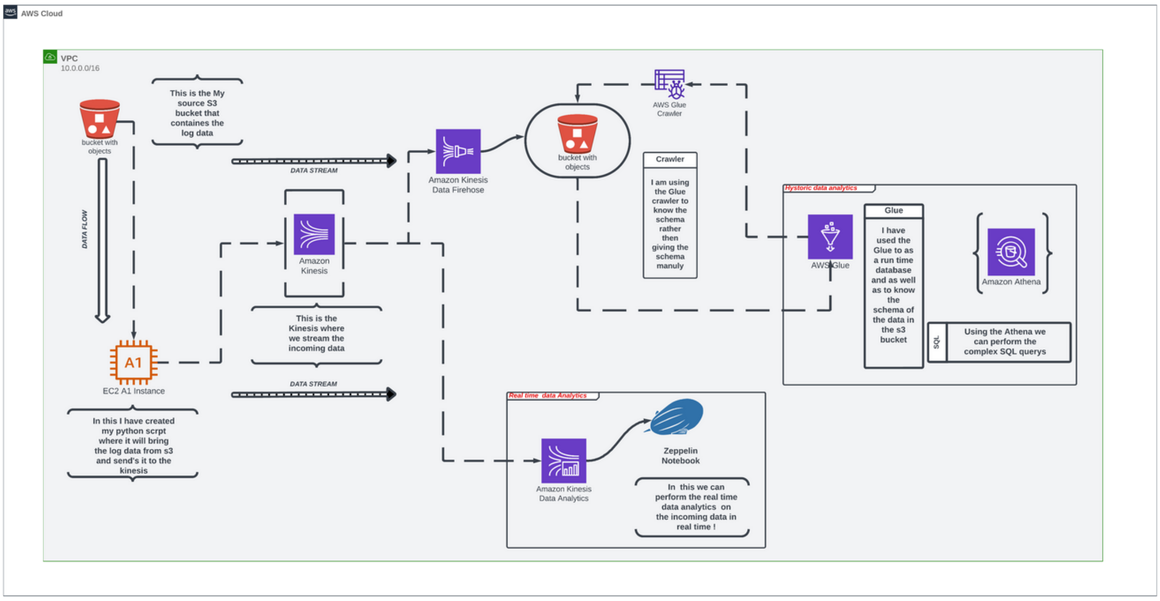
Data Cataloging and ETL: Utilizing AWS Glue for data cataloging, schema discovery, and performing Extract, Transform, Load (ETL) operations to prepare data for querying.

Historical Data Querying: Utilizing Amazon Athena to query historical log data stored in Amazon S3, enabling ad-hoc analysis and insights into past system behavior.

LOG Aggregation And Analysis

By implementing this log aggregation and analysis system, organizations can gain valuable insights into their systems' health, performance, and user behavior in real-time and over historical periods. This project serves as a foundational framework that can be extended and customized to suit specific use cases and requirements, empowering organizations to make informed decisions and optimize their operations effectively.

THE ARCHITECTURE:-



This is the My source S3 bucket that containes the log data

I am using the Glue crawler to know the schema rather then giving the schema manuly

This is the Kinesis where we stream the incoming data

In this I have created my python scrpt where it will bring the log data from s3 and send's it to the kinesis

I have used the Glue to as a run time database and as well as to know the schema of the data in the s3 bucket

Using the Athena we can perform the complex SQL querys

In this we can perform the real time data analytics on the incoming data in real time !

Diagram illustrating the architecture of the log aggregation and analysis system.

The architecture of the "Log Aggregation and Analysis" system is designed to provide a scalable, real-time, and comprehensive solution for collecting, analyzing, and storing log data. Leveraging a combination of AWS services, the architecture ensures reliability, flexibility, and ease of management. Below is a detailed description of each component in the architecture:

Components:-

Data Source (Kaggle Dataset in Amazon S3):

The project starts with a dataset sourced from Kaggle, containing simulated log data representing various system events, errors, and user interactions.

kaggle

The dataset is stored in Amazon S3, a highly scalable and durable object storage service.

Data Preprocessing (EC2 Instance with Python Script):

An EC2 instance is utilized to run a Python script responsible for preprocessing the raw log data.

The Python script parses the log data, performs necessary transformations, and converts it into a structured format (e.g., JSON).

This structured data is then sent to Amazon Kinesis Data Firehose for streaming to downstream services

Real-Time Log Analysis (Amazon Kinesis Data Analytics):

Amazon Kinesis Data Analytics is used for real-time analysis of streaming log data.

It provides capabilities to run SQL queries on the streaming data, enabling immediate insights into system behavior, performance metrics, and anomalies.

Real-time analytics results can be visualized and analyzed in tools like Apache Zeppelin notebooks.

Data Storage (Amazon S3):

Amazon S3 serves as the primary storage solution for both streaming and historical log data.

Streaming log data sent through Amazon Kinesis Data Firehose is directly stored in Amazon S3 in near real-time.

Historical log data can be archived and stored in Amazon S3 for long-term retention and querying.

Data Cataloging and ETL (AWS Glue):

AWS Glue is utilized for data cataloging, schema discovery, and performing Extract, Transform, Load (ETL) operations on the log data.

Glue crawlers automatically discover the schema of streaming log data stored in Amazon S3 and create corresponding metadata tables.

Glue ETL jobs can be configured to perform transformations on the data, preparing it for querying and analysis.

Historical Data Querying (Amazon Athena):

Amazon Athena is used for querying historical log data stored in Amazon S3.

It allows users to run ad-hoc SQL queries on the log data without the need for managing infrastructure.

Athena integrates seamlessly with AWS Glue data catalog, enabling easy discovery and querying of log data tables.

This architecture provides a scalable and comprehensive solution for log aggregation and analysis, enabling organizations to gain real-time insights into their systems' behavior and performance while also facilitating historical data analysis for long-term optimization and decision-making.

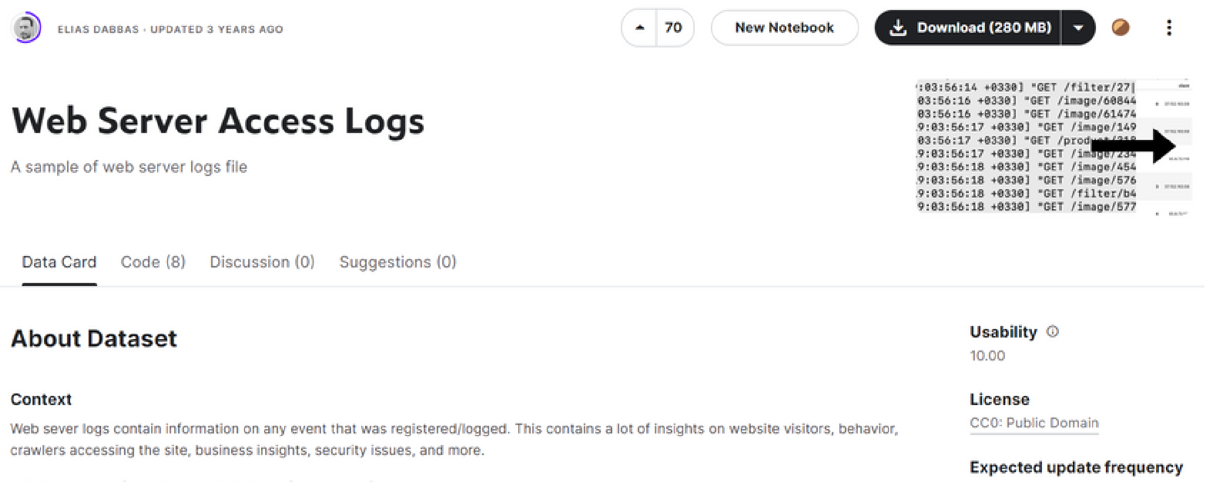
IMPLEMENTATION STEPS:-

The implementation of the "Log Aggregation and Analysis" system involves several sequential steps to set up and configure the AWS services, ingest data, perform real-time analysis, and enable historical data querying. Below are the detailed implementation steps:

DATA INGESTION FROM KAGGLE DATASET TO AMAZON S3:

Download the log dataset from Kaggle or any other source and store it in an Amazon S3 bucket.

Ensure appropriate permissions are set for accessing the S3 bucket.



Web Server Access Logs

A sample of web server logs file

```
:03:56:14 +0330] "GET /filter/27|
03:56:16 +0330] "GET /image/60844
03:56:16 +0330] "GET /image/61474
9:03:56:17 +0330] "GET /image/149
03:56:17 +0330] "GET /prod
9:03:56:17 +0330] "GET /image/234
9:03:56:18 +0330] "GET /image/454
9:03:56:18 +0330] "GET /image/576
9:03:56:18 +0330] "GET /filter/b4
9:03:56:18 +0330] "GET /image/577
```

About Dataset

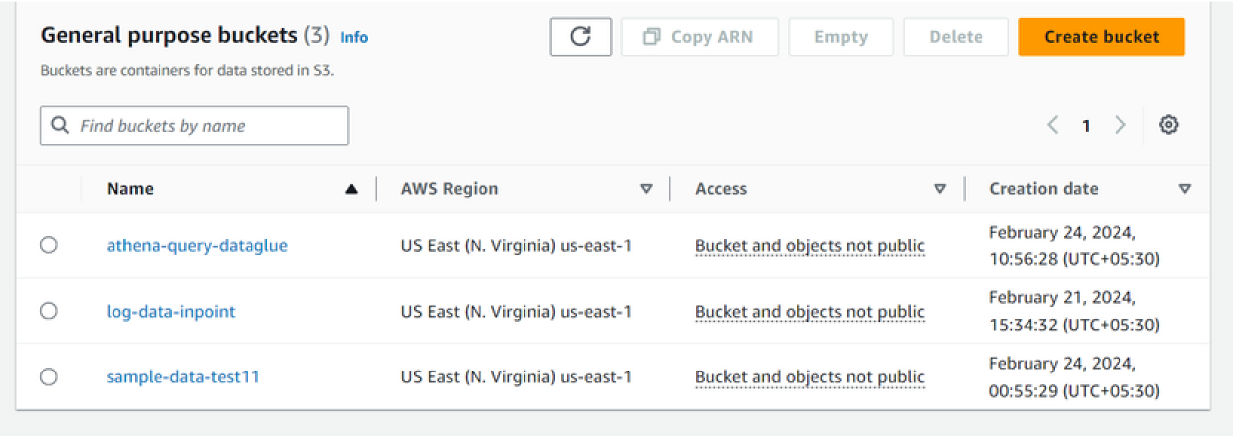
Context

Web sever logs contain information on any event that was registered/logged. This contains a lot of insights on website visitors, behavior, crawlers accessing the site, business insights, security issues, and more.

