Amazon Web Services
Data Engineering Immersion Day

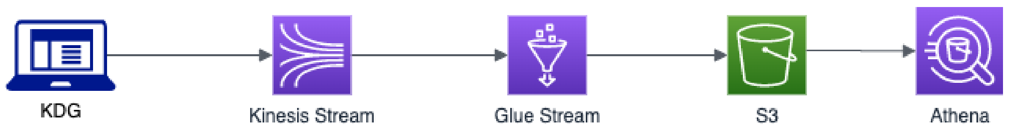
Lab 1. Hydrating the Data Lake with Glue Streaming ETL

March 2021

Table of Contents

Introduction

This lab will guide you to understand AWS Glue Streaming ETL feature. You will start with hydrating your Data Lake from Amazon Kinesis Data Generator (KDG). The final outcome is to query the Data Lake in near real-time.



In this lab you will complete the following tasks:

1. Setup a Streaming Data Generator for Kinesis
2. Create Glue Streaming job
3. Query the data stream in Athena

   If you'd like to run the workshop on your own after the AWS hosted event, please follow the lab instruction here: https://github.com/aws-samples/data-engineering-for-aws-immersion-day

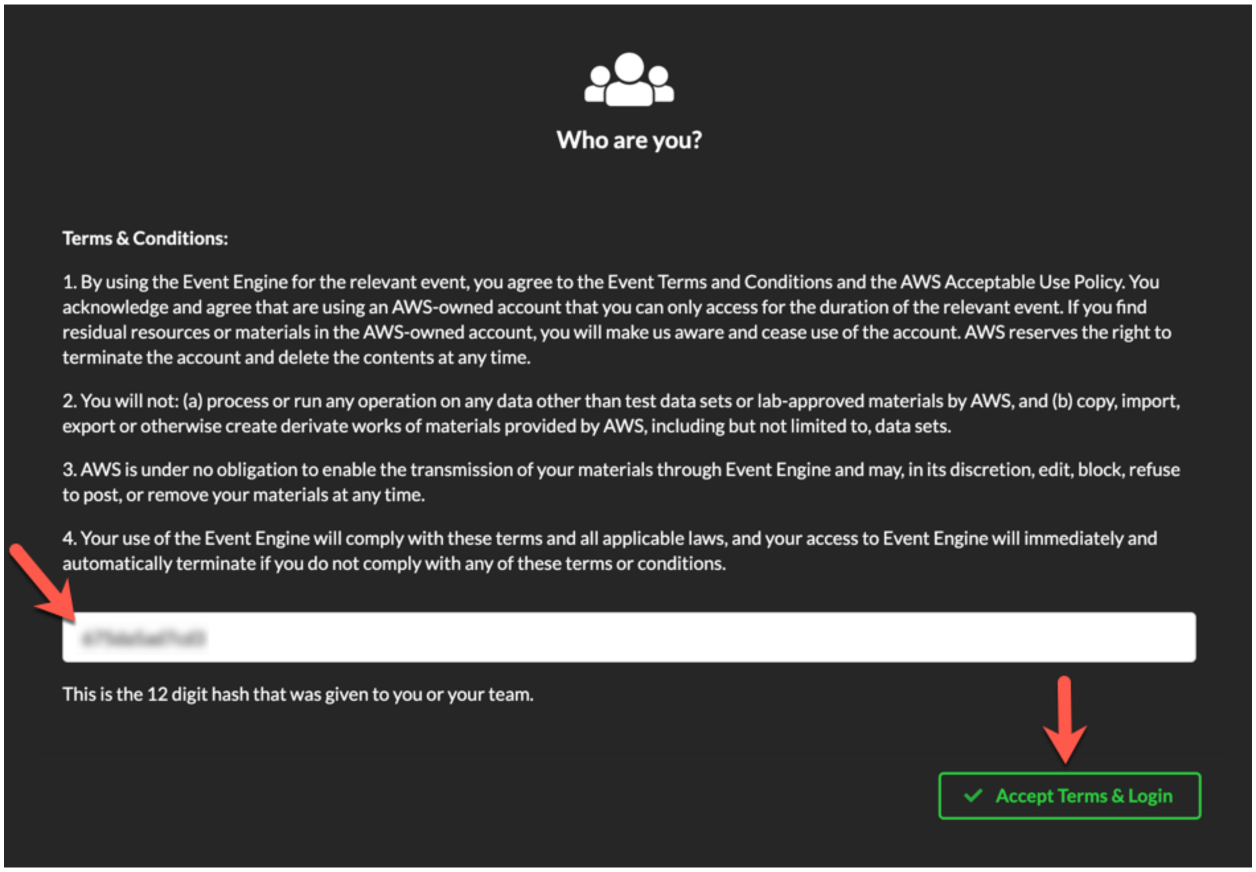
Go To Lab Environment

Please skip this section if you are running the lab on your own AWS account.

Today, you are attending a formal event and you will have been sent your access details beforehand. If in the future you might want to perform these labs in your own AWS environment by yourself, you can follow instructions on GitHub - https://github.com/aws-samples/data-engineering-for-aws-immersion-day.

A 12-character access code (or 'hash') is the access code that grants you permission to use a dedicated AWS account for the purposes of this workshop.

1. Go to https://dashboard.eventengine.run/, enter the access code and click Proceed:



Terms & Conditions:

1. By using the Event Engine for the relevant event, you agree to the Event Terms and Conditions and the AWS Acceptable Use Policy. You acknowledge and agree that are using an AWS-owned account that you can only access for the duration of the relevant event. If you find residual resources or materials in the AWS-owned account, you will make us aware and cease use of the account. AWS reserves the right to terminate the account and delete the contents at any time.

2. You will not: (a) process or run any operation on any data other than test data sets or lab-approved materials by AWS, and (b) copy, import, export or otherwise create derivate works of materials provided by AWS, including but not limited to, data sets.

3. AWS is under no obligation to enable the transmission of your materials through Event Engine and may, in its discretion, edit, block, refuse to post, or remove your materials at any time.

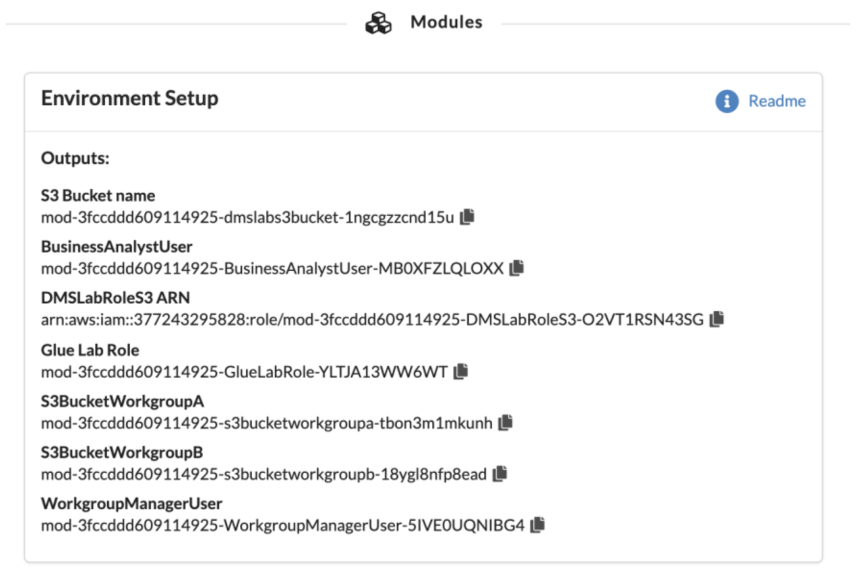
4. Your use of the Event Engine will comply with these terms and all applicable laws, and your access to Event Engine will immediately and automatically terminate if you do not comply with any of these terms or conditions.

This is the 12 digit hash that was given to you or your team.

2. On the Team Dashboard web page, you will see a set of parameters that you will need during the labs. Best to save them to a text file locally, alternatively you can always go to this page to review them. Replace the parameters with the corresponding values from here where indicated in subsequent labs:

Because you're at a formal event, some AWS resources have been pre-deployed for your convenience, for example:

- S3 Bucket, IAM roles etc



Modules

Environment Setup Readme

Outputs:

S3 Bucket name
mod-3fccddd609114925-dmslabs3bucket-1ngcgzzcnd15u

BusinessAnalystUser
mod-3fccddd609114925-BusinessAnalystUser-MB0XFZLQLOXX

DMSLabRoleS3 ARN
arn:aws:iam::377243295828:role/mod-3fccddd609114925-DMSLabRoleS3-O2VT1RSN43SG

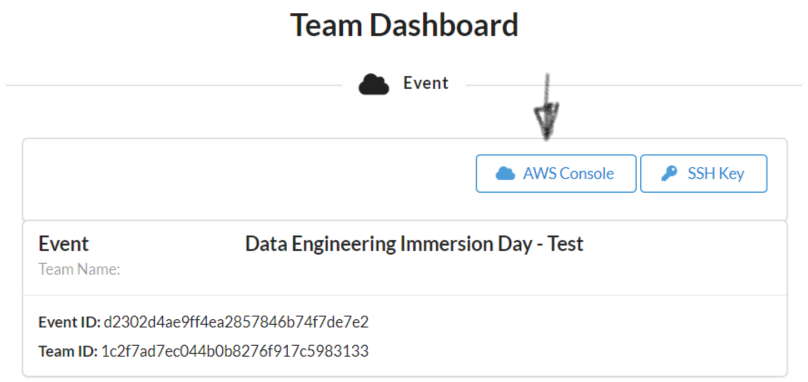
Glue Lab Role
mod-3fccddd609114925-GlueLabRole-YLTJA13WW6WT

S3BucketWorkgroupA
mod-3fccddd609114925-s3bucketworkgroupa-tbon3m1mkunh

S3BucketWorkgroupB
mod-3fccddd609114925-s3bucketworkgroupb-18ygl8nfp8ead

WorkgroupManagerUser
mod-3fccddd609114925-WorkgroupManagerUser-5IVE0UQNIBG4

3. On the Team Dashboard, please click AWS Console to log into the AWS Management Console:



Team Dashboard

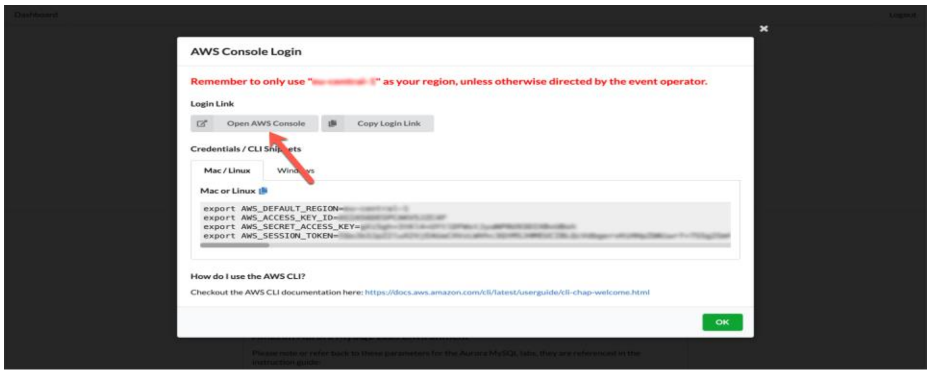
Event

AWS Console | SSH Key

Event Team Name:	Data Engineering Immersion Day - Test

Event ID: d2302d4ae9ff4ea2857846b74f7de7e2
Team ID: 1c2f7ad7ec044b0b8276f917c5983133

4. Click Open AWS Console. For the purposes of this workshop, you will not need to use command line and API access credentials:



AWS Console Login

Remember to only use "[redacted]" as your region, unless otherwise directed by the event operator.

Login Link

Open AWS Console | Copy Login Link

Credentials / CLI Snippets

Mac / Linux | Windows

Mac or Linux

```
export AWS_DEFAULT_REGION=[redacted]
export AWS_ACCESS_KEY_ID=[redacted]
export AWS_SECRET_ACCESS_KEY=[redacted]
export AWS_SESSION_TOKEN=[redacted]
```

How do I use the AWS CLI?

Checkout the AWS CLI documentation here: https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-welcome.html

OK

Once you have completed these steps, you can continue with the rest of this lab.

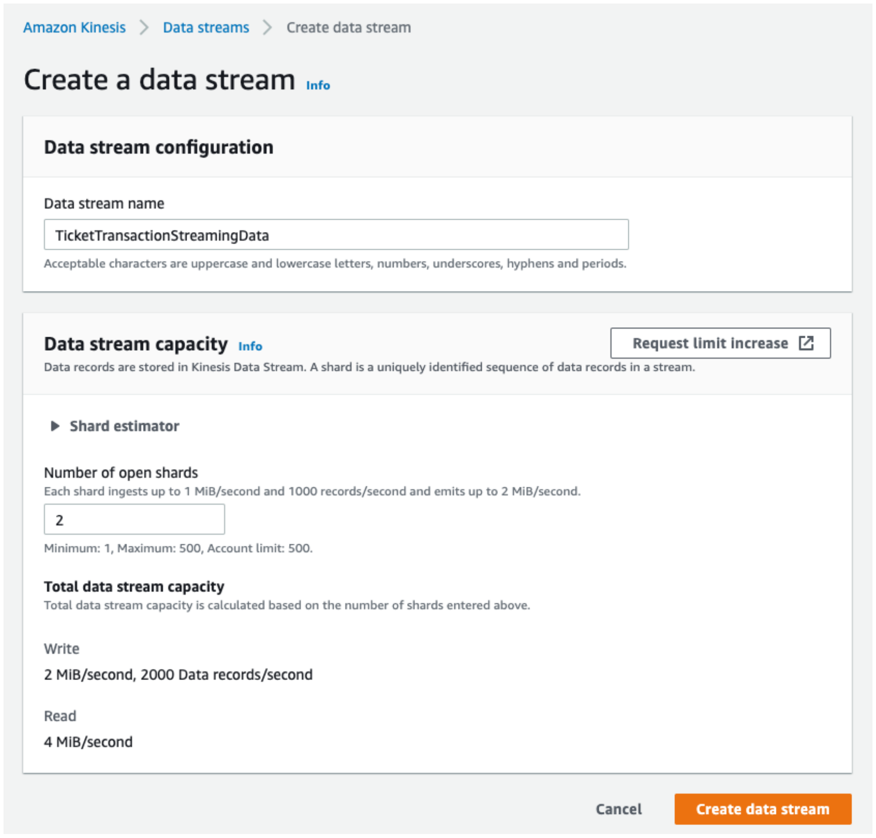
Setup Streaming Data Generator

We need an Amazon Kinesis Data Generator (Amazon KDG) to simulate the streaming data. If you have set up Amazon KDG with Kinesis Clickstream Lab, you should be able to reuse the tool. Otherwise, please follow the instruction in Streaming Data Prelab to launch the CloudFormation template, in order to set up your Amazon Kinesis Data Generator.

After the KDG setup is completed, you can find a URL from the output tab of the Streaming Data Prelab CloudFormation Stack, with the key name KinesisDataGeneratorUrl. Make sure you can login to the console using the username and password you provided when launching your prelab CloudFormation template. Bookmark the URL for further use.

Create Kinesis Data Stream

1. Navigate to **AWS Kinesis** console by using this link
2. Click "**Create data stream**"
3. Put "**TicketTransactionStreamingData**" as data stream name and put number of open shards as 2, then click "**Create data stream**".


Amazon Kinesis > Data streams > Create data stream

Create a data stream Info

Data stream configuration

Data stream name

TicketTransactionStreamingData

Acceptable characters are uppercase and lowercase letters, numbers, underscores, hyphens and periods.

Data stream capacity Info

Request limit increase

Data records are stored in Kinesis Data Stream. A shard is a uniquely identified sequence of data records in a stream.

▶ Shard estimator

Number of open shards

Each shard ingests up to 1 MiB/second and 1000 records/second and emits up to 2 MiB/second.

2

Minimum: 1, Maximum: 500, Account limit: 500.

Total data stream capacity

Total data stream capacity is calculated based on the number of shards entered above.

Write

2 MiB/second, 2000 Data records/second

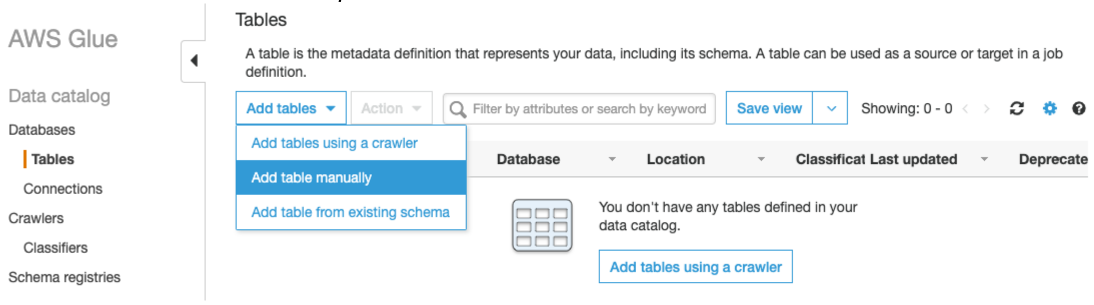
Read

4 MiB/second

Cancel | Create data stream

Create Table for Kinesis Stream Source in Glue Data Catalog

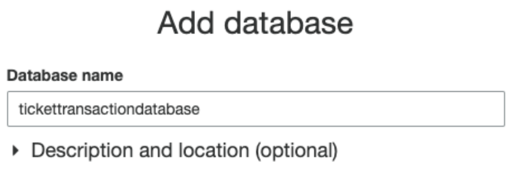
1. Navigate to **AWS Glue** console by using this link
2. On the AWS Glue menu, select Tables



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.

You don't have any tables defined in your data catalog.

3. Put **TicketTransactionStreamData** as the table name
4. Click Add database and put **tickettransactiondatabase** as the database name, and click create.


Add database

Database name

tickettransactiondatabase

▸ Description and location (optional)

5. Using drop down to select the database we just created, and click Next

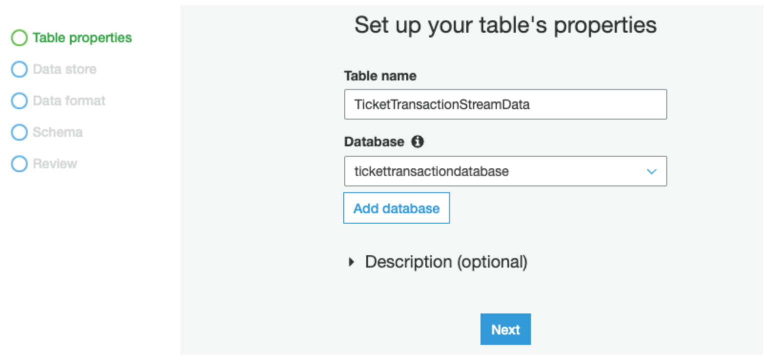
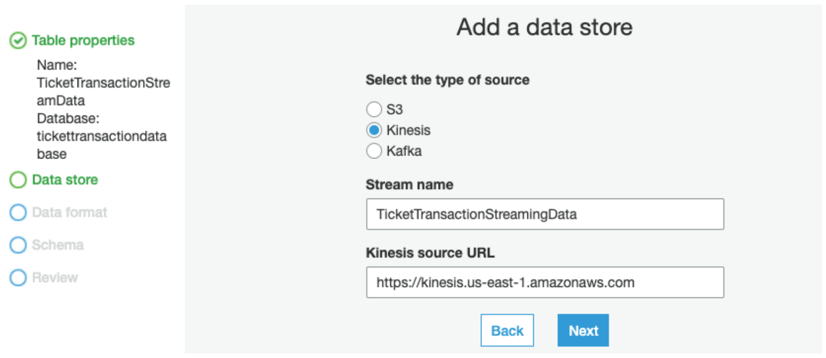


Table name: TicketTransactionStreamData

Database: tickettransactiondatabase

6. Select **Kinesis** as the source, put stream name as **TicketTransactionStreamingData**, for the kinesis source URL, put in https://kinesis.us-east-1.amazonaws.com, click **Next.**