Usability 10.00

License CC0: Public Domain

Expected update frequency



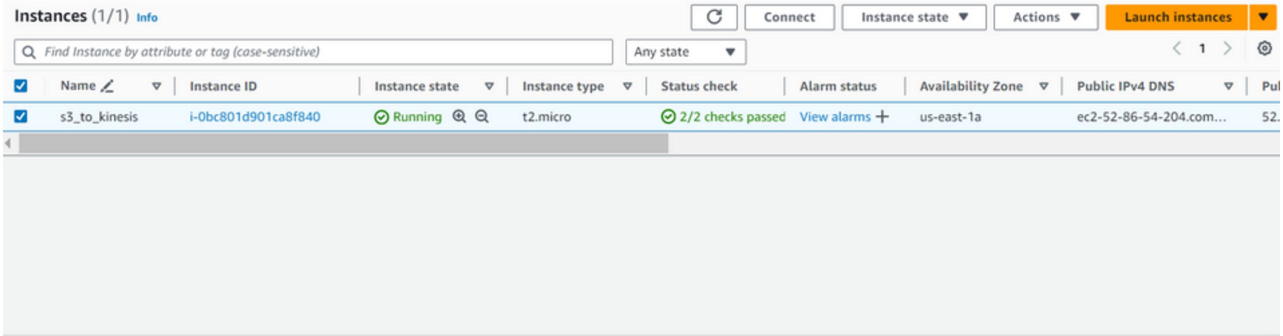
General purpose buckets (3) Info

Buckets are containers for data stored in S3.

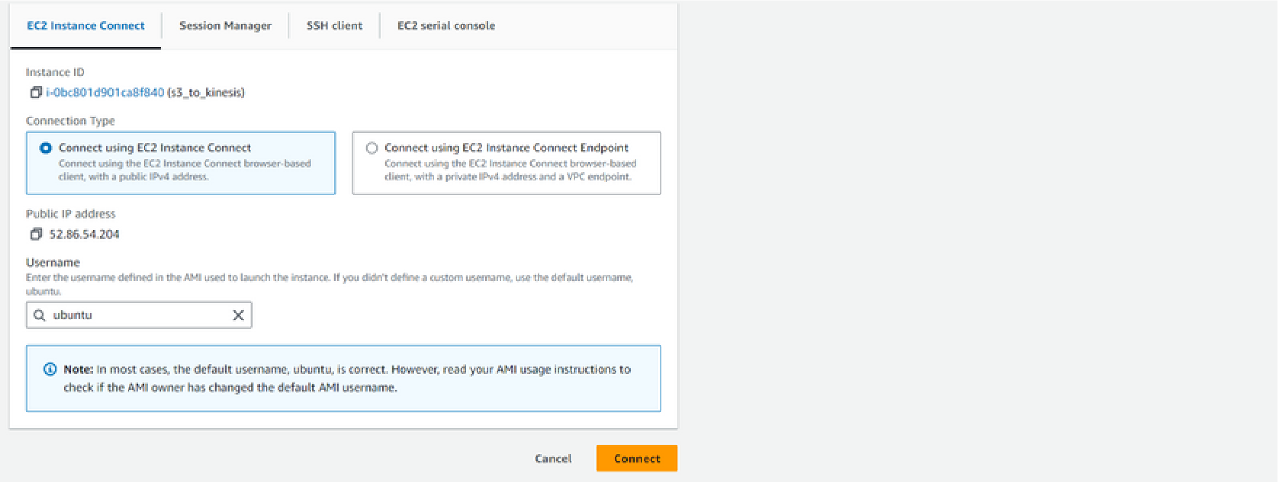
	Name	AWS Region	Access	Creation date
○	athena-query-dataglue	US East (N. Virginia) us-east-1	Bucket and objects not public	February 24, 2024, 10:56:28 (UTC+05:30)
○	log-data-inpoint	US East (N. Virginia) us-east-1	Bucket and objects not public	February 21, 2024, 15:34:32 (UTC+05:30)
○	sample-data-test11	US East (N. Virginia) us-east-1	Bucket and objects not public	February 24, 2024, 00:55:29 (UTC+05:30)

Data Preprocessing with Python Script on EC2 Instance:

Launch an EC2 instance and configure it with the necessary permissions to access the S3 bucket.



☑	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Pul
☑	s3_to_kinesis	i-0bc801d901ca8f840	✓ Running	t2.micro	✓ 2/2 checks passed	View alarms +	us-east-1a	ec2-52-86-54-204.com...	52.



Instance ID
i-0bc801d901ca8f840 (s3_to_kinesis)

Connection Type

● Connect using EC2 Instance Connect
Connect using the EC2 Instance Connect browser-based client, with a public IPv4 address.

○ Connect using EC2 Instance Connect Endpoint
Connect using the EC2 Instance Connect browser-based client, with a private IPv4 address and a VPC endpoint.

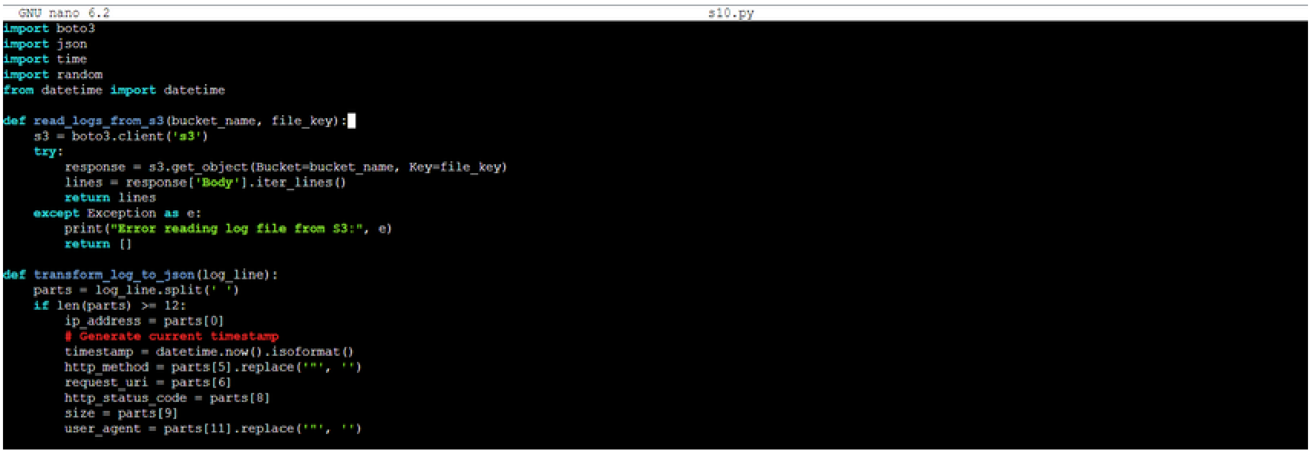
Public IP address
52.86.54.204

Username
Enter the username defined in the AMI used to launch the instance. If you didn't define a custom username, use the default username, ubuntu.

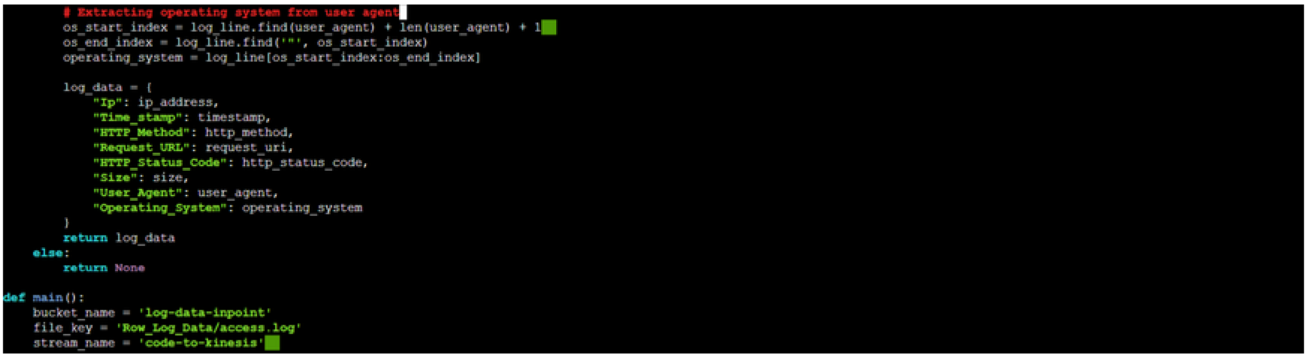
ubuntu

Note: In most cases, the default username, ubuntu, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.

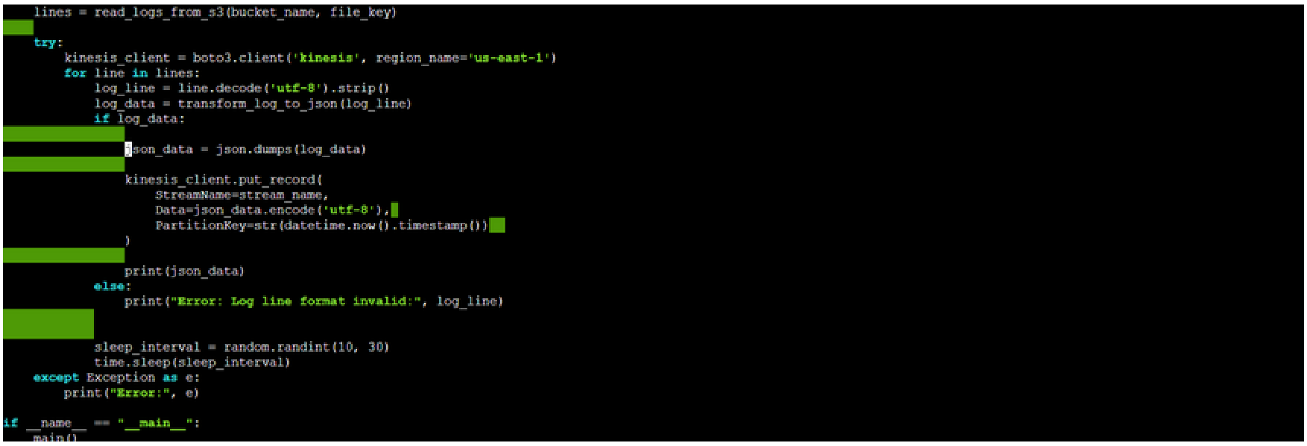
Develop a Python script to preprocess the raw log data, parsing it into a structured format (e.g., JSON).



```
import boto3
import json
import time
import random
from datetime import datetime

def read_logs_from_s3(bucket_name, file_key):
    s3 = boto3.client('s3')
    try:
        response = s3.get_object(Bucket=bucket_name, Key=file_key)
        lines = response['Body'].iter_lines()
        return lines
    except Exception as e:
        print("Error reading log file from S3:", e)
        return []

def transform_log_to_json(log_line):
    parts = log_line.split(' ')
    if len(parts) >= 12:
        ip_address = parts[0]
        # Generate current timestamp
        timestamp = datetime.now().isoformat()
        http_method = parts[5].replace('"', '')
        request_uri = parts[6]
        http_status_code = parts[8]
        size = parts[9]
        user_agent = parts[11].replace('"', '')
```



```
        # Extracting operating system from user agent
        os_start_index = log_line.find(user_agent) + len(user_agent) + 1
        os_end_index = log_line.find('"', os_start_index)
        operating_system = log_line[os_start_index:os_end_index]

        log_data = {
            "Ip": ip_address,
            "Time_stamp": timestamp,
            "HTTP_Method": http_method,
            "Request_URL": request_uri,
            "HTTP_Status_Code": http_status_code,
            "Size": size,
            "User_Agent": user_agent,
            "Operating_System": operating_system
        }
        return log_data
    else:
        return None

def main():
    bucket_name = 'log-data-inpoint'
    file_key = 'Row_Log_Data/access.log'
    stream_name = 'code-to-kinesis'
```



```
    lines = read_logs_from_s3(bucket_name, file_key)

    try:
        kinesis_client = boto3.client('kinesis', region_name='us-east-1')
        for line in lines:
            log_line = line.decode('utf-8').strip()
            log_data = transform_log_to_json(log_line)
            if log_data:

                json_data = json.dumps(log_data)

                kinesis_client.put_record(
                    StreamName=stream_name,
                    Data=json_data.encode('utf-8'),
                    PartitionKey=str(datetime.now().timestamp())
                )

                print(json_data)
            else:
                print("Error: Log line format invalid:", log_line)


            sleep_interval = random.randint(10, 30)
            time.sleep(sleep_interval)
    except Exception as e:
        print("Error:", e)

if __name__ == "__main__":
    main()
```

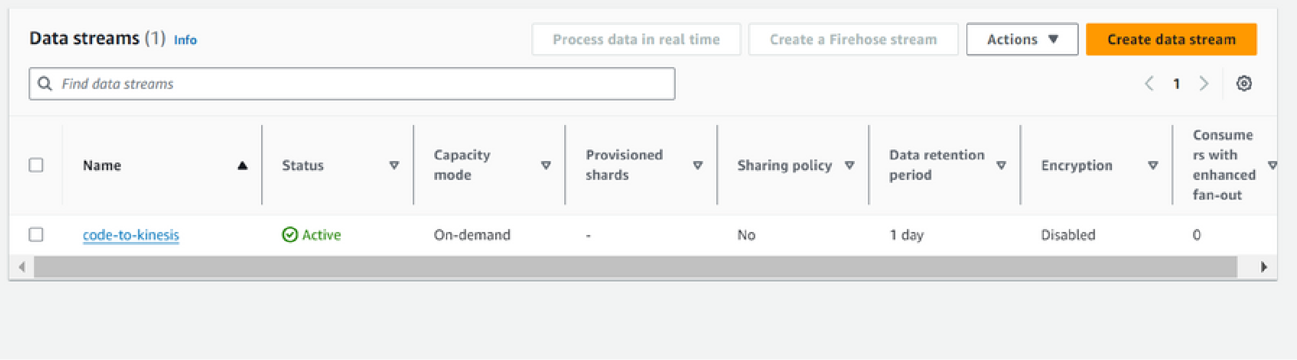
Install any required dependencies (e.g., boto3 for AWS SDK) on the EC2 instance.

Schedule the Python script to run periodically or trigger it manually to preprocess new log data.

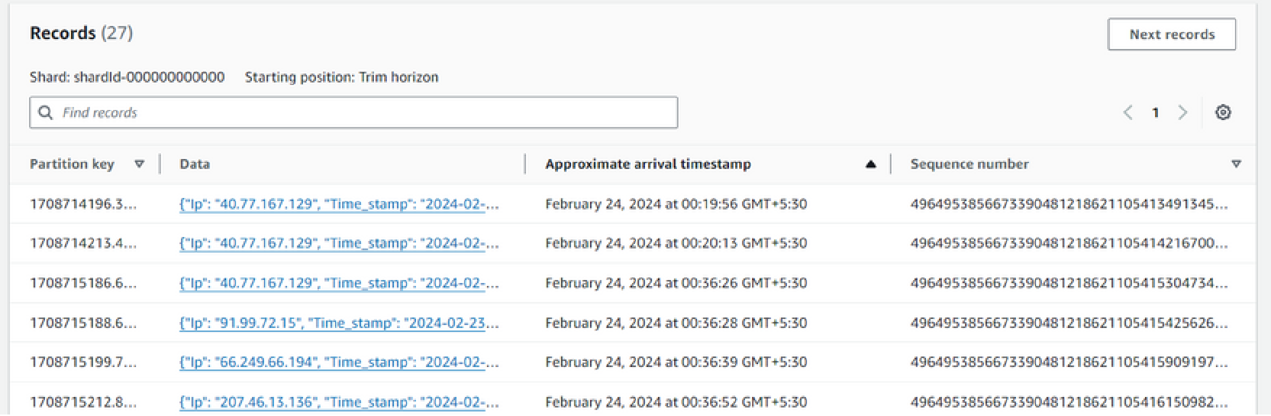
Streaming Data to Amazon Kinesis Data Stream:

Create an Amazon Kinesis Data Stream delivery stream.

Configure the delivery stream to receive data from the EC2 instance.



□	Name ▲	Status	Capacity mode	Provisioned shards	Sharing policy	Data retention period	Encryption	Consumers with enhanced fan-out
□	code-to-kinesis	✓ Active	On-demand	-	No	1 day	Disabled	0

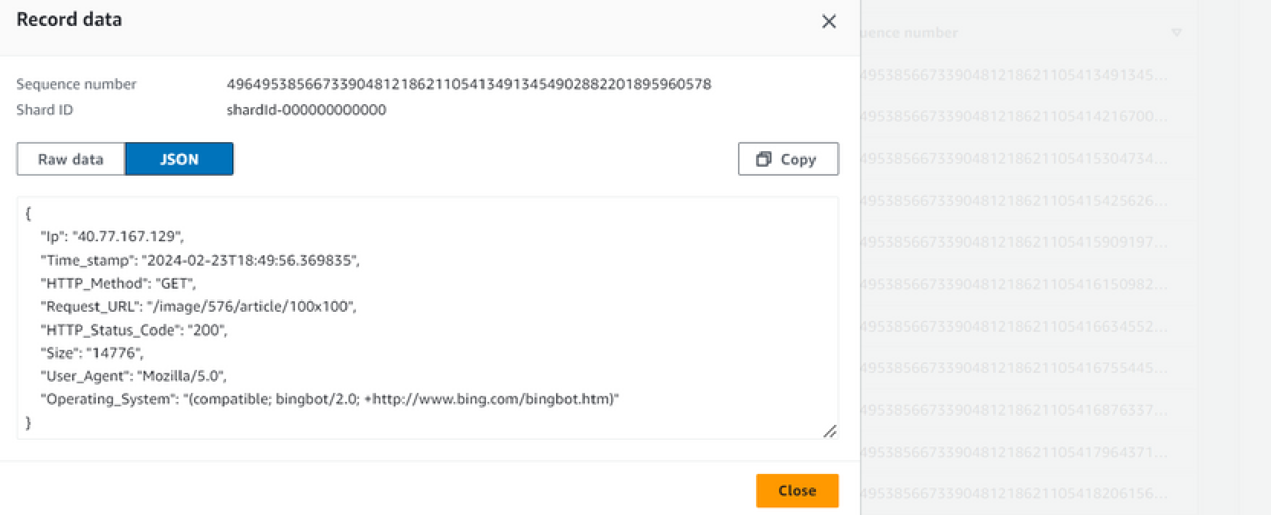


Records (27)

Next records

Shard: shardId-000000000000   Starting position: Trim horizon

Partition key	Data	Approximate arrival timestamp	Sequence number
1708714196.3...	{"Ip": "40.77.167.129", "Time_stamp": "2024-02-...	February 24, 2024 at 00:19:56 GMT+5:30	49649538566733904812186211054134913 45...
1708714213.4...	{"Ip": "40.77.167.129", "Time_stamp": "2024-02-...	February 24, 2024 at 00:20:13 GMT+5:30	4964953856673390481218621105414216700...
1708715186.6...	{"Ip": "40.77.167.129", "Time_stamp": "2024-02-...	February 24, 2024 at 00:36:26 GMT+5:30	4964953856673390481218621105415304734...
1708715188.6...	{"Ip": "91.99.72.15", "Time_stamp": "2024-02-23...	February 24, 2024 at 00:36:28 GMT+5:30	4964953856673390481218621105415425626...
1708715199.7...	{"Ip": "66.249.66.194", "Time_stamp": "2024-02-...	February 24, 2024 at 00:36:39 GMT+5:30	4964953856673390481218621105415909197...
1708715212.8...	{"Ip": "207.46.13.136", "Time_stamp": "2024-02-...	February 24, 2024 at 00:36:52 GMT+5:30	4964953856673390481218621105416150982...



Record data

Sequence number 49649538566733904812186211054134913454902882201895960578
Shard ID shardId-000000000000

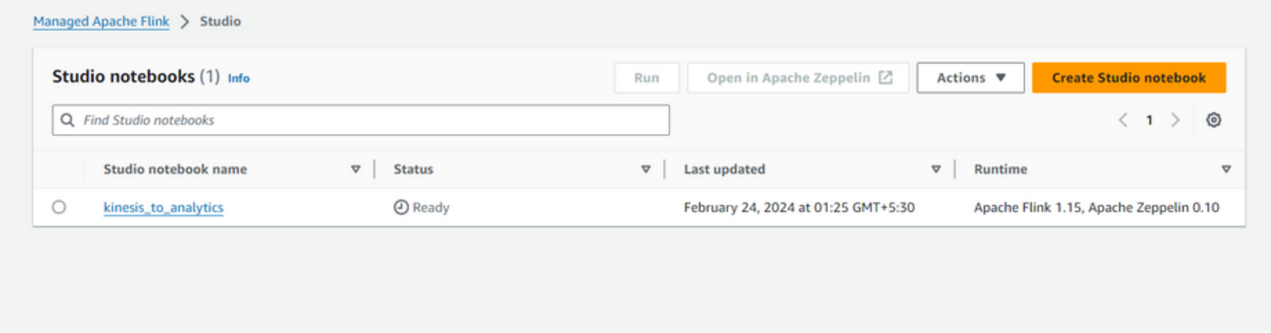
```
{
   "Ip": "40.77.167.129",
   "Time_stamp": "2024-02-23T18:49:56.369835",
   "HTTP_Method": "GET",
   "Request_URL": "/image/576/article/100x100",
   "HTTP_Status_Code": "200",
   "Size": "14776",
   "User_Agent": "Mozilla/5.0",
   "Operating_System": "(compatible; bingbot/2.0; +http://www.bing.com/bingbot.htm)"
}
```

uence number
49538566733904812186211054134913 45...
4953856673390481218621105414216700...
4953856673390481218621105415304734...
4953856673390481218621105415425626...
4953856673390481218621105415909197...
4953856673390481218621105416150982...
4953856673390481218621105416634552...
4953856673390481218621105416755445...
4953856673390481218621105416876337...
4953856673390481218621105417964371...
4953856673390481218621105418206156...