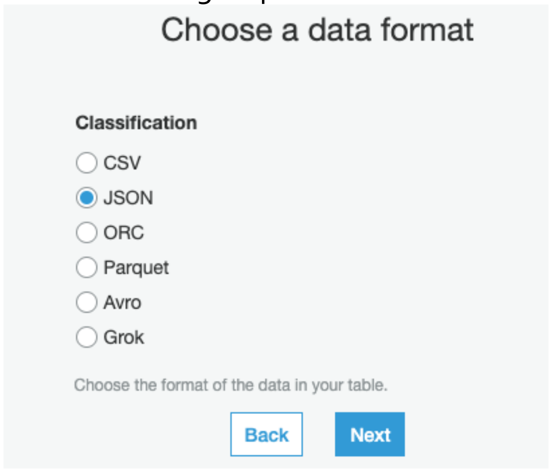
Name: TicketTransactionStreamData
Database: tickettransactiondatabase

Select the type of source: S3 [✓ Kinesis] Kafka

Stream name: TicketTransactionStreamingData

Kinesis source URL: https://kinesis.us-east-1.amazonaws.com

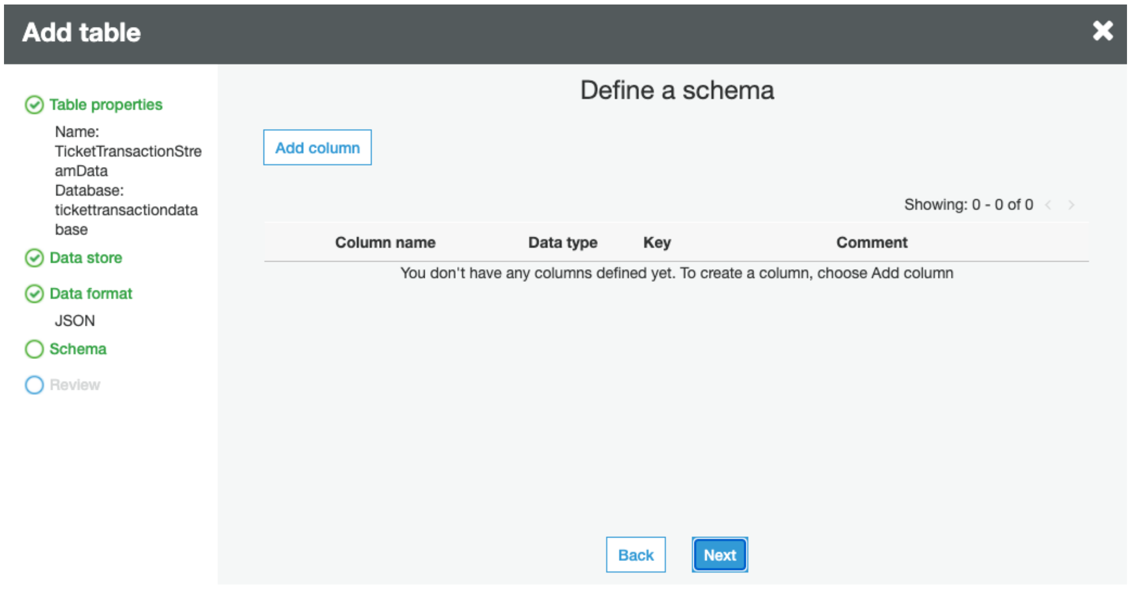
7. Choose **JSON** as the incoming data format, as we will trigger JSON payload from Kinesis Data Generator in following steps. Click **Next**.



Classification: CSV [✓ JSON] ORC Parquet Avro Grok

Choose the format of the data in your table.

8. Leave the schema as empty, as we will enable schema detection feature when defining glue stream job. Click **Next**.



Name: TicketTransactionStreamData
Database: tickettransactiondatabase

Data format: JSON

Showing: 0 - 0 of 0

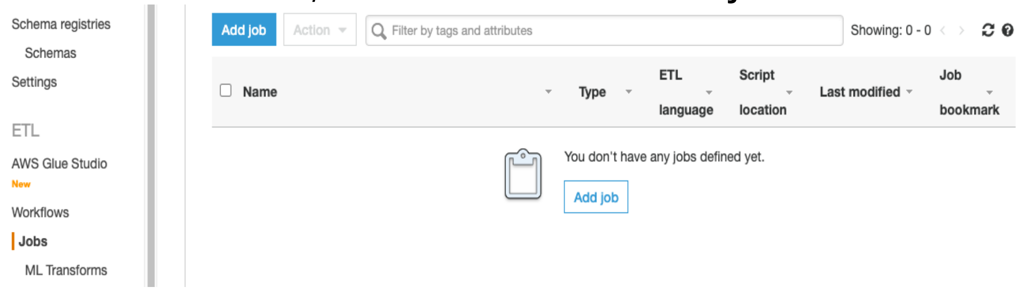
Column name	Data type	Key	Comment

You don't have any columns defined yet. To create a column, choose Add column

9. Review all the details and click **Finish**.

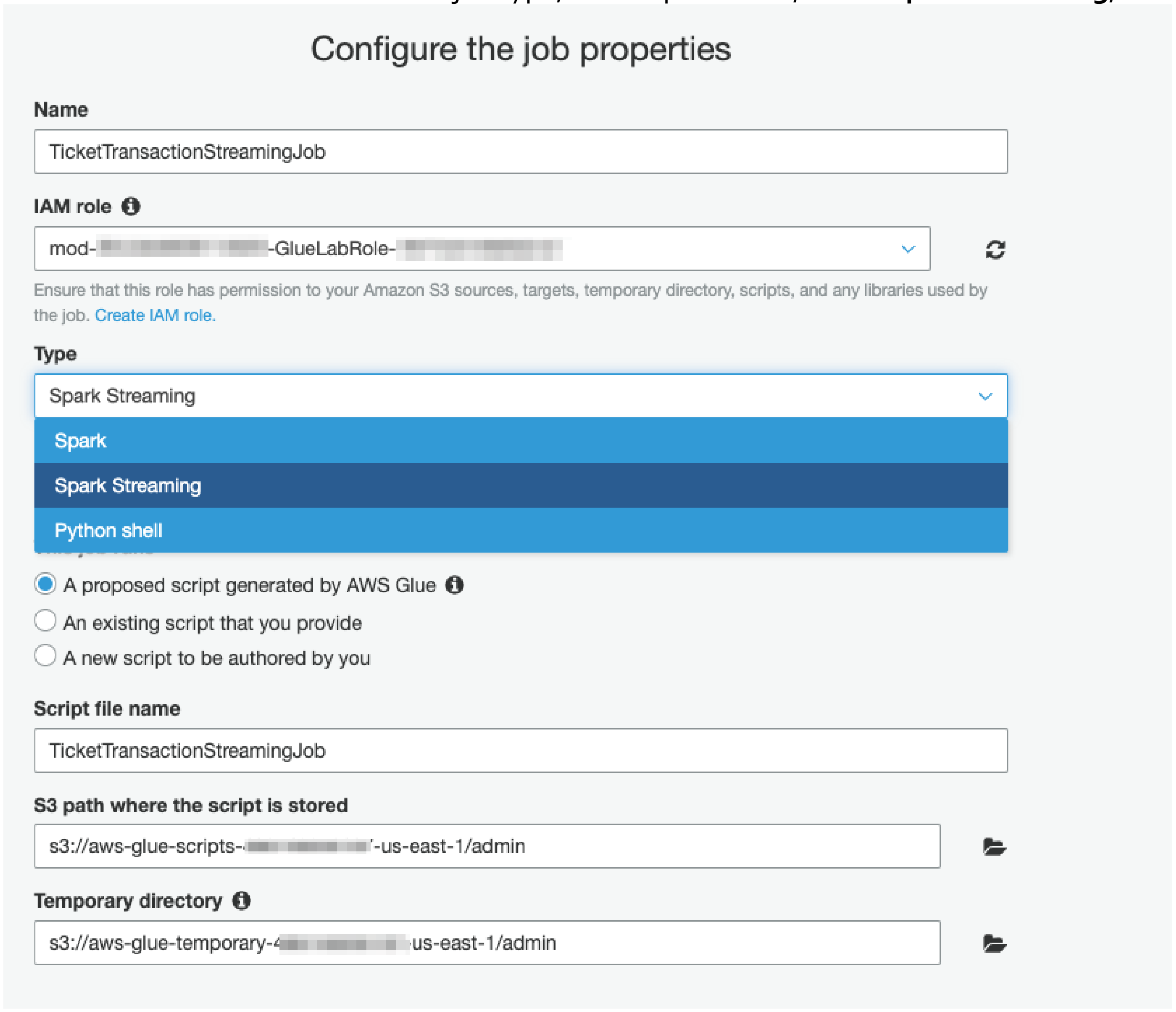
Create and trigger Glue Stream job

1. Navigate to **AWS Glue** console
2. On the AWS Glue menu, select **Jobs** and then click **Add job**



You don't have any jobs defined yet.

3. Put **TicketTransactionStreamingJob** as the job name, select the IAM role with "GlueLabRole" in the name. For job type, use dropdown list, select **Spark Streaming**;



Configure the job properties

Name: TicketTransactionStreamingJob

IAM role: mod-[illegible]-GlueLabRole-[illegible]

Ensure that this role has permission to your Amazon S3 sources, targets, temporary directory, scripts, and any libraries used by the job. Create IAM role.

Type: Spark Streaming

(●) A proposed script generated by AWS Glue
( ) An existing script that you provide
( ) A new script to be authored by you

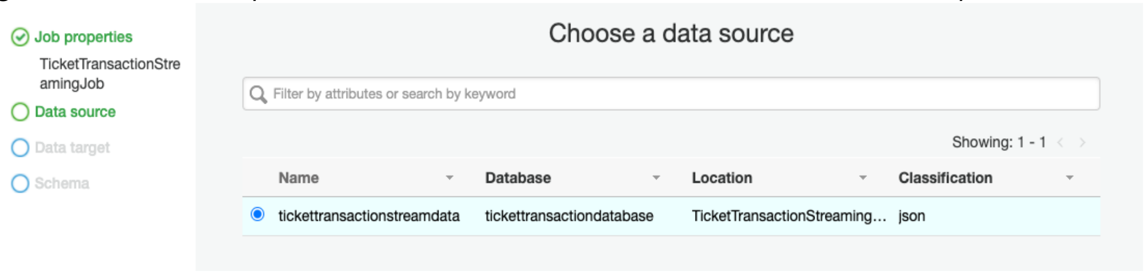
Script file name: TicketTransactionStreamingJob

S3 path where the script is stored: s3://aws-glue-scripts-[illegible]-us-east-1/admin

Temporary directory: s3://aws-glue-temporary-[illegible]-us-east-1/admin

4. leave the rest configurations as is and click **Next**.

5. For Data source, select the data source **tickettransactionstreamdata**, then click **Next**.



Job properties
TicketTransactionStreamingJob
Data source
Data target
Schema

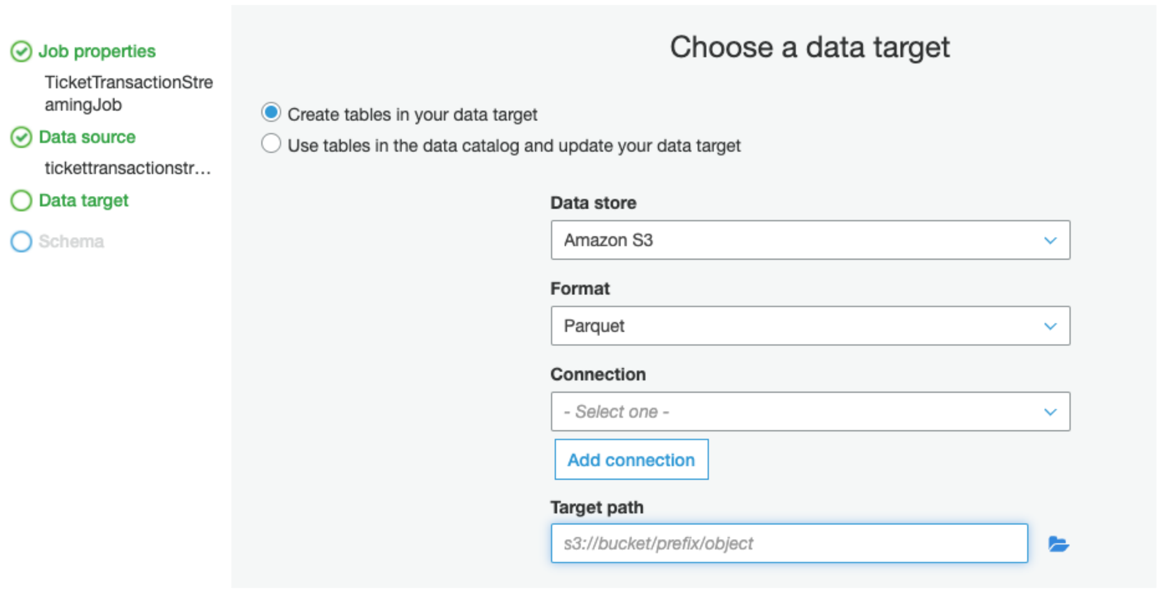
Choose a data source

Filter by attributes or search by keyword

Showing: 1 - 1

	Name	Database	Location	Classification
●	tickettransactionstreamdata	tickettransactiondatabase	TicketTransactionStreaming...	json

6. In Data target, select **Create tables in your data target**. In Data store dropdown list, select **Amazon S3**. Select **Parquet** format from dropdown list.



Job properties
TicketTransactionStreamingJob
Data source
tickettransactionstr...
Data target
Schema

Choose a data target

(●) Create tables in your data target
( ) Use tables in the data catalog and update your data target

Data store: Amazon S3

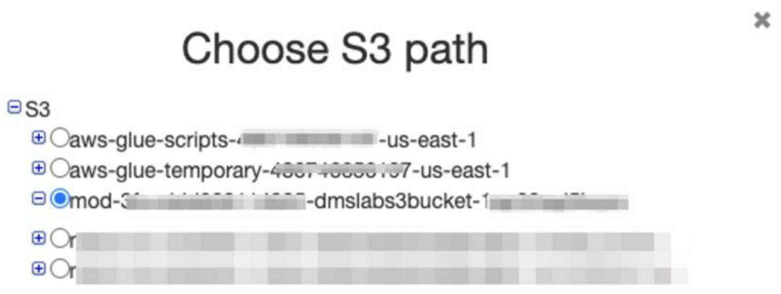
Format: Parquet

Connection: - Select one -

Add connection

Target path: s3://bucket/prefix/object

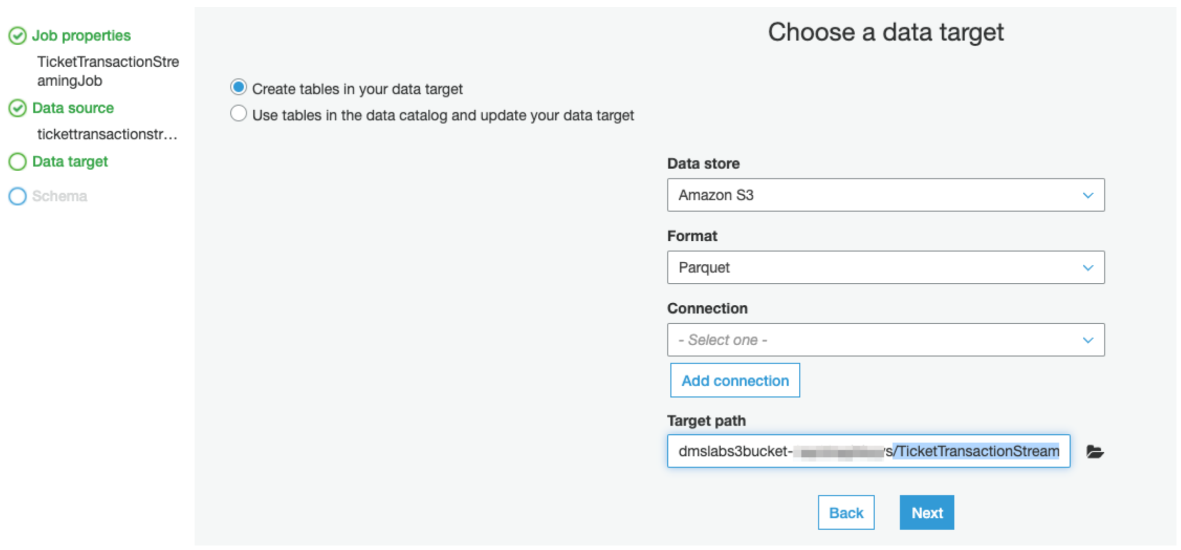
7. Click the **folder** button next to **Target path** to select a S3 bucket. From the pop-up window, select a S3 bucket name that contains "**dmslabs3bucket"**.



Choose S3 path

S3
- aws-glue-scripts-[redacted]-us-east-1
- aws-glue-temporary-[redacted]-us-east-1
- ● mod-[redacted]-dmslabs3bucket-[redacted]
- [redacted]
- [redacted]

8. Make sure you add a path at the end /**TicketTransactionStream**



Choose a data target

[✓ Create tables in your data target]
Use tables in the data catalog and update your data target

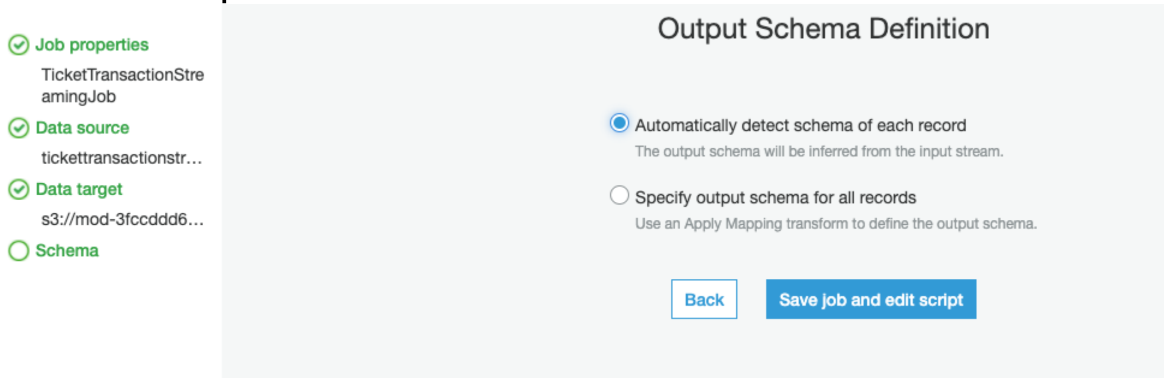
Data store: Amazon S3

Format: Parquet

Connection: - Select one -

Target path: dmslabs3bucket-[redacted]s/TicketTransactionStream

9. Make sure you select **Automatically detect schema of each record**, then click **Save job and edit script**.

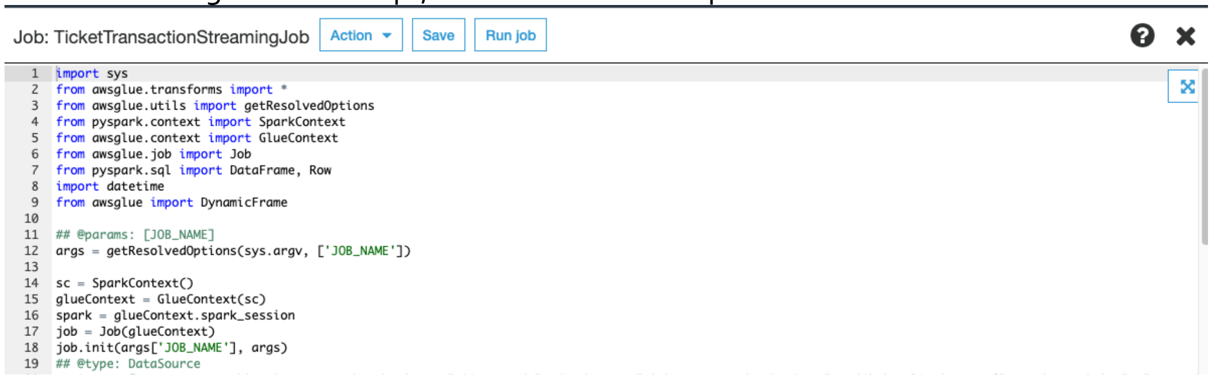


Output Schema Definition

[✓ Automatically detect schema of each record]
The output schema will be inferred from the input stream.