Real-Time Log Analysis with Amazon Kinesis Data Analytics:

Create an Amazon Kinesis Data Analytics application.

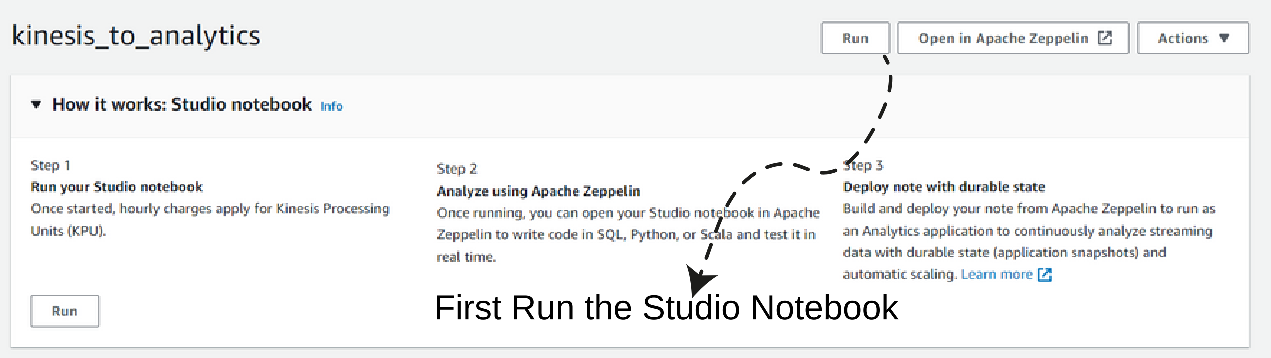
Define the input source as the Kinesis Data Firehose delivery stream.(OPT)



Managed Apache Flink > Studio

Studio notebooks (1) Info

	Studio notebook name	Status	Last updated	Runtime
○	kinesis_to_analytics	Ready	February 24, 2024 at 01:25 GMT+5:30	Apache Flink 1.15, Apache Zeppelin 0.10

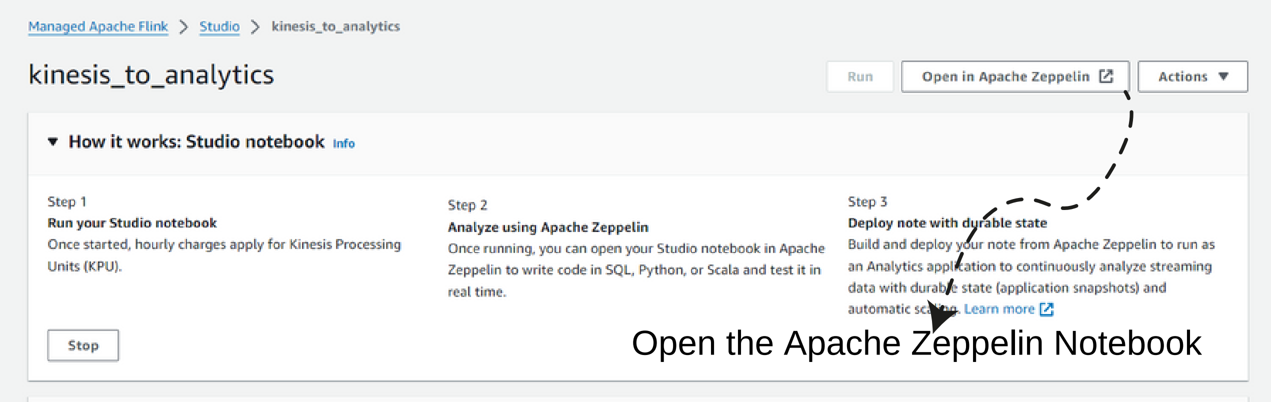


Step 1
Run your Studio notebook
Once started, hourly charges apply for Kinesis Processing Units (KPU).

Step 2
Analyze using Apache Zeppelin
Once running, you can open your Studio notebook in Apache Zeppelin to write code in SQL, Python, or Scala and test it in real time.

Step 3
Deploy note with durable state
Build and deploy your note from Apache Zeppelin to run as an Analytics application to continuously analyze streaming data with durable state (application snapshots) and automatic scaling. Learn more

First Run the Studio Notebook



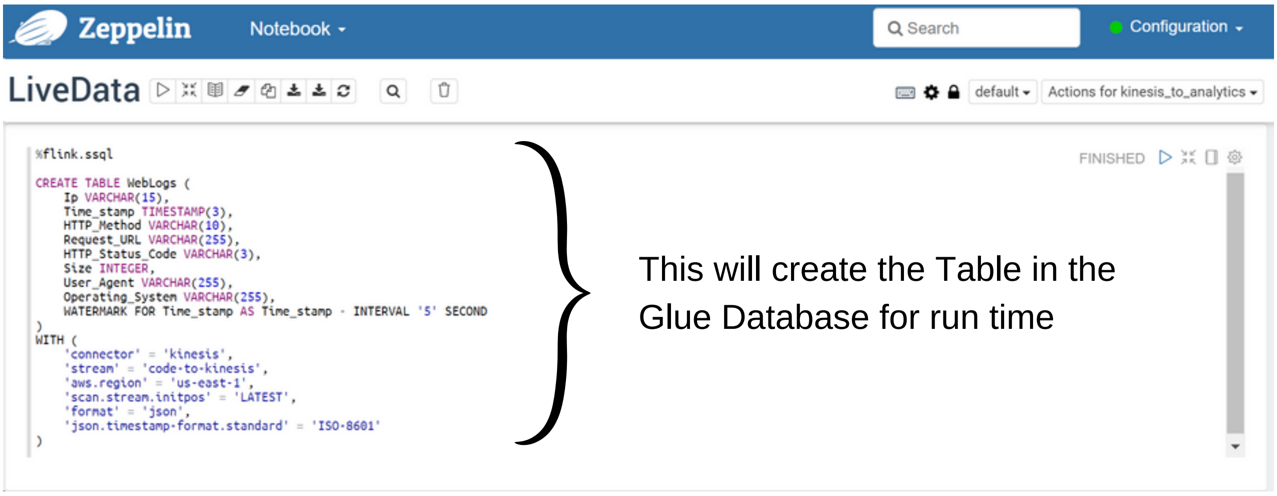
Managed Apache Flink > Studio > kinesis_to_analytics
kinesis_to_analytics
Run
Open in Apache Zeppelin
Actions ▼
▼ How it works: Studio notebook Info

Step 1
Run your Studio notebook
Once started, hourly charges apply for Kinesis Processing Units (KPU).

Step 2
Analyze using Apache Zeppelin
Once running, you can open your Studio notebook in Apache Zeppelin to write code in SQL, Python, or Scala and test it in real time.

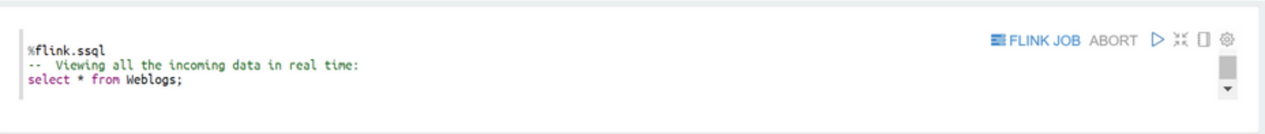
Step 3
Deploy note with durable state
Build and deploy your note from Apache Zeppelin to run as an Analytics application to continuously analyze streaming data with durable state (application snapshots) and automatic scaling. Learn more

Open the Apache Zeppelin Notebook

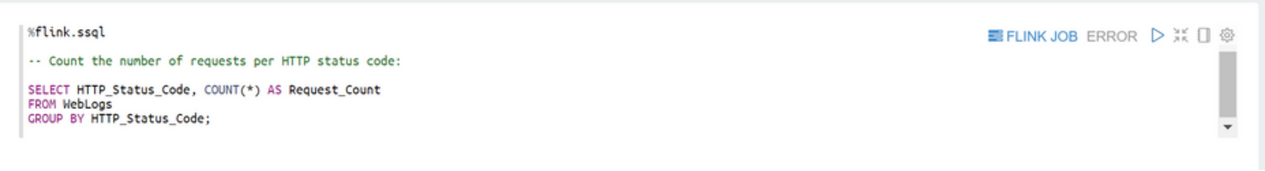


```
%flink.ssql

CREATE TABLE WebLogs (
    Ip VARCHAR(15),
    Time_stamp TIMESTAMP(3),
    HTTP_Method VARCHAR(10),
    Request_URL VARCHAR(255),
    HTTP_Status_Code VARCHAR(3),
    Size INTEGER,
    User_Agent VARCHAR(255),
    Operating_System VARCHAR(255),
    WATERMARK FOR Time_stamp AS Time_stamp - INTERVAL '5' SECOND
)
WITH (
    'connector' = 'kinesis',
    'stream' = 'code-to-kinesis',
    'aws.region' = 'us-east-1',
    'scan.stream.initpos' = 'LATEST',
    'format' = 'json',
    'json.timestamp-format.standard' = 'ISO-8601'
)
```

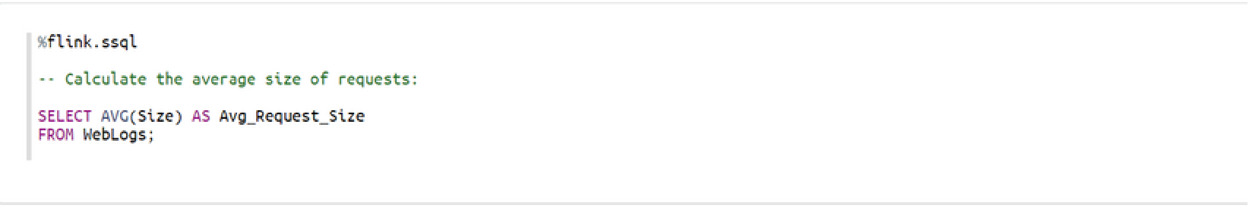
This will create the Table in the Glue Database for run time



```
%flink.ssql
-- Viewing all the incoming data in real time:
select * from Weblogs;
```



```
%flink.ssql

-- Count the number of requests per HTTP status code:

SELECT HTTP_Status_Code, COUNT(*) AS Request_Count
FROM WebLogs
GROUP BY HTTP_Status_Code;
```

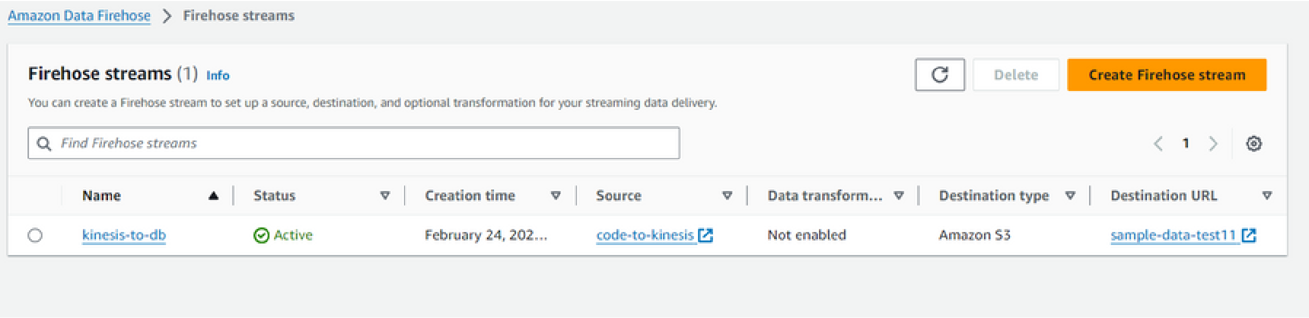


```
%flink.ssql

-- Calculate the average size of requests:

SELECT AVG(Size) AS Avg_Request_Size
FROM WebLogs;
```

Configuring Amazon Kinesis Data Firehose for Data Delivery to Amazon S3:

Configure the Amazon Kinesis Data Firehose delivery stream to deliver streaming data to an Amazon S3 bucket.