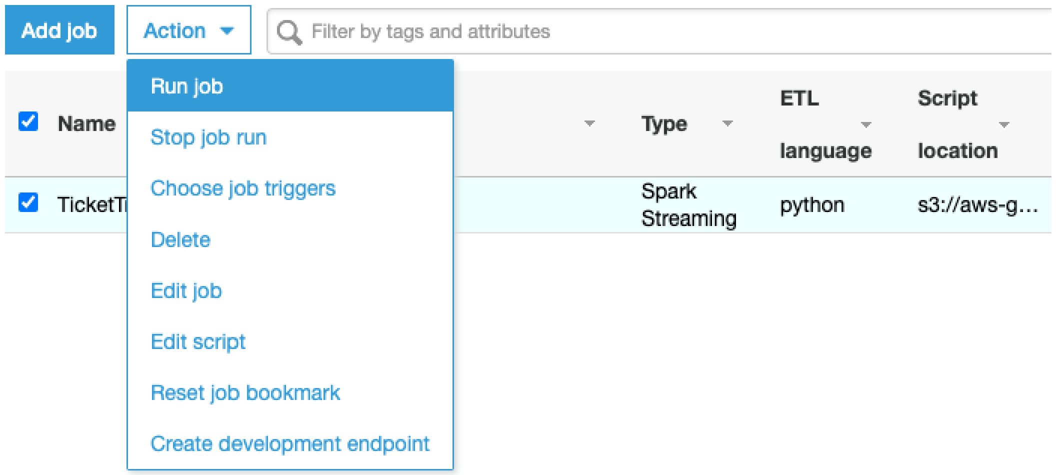
Specify output schema for all records
Use an Apply Mapping transform to define the output schema.

10. Review the generated script, click **Save** and then quit the editor.

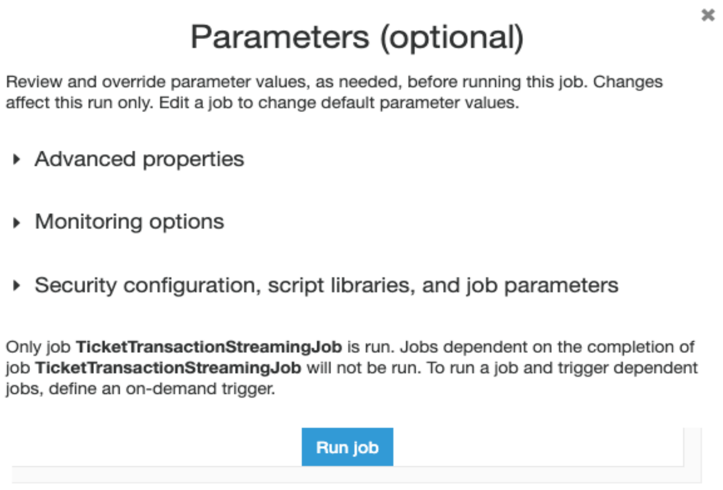


```
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job
from pyspark.sql import DataFrame, Row
import datetime
from awsglue import DynamicFrame

## @params: [JOB_NAME]
args = getResolvedOptions(sys.argv, ['JOB_NAME'])

sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args['JOB_NAME'], args)
## @type: DataSource
```

11. Select the **TicketTransactionStreamingJob** we just created, from the Action dropdown list, select **Run job**.



12. Leave the optional parameters as default and click **Run job** to trigger the Glue Stream Job


Parameters (optional)

Review and override parameter values, as needed, before running this job. Changes affect this run only. Edit a job to change default parameter values.

- Advanced properties
- Monitoring options
- Security configuration, script libraries, and job parameters

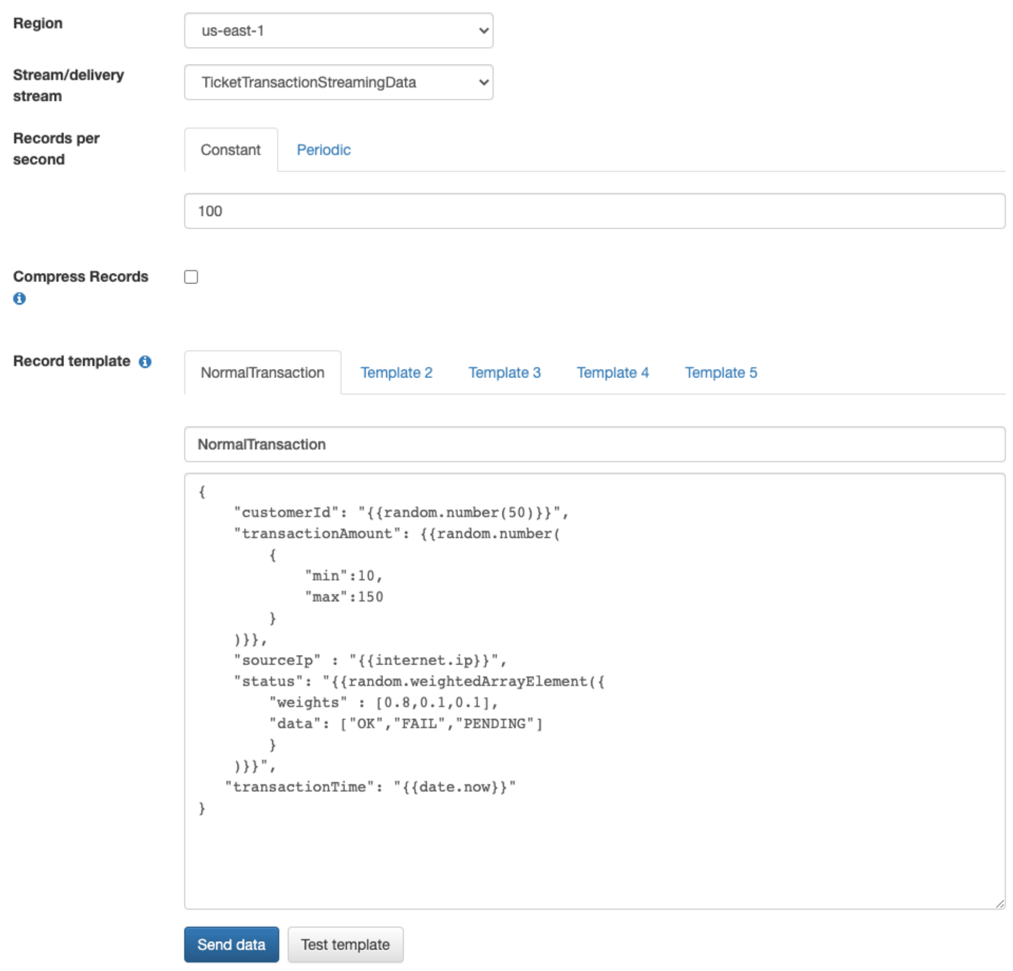
Only job **TicketTransactionStreamingJob** is run. Jobs dependent on the completion of job **TicketTransactionStreamingJob** will not be run. To run a job and trigger dependent jobs, define an on-demand trigger.

Trigger stream data from KDG

1. Launch KDG using the URL you bookmarked from the lab setup, login using the username and password you specified when deploying the CloudFormation stack.

2. Make sure you select **us-east-1** region, from the dropdown list, select the **TicketTransactionStreamingData** as the target Kinesis stream, leave Records per second as default (**100** records per second); for the record template, type in **NormalTransaction** as the payload name, and copy the template payload as below:

```
{
   "customerId": "{{random.number(50)}}",
        "transactionAmount": {{random.number(
          {
            "min":10,
            "max":150
          }
        )}},
        "sourceIp" : "{{internet.ip}}",
        "status": "{{random.weightedArrayElement({
          "weights" : [0.8,0.1,0.1],
          "data": ["OK","FAIL","PENDING"]
          }
        )}}",
       "transactionTime": "{{date.now}}"
     }
```

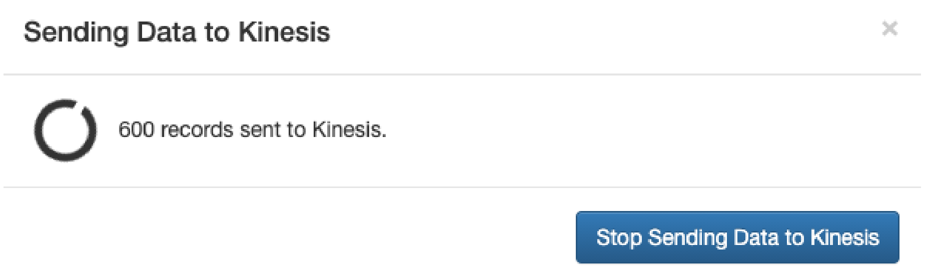


Region	us-east-1
Stream/delivery stream	TicketTransactionStreamingData
Records per second	Constant: 100
Compress Records	☐
Record template	NormalTransaction

```
{
    "customerId": "{{random.number(50)}}",
    "transactionAmount": {{random.number(
        {
            "min":10,
            "max":150
        }
    )}},
    "sourceIp" : "{{internet.ip}}",
    "status": "{{random.weightedArrayElement({
        "weights" : [0.8,0.1,0.1],
        "data": ["OK","FAIL","PENDING"]
        }
    )}}",
   "transactionTime": "{{date.now}}"
}
```

To learn more about what the payload will look like when sending from KDG simulator, refer to the document as this link, https://awslabs.github.io/amazon-kinesis-data-generator/web/help.html

3. Click **Send data** to trigger the simulated ticket purchasing transaction streaming data.



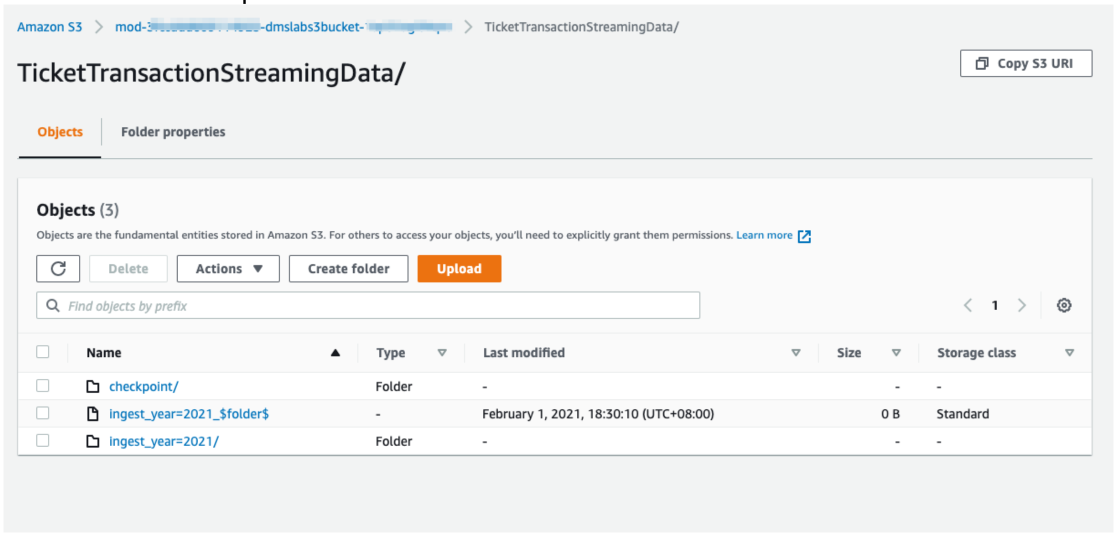
Sending Data to Kinesis

600 records sent to Kinesis.