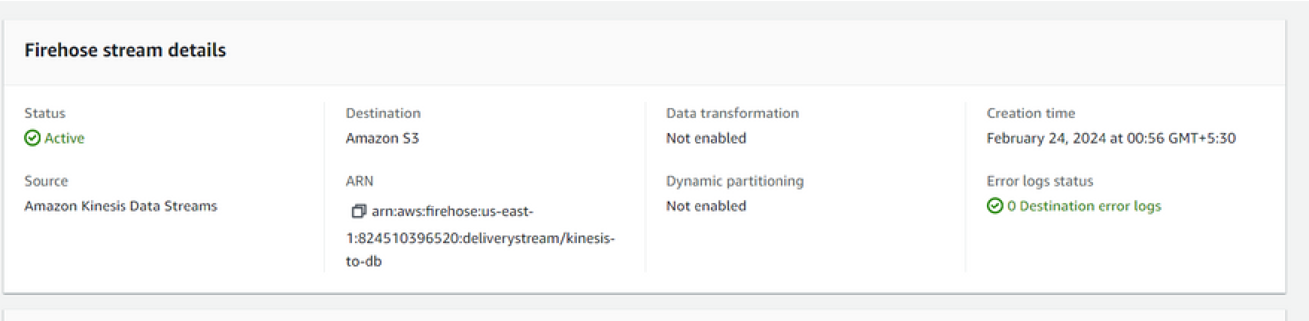
Define the S3 destination bucket and any required data transformation settings.



Firehose streams (1) Info

You can create a Firehose stream to set up a source, destination, and optional transformation for your streaming data delivery.

	Name	Status	Creation time	Source	Data transform...	Destination type	Destination URL
○	kinesis-to-db	✓ Active	February 24, 202...	code-to-kinesis	Not enabled	Amazon S3	sample-data-test11



Firehose stream details

Status
✓ Active

Source
Amazon Kinesis Data Streams

Destination
Amazon S3

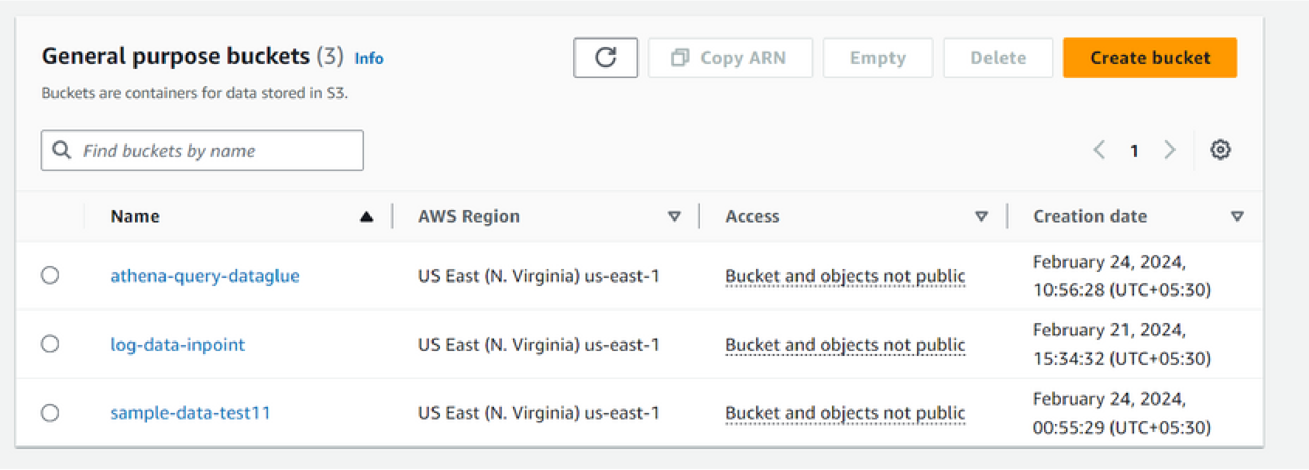
ARN
arn:aws:firehose:us-east-1:824510396520:deliverystream/kinesis-to-db

Data transformation
Not enabled

Dynamic partitioning
Not enabled

Creation time
February 24, 2024 at 00:56 GMT+5:30

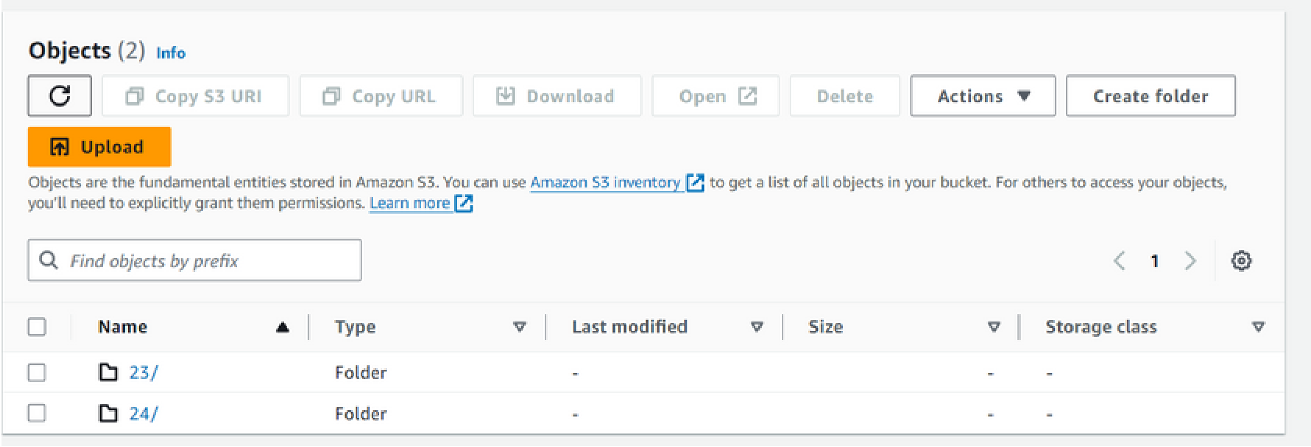
Error logs status
✓ 0 Destination error logs



General purpose buckets (3) Info

Buckets are containers for data stored in S3.

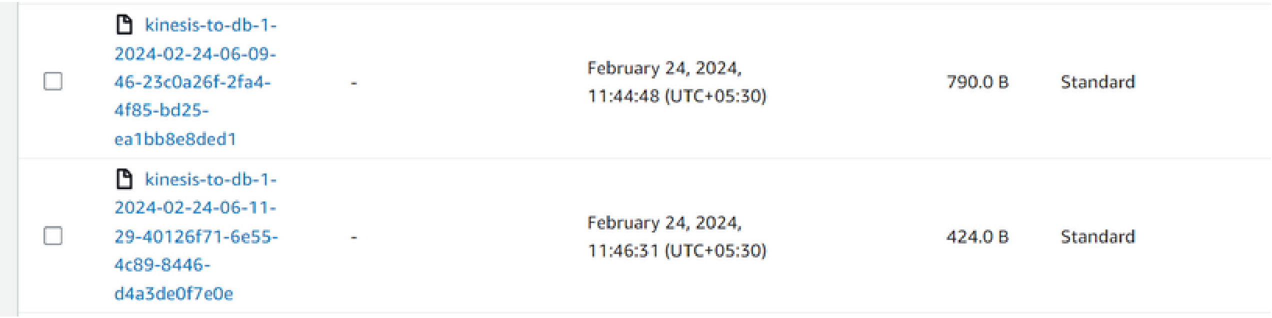
	Name	AWS Region	Access	Creation date
○	athena-query-dataglue	US East (N. Virginia) us-east-1	Bucket and objects not public	February 24, 2024, 10:56:28 (UTC+05:30)
○	log-data-inpoint	US East (N. Virginia) us-east-1	Bucket and objects not public	February 21, 2024, 15:34:32 (UTC+05:30)
○	sample-data-test11	US East (N. Virginia) us-east-1	Bucket and objects not public	February 24, 2024, 00:55:29 (UTC+05:30)



Objects (2) Info

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 objects, you'll need to explicitly grant them permissions. Learn more

□	Name	Type	Last modified	Size	Storage class
□	23/	Folder	-	-	-
□	24/	Folder	-	-	-

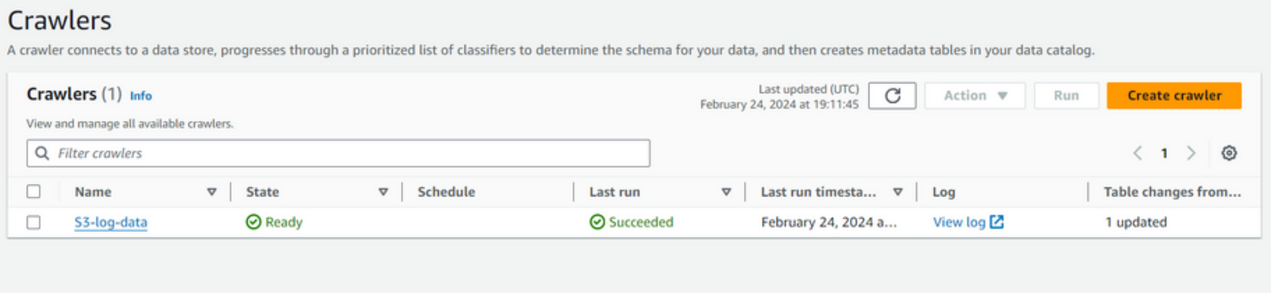


□	kinesis-to-db-1-2024-02-24-06-09-46-23c0a26f-2fa4-4f85-bd25-ea1bb8e8ded1	-	February 24, 2024, 11:44:48 (UTC+05:30)	790.0 B	Standard
□	kinesis-to-db-1-2024-02-24-06-11-29-40126f71-6e55-4c89-8446-d4a3de0f7e0e	-	February 24, 2024, 11:46:31 (UTC+05:30)	424.0 B	Standard

Data Cataloging and ETL with AWS Glue:

Create an AWS Glue crawler to discover the schema of the streaming log data stored in Amazon S3.