Verify the Glue stream job

1. Navigate to **Amazon S3** console by using this link https://s3.console.aws.amazon.com/s3/home?region=us-east-1

2. Navigate to the S3 bucket path we've set as Glue Stream Job target, note the folder structure of the processed data.



Amazon S3 > mod-[redacted]-dmslabs3bucket-[redacted] > TicketTransactionStreamingData/

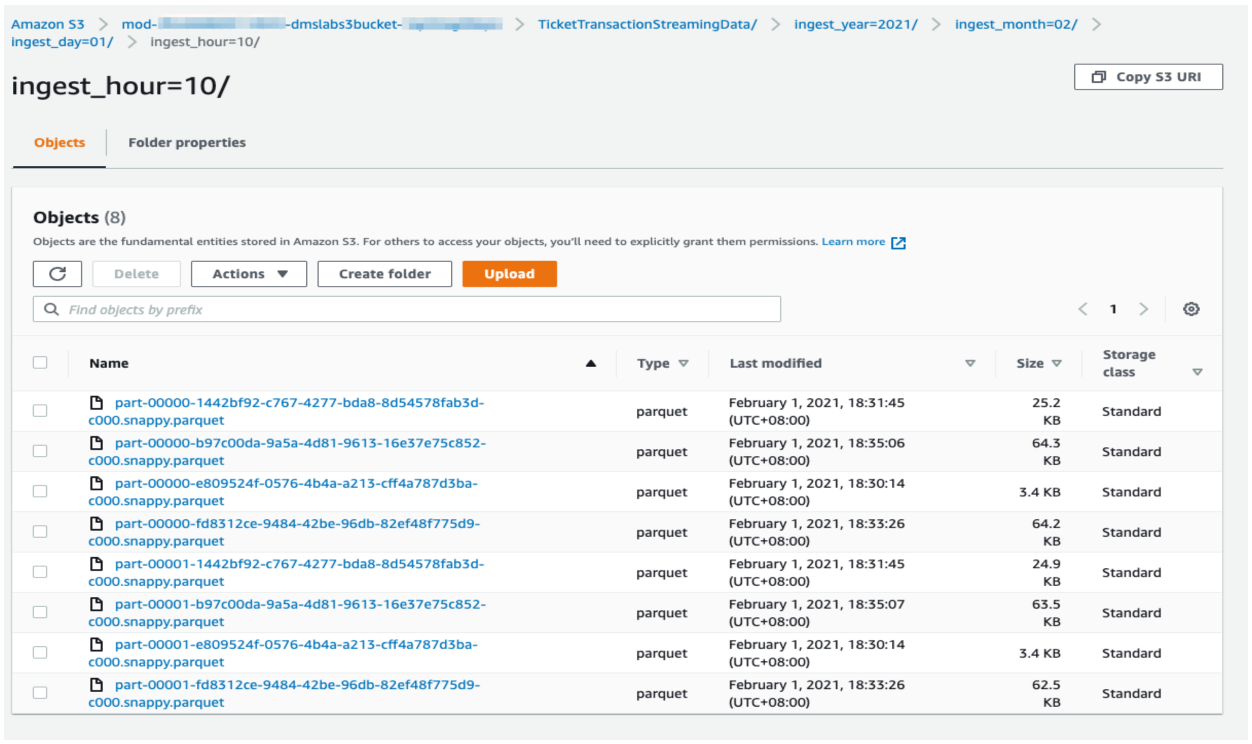
TicketTransactionStreamingData/

Objects (3)

Objects are the fundamental entities stored in Amazon S3. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
checkpoint/	Folder	-	-	-
ingest_year=2021_$folder$	-	February 1, 2021, 18:30:10 (UTC+08:00)	0 B	Standard
ingest_year=2021/	Folder	-	-	-

3. Check the folder content using current date and time as the folder name. Verify that the streaming data has been transformed into parquet format and persisted into corresponding folders.



Amazon S3 > mod-[redacted]-dmslabs3bucket-[redacted] > TicketTransactionStreamingData/ > ingest_year=2021/ > ingest_month=02/ > ingest_day=01/ > ingest_hour=10/

ingest_hour=10/

Copy S3 URI

Objects | Folder properties

Objects (8)

Objects are the fundamental entities stored in Amazon S3. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

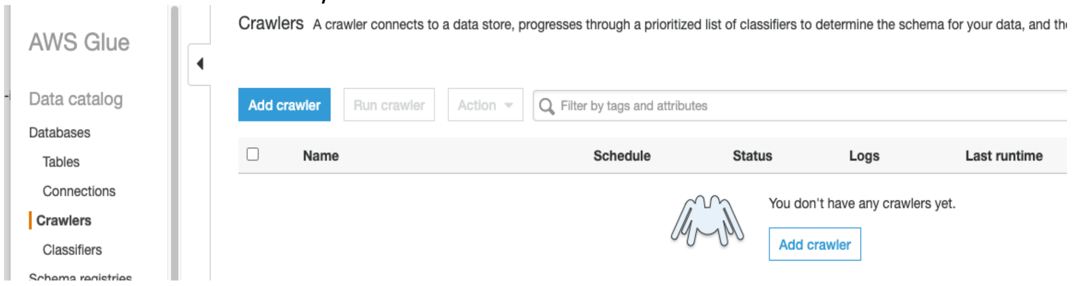
Delete | Actions | Create folder | Upload

Find objects by prefix

Name	Type	Last modified	Size	Storage class
part-00000-1442bf92-c767-4277-bda8-8d54578fab3d-c000.snappy.parquet	parquet	February 1, 2021, 18:31:45 (UTC+08:00)	25.2 KB	Standard
part-00000-b97c00da-9a5a-4d81-9613-16e37e75c852-c000.snappy.parquet	parquet	February 1, 2021, 18:35:06 (UTC+08:00)	64.3 KB	Standard
part-00000-e809524f-0576-4b4a-a213-cff4a787d3ba-c000.snappy.parquet	parquet	February 1, 2021, 18:30:14 (UTC+08:00)	3.4 KB	Standard
part-00000-fd8312ce-9484-42be-96db-82ef48f775d9-c000.snappy.parquet	parquet	February 1, 2021, 18:33:26 (UTC+08:00)	64.2 KB	Standard
part-00001-1442bf92-c767-4277-bda8-8d54578fab3d-c000.snappy.parquet	parquet	February 1, 2021, 18:31:45 (UTC+08:00)	24.9 KB	Standard
part-00001-b97c00da-9a5a-4d81-9613-16e37e75c852-c000.snappy.parquet	parquet	February 1, 2021, 18:35:07 (UTC+08:00)	63.5 KB	Standard
part-00001-e809524f-0576-4b4a-a213-cff4a787d3ba-c000.snappy.parquet	parquet	February 1, 2021, 18:30:14 (UTC+08:00)	3.4 KB	Standard
part-00001-fd8312ce-9484-42be-96db-82ef48f775d9-c000.snappy.parquet	parquet	February 1, 2021, 18:33:26 (UTC+08:00)	62.5 KB	Standard

Create Glue Crawler for the transformed data

1. Navigate to AWS Glue console
2. On the AWS Glue menu, select **Crawlers** and click **Add crawler**.



Crawlers A crawler connects to a data store, progresses through a prioritized list of classifiers to determine the schema for your data, and th

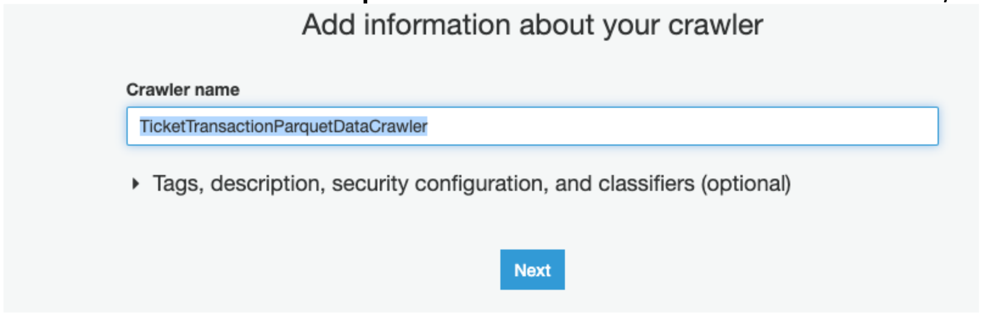
Add crawler | Run crawler | Action | Filter by tags and attributes

Name	Schedule	Status	Logs	Last runtime

You don't have any crawlers yet.

Add crawler

3. Put **TicketTransactionParquetDataCrawler** as the name of the crawler, click **Next**.



Add information about your crawler

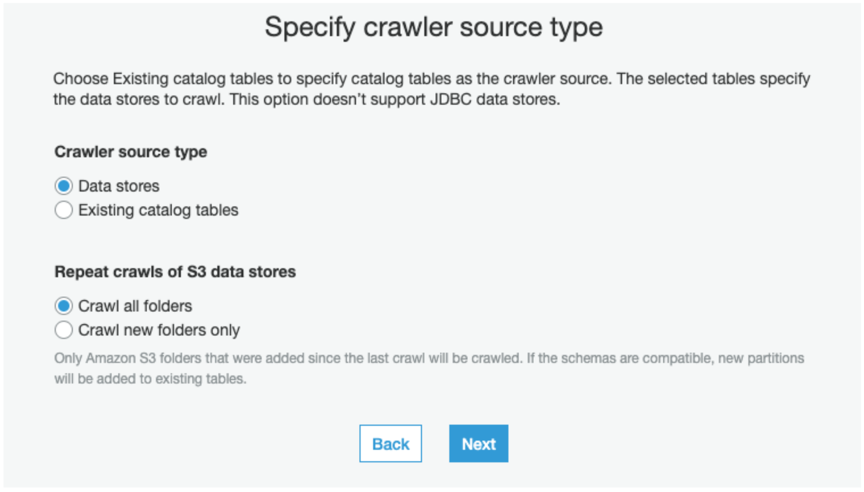
Crawler name

TicketTransactionParquetDataCrawler

Tags, description, security configuration, and classifiers (optional)

Next

4. Leave the default to specify **Data stores** as Crawler source type and **Crawl all folders**, click **Next**.



Specify crawler source type

Choose Existing catalog tables to specify catalog tables as the crawler source. The selected tables specify the data stores to crawl. This option doesn't support JDBC data stores.

Crawler source type

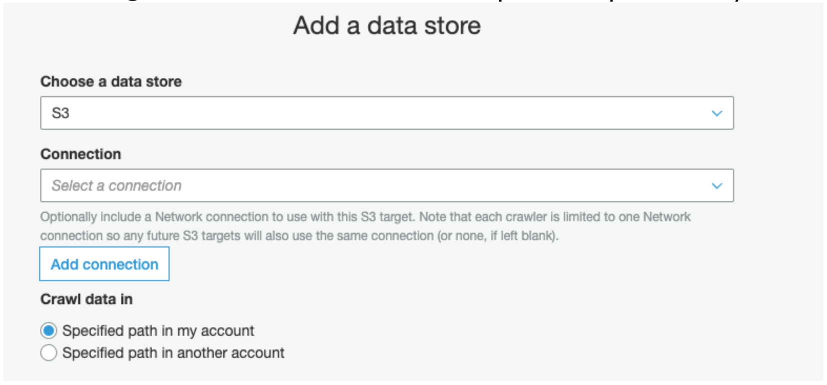
[✓ Data stores]
Existing catalog tables

Repeat crawls of S3 data stores

[✓ Crawl all folders]
Crawl new folders only

Only Amazon S3 folders that were added since the last crawl will be crawled. If the schemas are compatible, new partitions will be added to existing tables.

5. Choose S3 as data store and choose Specified path in my account.



Add a data store

Choose a data store

S3

Connection

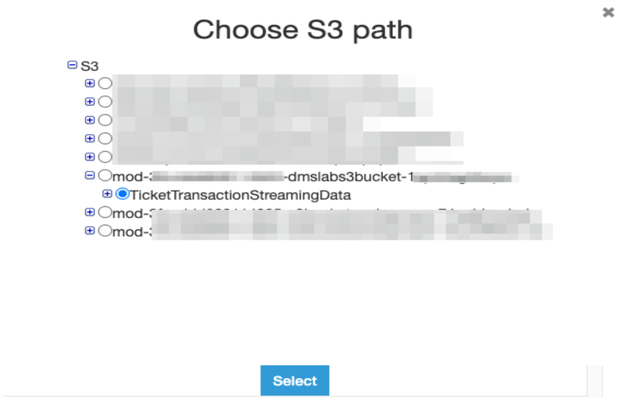
Select a connection

Optionally include a Network connection to use with this S3 target. Note that each crawler is limited to one Network connection so any future S3 targets will also use the same connection (or none, if left blank).

Crawl data in

[✓ Specified path in my account]
Specified path in another account

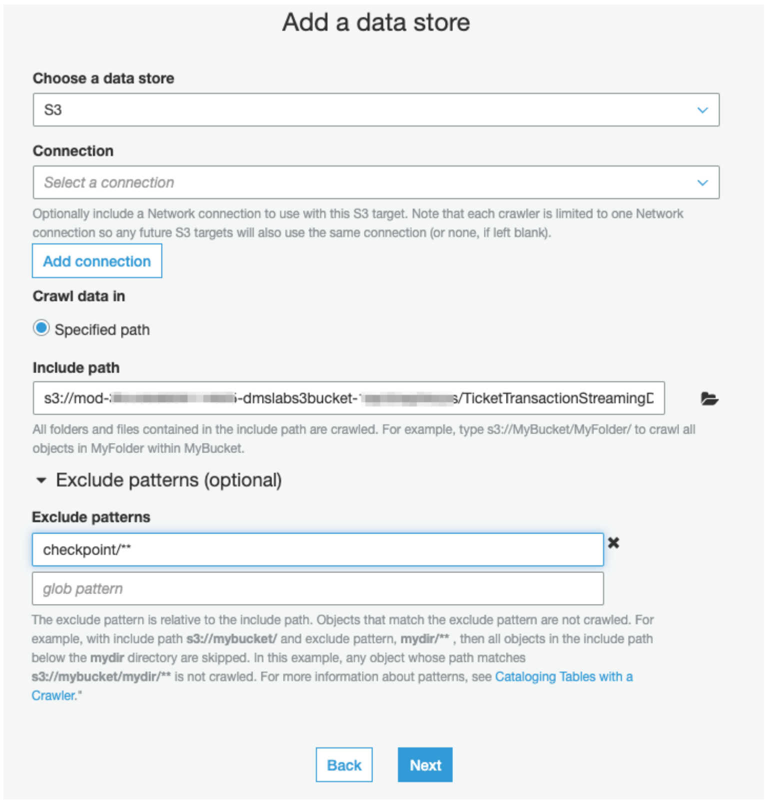
6. Click the icon next to Include path input to select the S3 bucket. Make sure you select the folder **TicketTransactionStreamingData**. Click **Select**.



Choose S3 path

S3
mod-[redacted]-dmslabs3bucket-1[redacted]
[✓ TicketTransactionStreamingData]
mod-[redacted]
mod-[redacted]

7. Expand the **Exclude patterns**, put **checkpoint/**** to exclude the data in checkpoint folder. Review the current input and click **Next**.



Add a data store

Choose a data store

S3

Connection

Select a connection

Optionally include a Network connection to use with this S3 target. Note that each crawler is limited to one Network connection so any future S3 targets will also use the same connection (or none, if left blank).

Crawl data in

[✓ Specified path]

Include path

s3://mod-[redacted]-dmslabs3bucket-[redacted]s/TicketTransactionStreamingD

All folders and files contained in the include path are crawled. For example, type s3://MyBucket/MyFolder/ to crawl all objects in MyFolder within MyBucket.

Exclude patterns (optional)

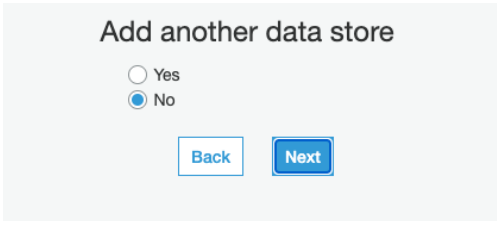
Exclude patterns

checkpoint/**

glob pattern

The exclude pattern is relative to the include path. Objects that match the exclude pattern are not crawled. For example, with include path **s3://mybucket/** and exclude pattern, **mydir/**** , then all objects in the include path below the **mydir** directory are skipped. In this example, any object whose path matches **s3://mybucket/mydir/**** is not crawled. For more information about patterns, see Cataloging Tables with a Crawler.

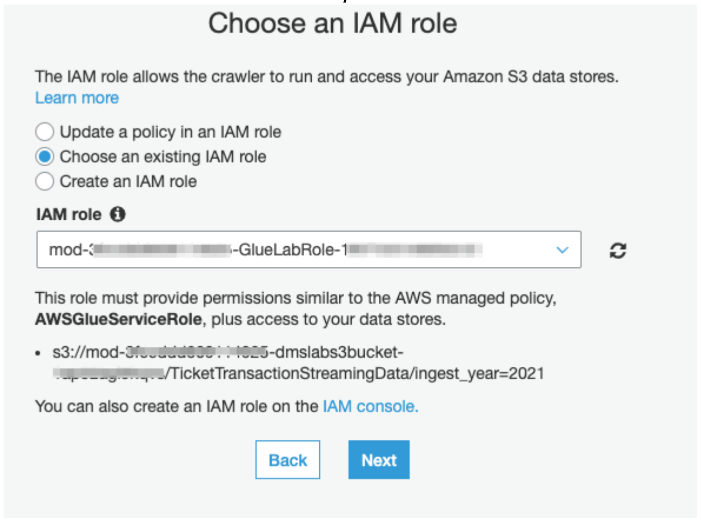
8. Select No to indicate no other data store needed, then click Next.



Add another data store

Yes [✓ No]

9. Choose an existing IAM role, using the dropdown list to select the role with **GlueLabRole** in the name, click **Next.**



Choose an IAM role

The IAM role allows the crawler to run and access your Amazon S3 data stores. Learn more

Update a policy in an IAM role
[✓ Choose an existing IAM role]
Create an IAM role

IAM role

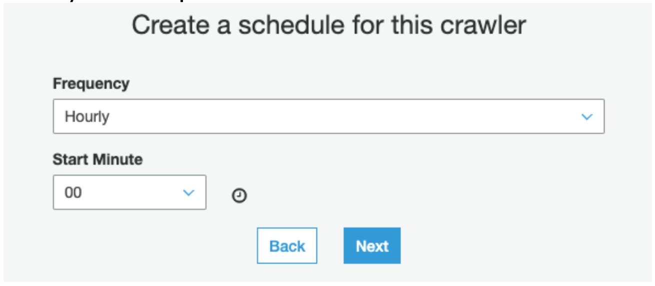
mod-[redacted]-GlueLabRole-1[redacted]

This role must provide permissions similar to the AWS managed policy, **AWSGlueServiceRole**, plus access to your data stores.

- s3://mod-[redacted]-dmslabs3bucket-[redacted]/TicketTransactionStreamingData/ingest_year=2021

You can also create an IAM role on the IAM console.

10. As the data is partitioned to hour, so we set the crawler to run every hour to make sure newly added partition is added. Click **Next**.