Run the Glue crawler to catalog the data and create corresponding metadata tables in the Glue Data Catalog.



Crawlers

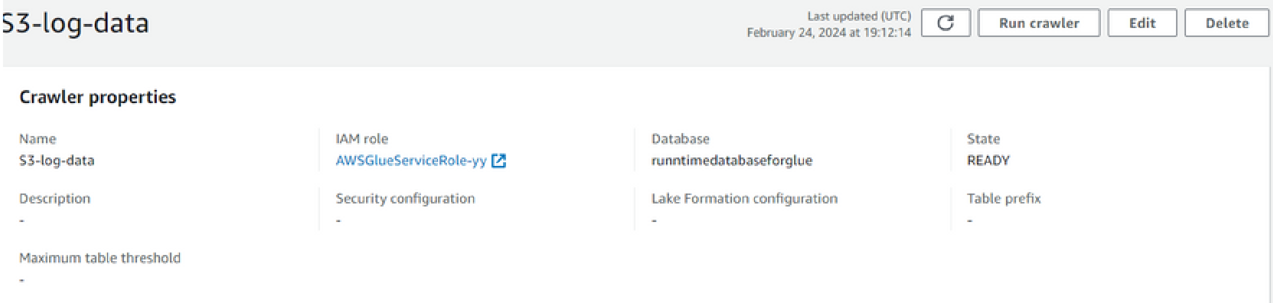
A crawler connects to a data store, progresses through a prioritized list of classifiers to determine the schema for your data, and then creates metadata tables in your data catalog.

Crawlers (1) Info

View and manage all available crawlers.

Last updated (UTC)
February 24, 2024 at 19:11:45

□	Name	State	Schedule	Last run	Last run timesta...	Log	Table changes from...
□	S3-log-data	✓ Ready		✓ Succeeded	February 24, 2024 a...	View log	1 updated



S3-log-data

Last updated (UTC)
February 24, 2024 at 19:12:14

Crawler properties

Name
S3-log-data

IAM role
AWSGlueServiceRole-yy

Database
runntimedatabaseforglue

State
READY

Description
-

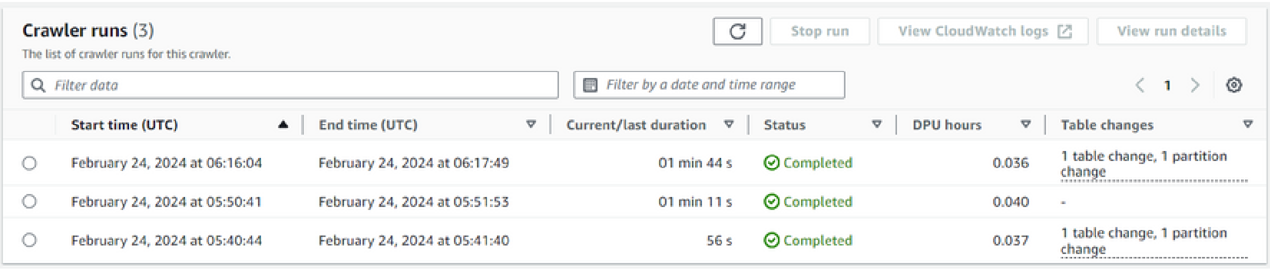
Security configuration
-

Lake Formation configuration
-

Table prefix
-

Maximum table threshold
-

Develop Glue ETL jobs to perform any required transformations on the data, preparing it for querying and analysis



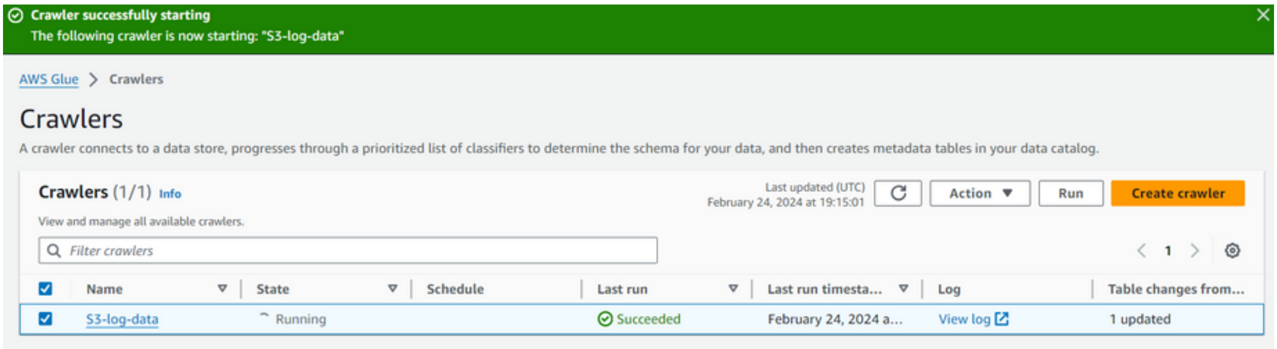
Crawler runs (3)

The list of crawler runs for this crawler.

	Start time (UTC)	End time (UTC)	Current/last duration	Status	DPU hours	Table changes
○	February 24, 2024 at 06:16:04	February 24, 2024 at 06:17:49	01 min 44 s	✓ Completed	0.036	1 table change, 1 partition change
○	February 24, 2024 at 05:50:41	February 24, 2024 at 05:51:53	01 min 11 s	✓ Completed	0.040	-
○	February 24, 2024 at 05:40:44	February 24, 2024 at 05:41:40	56 s	✓ Completed	0.037	1 table change, 1 partition change

Running AWS Glue Crawler for Cataloging:

Schedule the Glue crawler to run periodically to detect schema changes and update the Glue Data Catalog accordingly.



Crawler successfully starting

The following crawler is now starting: "S3-log-data"

AWS Glue > Crawlers

Crawlers

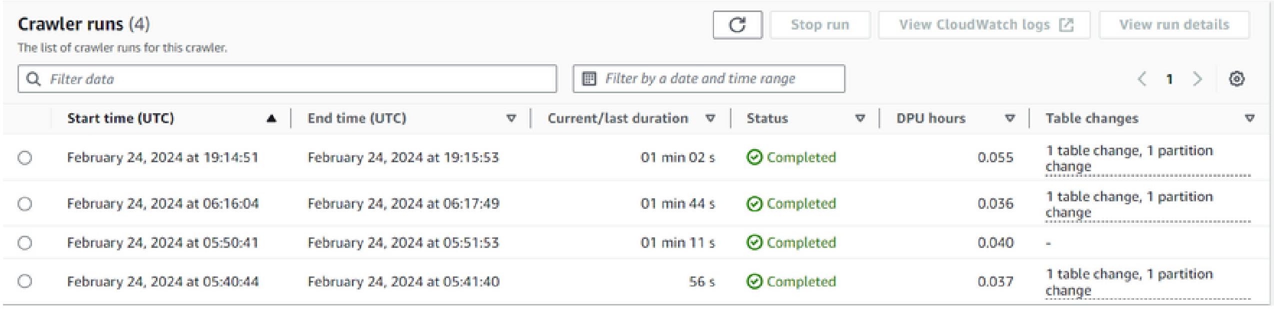
A crawler connects to a data store, progresses through a prioritized list of classifiers to determine the schema for your data, and then creates metadata tables in your data catalog.

Crawlers (1/1) Info

View and manage all available crawlers.

Last updated (UTC)
February 24, 2024 at 19:15:01

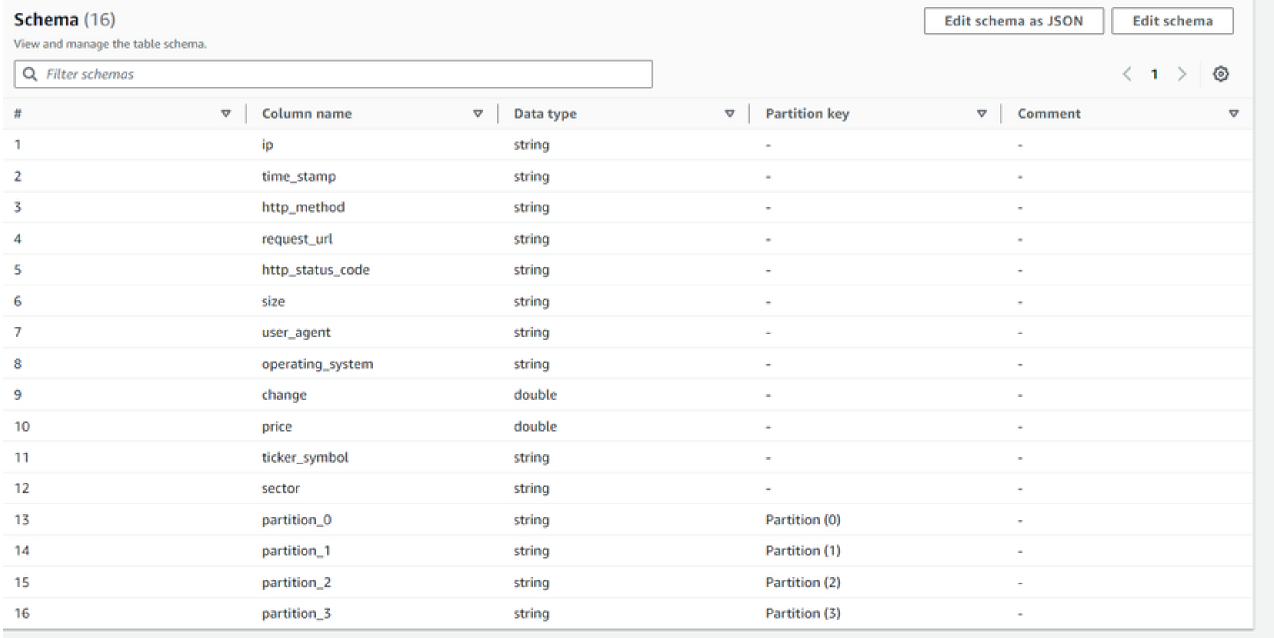
☑	Name	State	Schedule	Last run	Last run timesta...	Log	Table changes from...
☑	S3-log-data	Running		✓ Succeeded	February 24, 2024 a...	View log	1 updated



Crawler runs (4)

The list of crawler runs for this crawler.

	Start time (UTC)	End time (UTC)	Current/last duration	Status	DPU hours	Table changes
○	February 24, 2024 at 19:14:51	February 24, 2024 at 19:15:53	01 min 02 s	✓ Completed	0.055	1 table change, 1 partition change
○	February 24, 2024 at 06:16:04	February 24, 2024 at 06:17:49	01 min 44 s	✓ Completed	0.036	1 table change, 1 partition change
○	February 24, 2024 at 05:50:41	February 24, 2024 at 05:51:53	01 min 11 s	✓ Completed	0.040	-
○	February 24, 2024 at 05:40:44	February 24, 2024 at 05:41:40	56 s	✓ Completed	0.037	1 table change, 1 partition change



Schema (16)

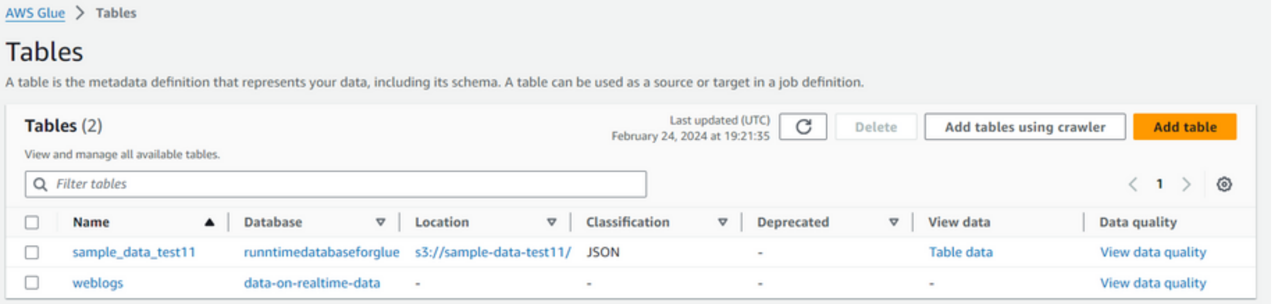
View and manage the table schema.

#	Column name	Data type	Partition key	Comment
1	ip	string	-	-
2	time_stamp	string	-	-
3	http_method	string	-	-
4	request_url	string	-	-
5	http_status_code	string	-	-
6	size	string	-	-
7	user_agent	string	-	-
8	operating_system	string	-	-
9	change	double	-	-
10	price	double	-	-
11	ticker_symbol	string	-	-
12	sector	string	-	-
13	partition_0	string	Partition (0)	-
14	partition_1	string	Partition (1)	-
15	partition_2	string	Partition (2)	-
16	partition_3	string	Partition (3)	-

Querying Historical Data with Amazon Athena:

Use Amazon Athena to query historical log data stored in Amazon S3.

Define the external tables in Athena using the Glue Data Catalog metadata.



AWS Glue > Tables

Tables

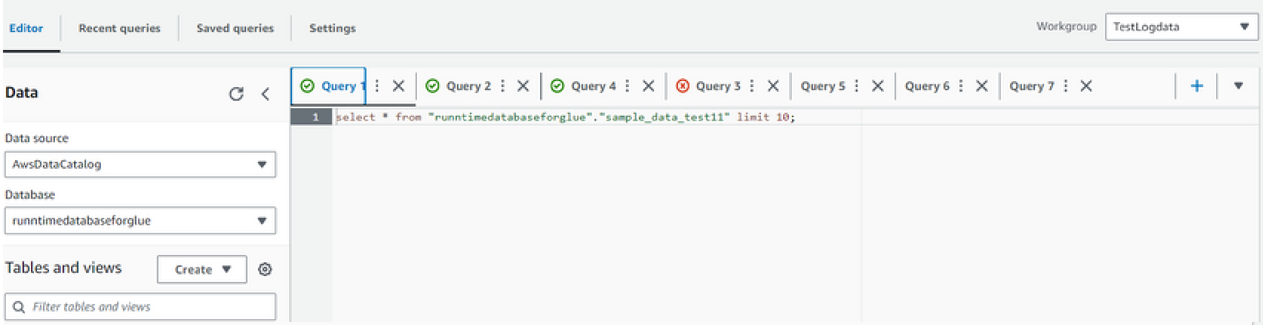
A table is the metadata definition that represents your data, including its schema. A table can be used as a source or target in a job definition.

Tables (2)

View and manage all available tables.

Last updated (UTC) February 24, 2024 at 19:21:35

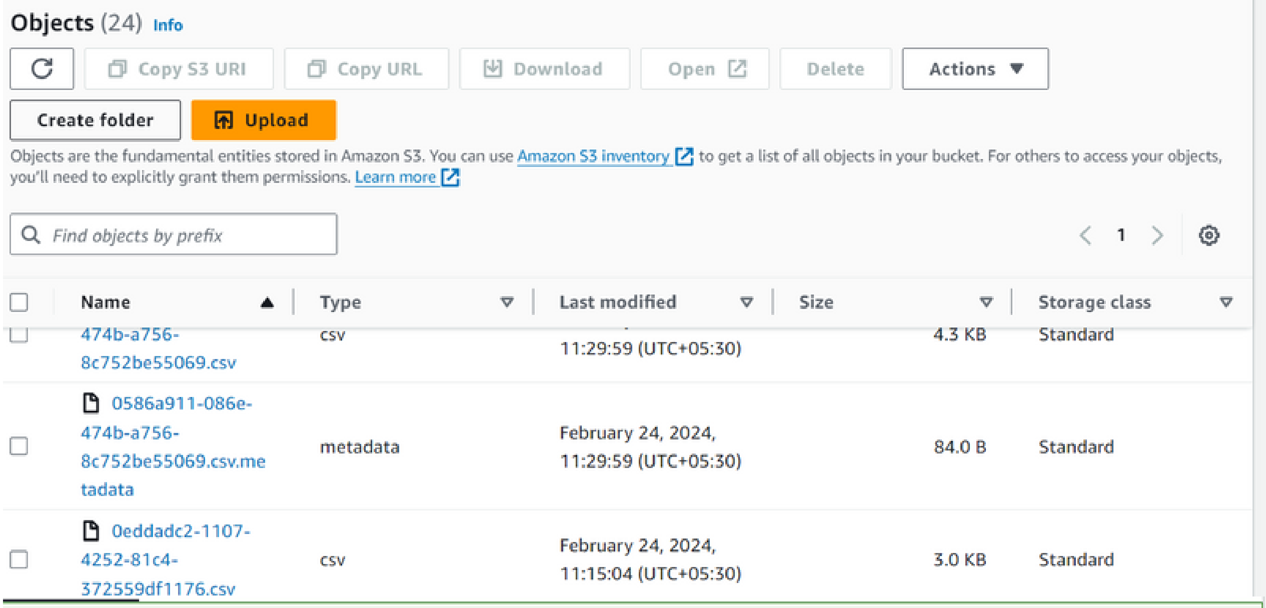
Name	Database	Location	Classification	Deprecated	View data	Data quality
sample_data_test11	runntimedatabaseforglue	s3://sample-data-test11/	JSON	-	Table data	View data quality
weblogs	data-on-realtime-data	-	-	-	-	View data quality



Data source: AwsDataCatalog

Database: runntimedatabaseforglue

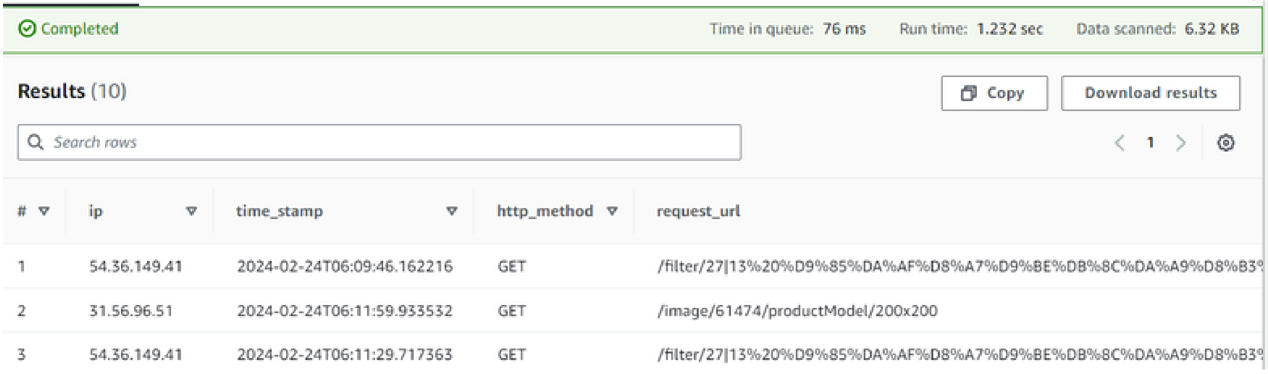
```
select * from "runntimedatabaseforglue"."sample_data_test11" limit 10;
```



Objects (24) Info

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 objects, you'll need to explicitly grant them permissions. Learn more

☐	Name	Type	Last modified	Size	Storage class
☐	474b-a756-8c752be55069.csv	csv	11:29:59 (UTC+05:30)	4.3 KB	Standard
☐	0586a911-086e-474b-a756-8c752be55069.csv.metadata	metadata	February 24, 2024, 11:29:59 (UTC+05:30)	84.0 B	Standard
☐	0eddadc2-1107-4252-81c4-372559df1176.csv	csv	February 24, 2024, 11:15:04 (UTC+05:30)	3.0 KB	Standard



Completed Time in queue: 76 ms Run time: 1.232 sec Data scanned: 6.32 KB

Results (10)

#	ip	time_stamp	http_method	request_url
1	54.36.149.41	2024-02-24T06:09:46.162216	GET	/filter/27|13%20%D9%85%DA%AF%D8%A7%D9%BE%DB%8C%DA%A9%D8%B3%
2	31.56.96.51	2024-02-24T06:11:59.933532	GET	/image/61474/productModel/200x200
3	54.36.149.41	2024-02-24T06:11:29.717363	GET	/filter/27|13%20%D9%85%DA%AF%D8%A7%D9%BE%DB%8C%DA%A9%D8%B3%

By following these implementation steps, the "Log Aggregation and Analysis" system can be effectively set up and configured to collect, analyze, and store log data in real-time and over historical periods, enabling organizations to gain valuable insights into their systems' behavior and performance.

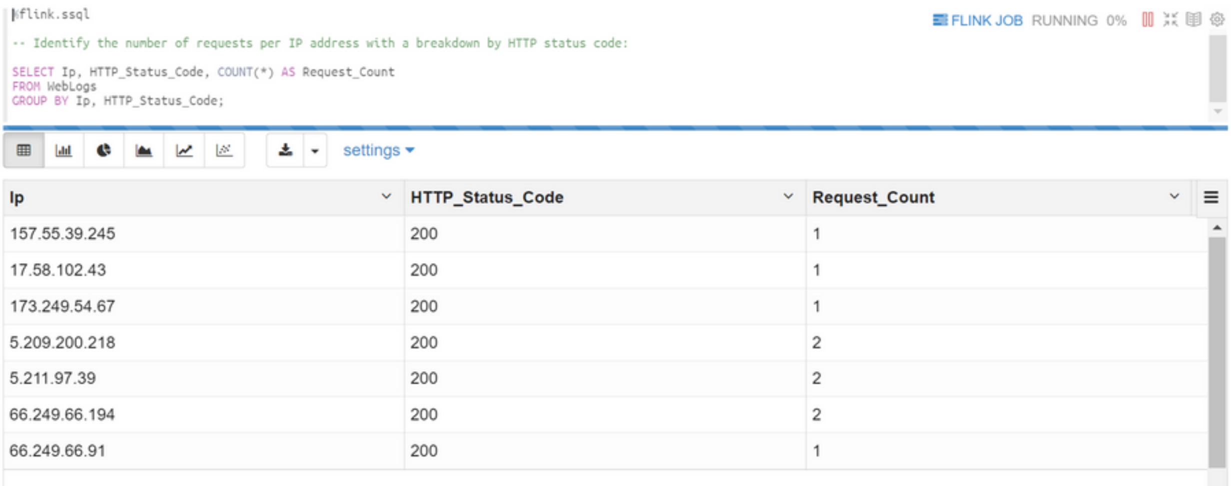
RESULTS:-

The implementation of the "Log Aggregation and Analysis" system has yielded significant results in terms of real-time insights and historical data analysis. Below are the key outcomes achieved:

Real-Time Log Analysis in Zeppelin Notebook: The integration of Amazon Kinesis Data Analytics with Apache Zeppelin notebooks has enabled real-time analysis of streaming log data.

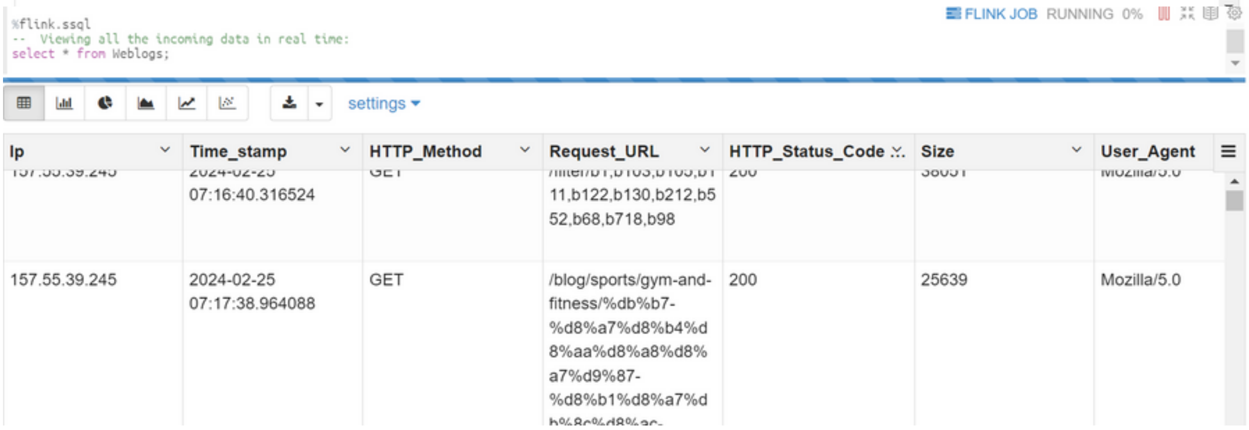
Queries written in SQL within Zeppelin notebooks provide immediate insights into system behavior, performance metrics, and anomalies.

Visualization capabilities in Zeppelin notebooks allow for the creation of dynamic dashboards and charts to monitor system health and performance in real-time.



```
%flink.ssql
-- Identify the number of requests per IP address with a breakdown by HTTP status code:

SELECT Ip, HTTP_Status_Code, COUNT(*) AS Request_Count
FROM WebLogs
GROUP BY Ip, HTTP_Status_Code;
```

Ip	HTTP_Status_Code	Request_Count
157.55.39.245	200	1
17.58.102.43	200	1
173.249.54.67	200	1
5.209.200.218	200	2
5.211.97.39	200	2
66.249.66.194	200	2
66.249.66.91	200	1



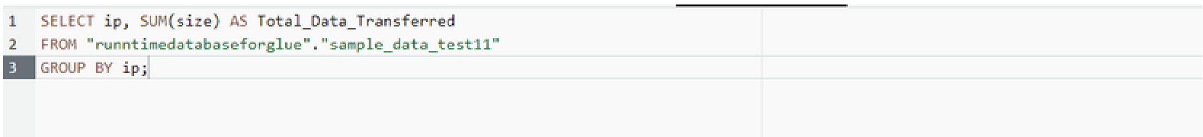
```
%flink.ssql
--  Viewing all the incoming data in real time:
select * from Weblogs;
```

Ip	Time_stamp	HTTP_Method	Request_URL	HTTP_Status_Code	Size	User_Agent
157.55.39.245	2024-02-25 07:16:40.316524	GET	/filter/b1,b103,b105,b111,b122,b130,b212,b552,b68,b718,b98	200	38051	Mozilla/5.0
157.55.39.245	2024-02-25 07:17:38.964088	GET	/blog/sports/gym-and-fitness/%db%b7-%d8%a7%d8%b4%d8%aa%d8%a8%d8%a7%d9%87-%d8%b1%d8%a7%db%8c%d8%ac-	200	25639	Mozilla/5.0

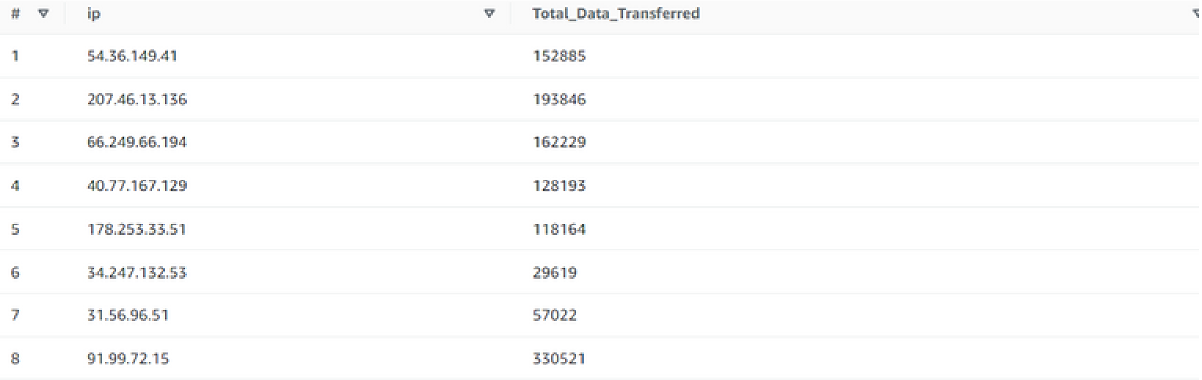
Historical Log Data Querying with Amazon Athena:

Amazon Athena has facilitated ad-hoc querying of historical log data stored in Amazon S3.

By defining external tables in Athena using the Glue Data Catalog metadata, querying historical log data becomes seamless and efficient.

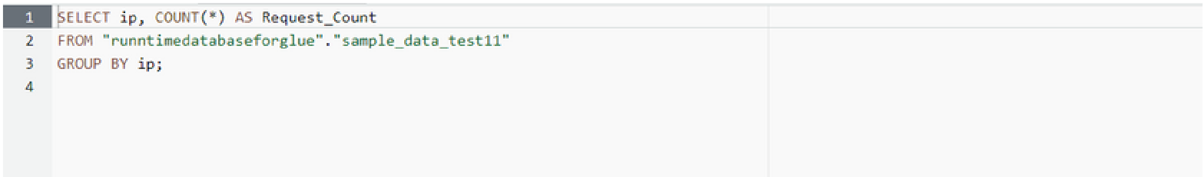


```
SELECT ip, SUM(size) AS Total_Data_Transferred
FROM "runntimedatabaseforglue"."sample_data_test11"
GROUP BY ip;
```

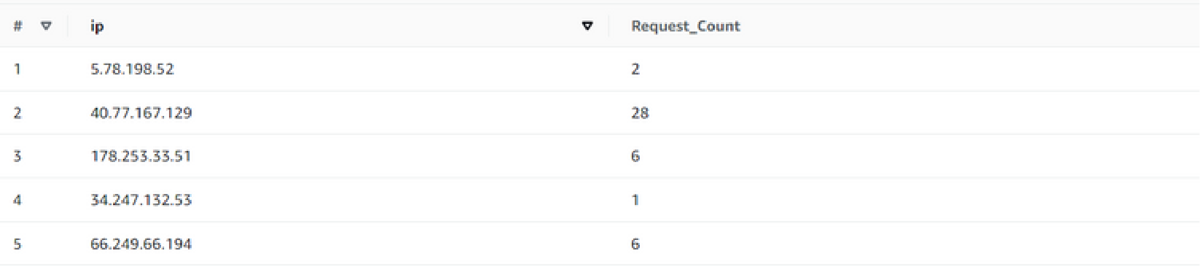


#	ip	Total_Data_Transferred
1	54.36.149.41	152885
2	207.46.13.136	193846
3	66.249.66.194	162229
4	40.77.167.129	128193
5	178.253.33.51	118164
6	34.247.132.53	29619
7	31.56.96.51	57022
8	91.99.72.15	330521

SQL queries executed in Amazon Athena provide insights into past system behavior, trends, and patterns, enabling retrospective analysis and decision-making.



```
SELECT ip, COUNT(*) AS Request_Count
FROM "runntimedatabaseforglue"."sample_data_test11"
GROUP BY ip;
```



#	ip	Request_Count
1	5.78.198.52	2
2	40.77.167.129	28
3	178.253.33.51	6
4	34.247.132.53	1
5	66.249.66.194	6

CONCLUSION:-

The development and implementation of the "Log Aggregation and Analysis" system have demonstrated the effectiveness of leveraging AWS cloud services to manage, analyze, and derive insights from log data. By integrating various AWS services such as Amazon Kinesis Data Firehose, Amazon Kinesis Data Analytics, Amazon S3, AWS Glue, and Amazon Athena, the system provides a scalable, real-time, and comprehensive solution for log management and analysis.

Through this project, several key conclusions can be drawn:

Operational Efficiency: The real-time log analysis capabilities provided by Amazon Kinesis Data Analytics enable organizations to monitor system health, detect anomalies, and respond promptly to operational issues, thereby improving overall operational efficiency.

Data-driven Decision Making: The insights derived from both real-time and historical log data analysis empower organizations to make informed, data-driven decisions to optimize system performance, enhance user experience, and drive business outcomes.

Scalability and Flexibility: The architecture of the system built on AWS cloud services ensures scalability and flexibility to handle varying volumes of log data, adapt to changing business requirements, and accommodate future growth.

Cost-effectiveness: Leveraging managed services offered by AWS eliminates the need for upfront investments in infrastructure and reduces operational costs. Pay-as-you-go pricing models ensure cost-effectiveness by aligning expenses with actual resource utilization.

Ease of Management: The managed nature of AWS services simplifies infrastructure management, reduces administrative overhead, and allows organizations to focus on core business activities rather than managing infrastructure.

In conclusion, the "**Log Aggregation and Analysis**" system serves as a foundational framework for organizations to effectively manage and derive insights from log data, enabling them to optimize operations, improve system performance, and drive business success. By harnessing the power of AWS cloud services, organizations can unlock the full potential of their log data and gain a competitive advantage in today's data-driven landscape.

THE END