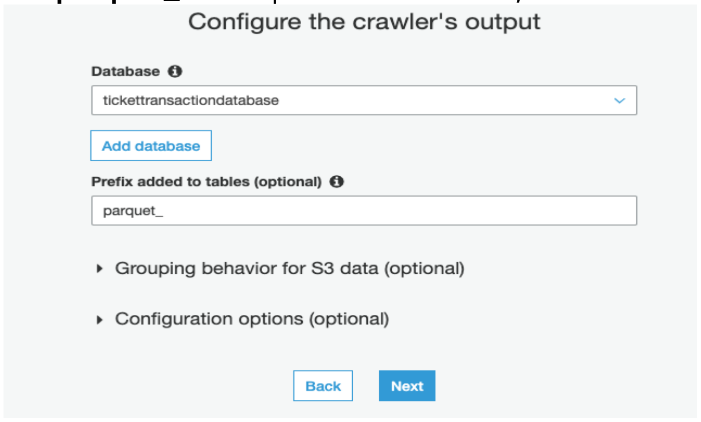


Create a schedule for this crawler

Frequency: Hourly

Start Minute: 00

11. Using the dropdown list to select **tickettransactiondatabase** as the output database, use **parquet_** as the prefix for the table, click **Next**.



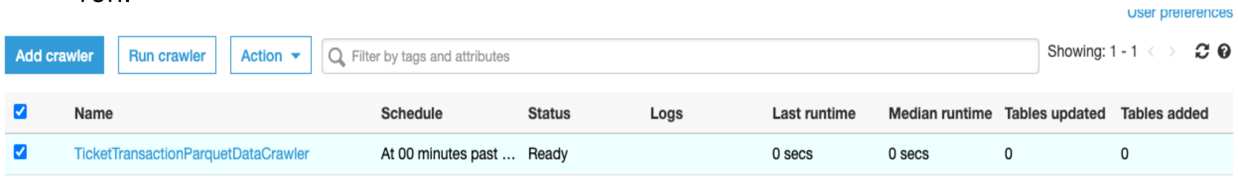
Configure the crawler's output

Database: tickettransactiondatabase

Prefix added to tables (optional): parquet_

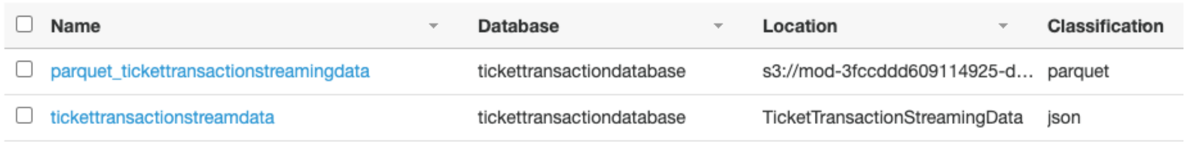
Add database
Grouping behavior for S3 data (optional)
Configuration options (optional)
Back
Next

12. Review the crawler configuration and click **Finish** to create the crawler.
13. Once the crawler is created, select the crawler and click **Run crawler** to trigger the first run.



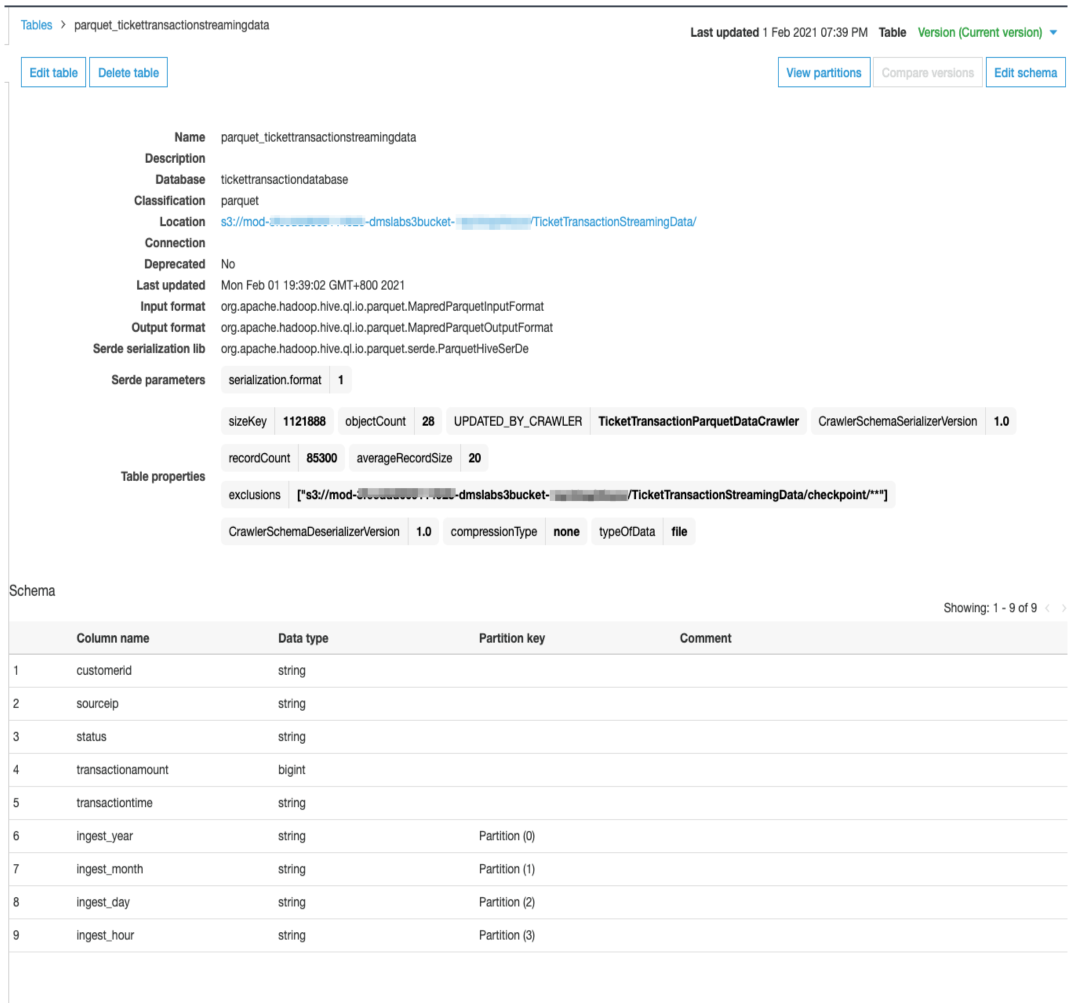
✓	Name	Schedule	Status	Logs	Last runtime	Median runtime	Tables updated	Tables added
✓	TicketTransactionParquetDataCrawler	At 00 minutes past …	Ready		0 secs	0 secs	0	0

14. When crawler job stopped, go to Glue Data catalog, under Tables, verify that **parquet_tickettransactionstreamingdata** table is listed.



	Name	Database	Location	Classification
☐	parquet_tickettransactionstreamingdata	tickettransactiondatabase	s3://mod-3fccddd609114925-d…	parquet
☐	tickettransactionstreamdata	tickettransactiondatabase	TicketTransactionStreamingData	json

15. Click the **parquet_tickettransactionstreamingdata** table, verify that Glue has correctly identified the streaming data format while transforming source data from Json format to Parquet.



Tables > parquet_tickettransactionstreamingdata

Last updated 1 Feb 2021 07:39 PM **Table** Version (Current version)

Edit table | Delete table | View partitions | Compare versions | Edit schema

Name	parquet_tickettransactionstreamingdata
Description	
Database	tickettransactiondatabase
Classification	parquet
Location	s3://mod-[redacted]-dmslabs3bucket-[redacted]/TicketTransactionStreamingData/
Connection	
Deprecated	No
Last updated	Mon Feb 01 19:39:02 GMT+800 2021
Input format	org.apache.hadoop.hive.ql.io.parquet.MapredParquetInputFormat
Output format	org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputFormat
Serde serialization lib	org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe
Serde parameters	serialization.format 1
Table properties	sizeKey 1121888; objectCount 28; UPDATED_BY_CRAWLER TicketTransactionParquetDataCrawler; CrawlerSchemaSerializerVersion 1.0; recordCount 85300; averageRecordSize 20; exclusions ["s3://mod-[redacted]-dmslabs3bucket-[redacted]/TicketTransactionStreamingData/checkpoint/**"]; CrawlerSchemaDeserializerVersion 1.0; compressionType none; typeOfData file

Schema

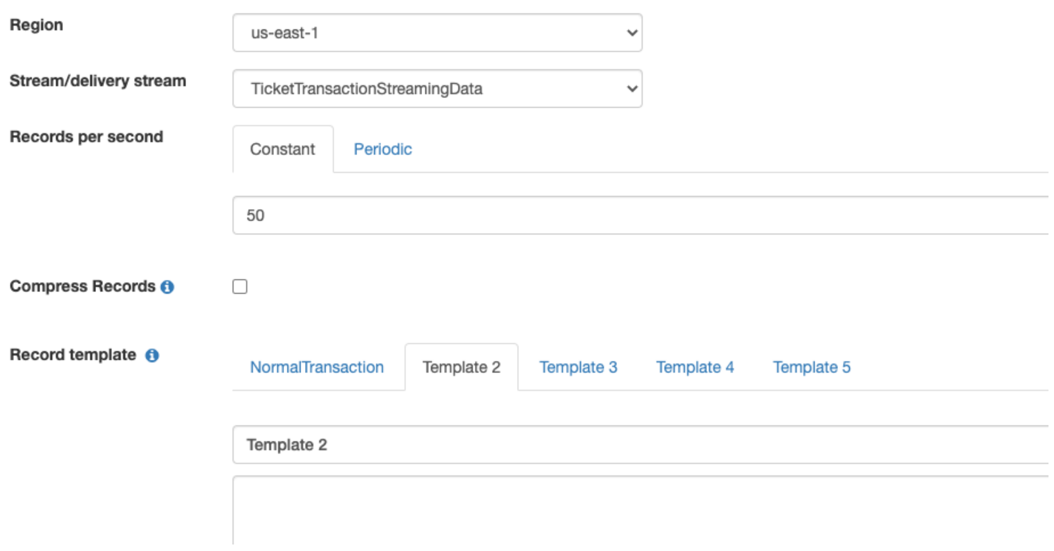
Showing: 1 - 9 of 9

	Column name	Data type	Partition key	Comment
1	customerid	string		
2	sourceip	string		
3	status	string		
4	transactionamount	bigint		
5	transactiontime	string		
6	ingest_year	string	Partition (0)	
7	ingest_month	string	Partition (1)	
8	ingest_day	string	Partition (2)	
9	ingest_hour	string	Partition (3)	

Trigger abnormal transaction data from KDG

1. Keep the KDG streaming data running, open another browser and launch KDG using the URL you bookmarked from the lab setup, login using the username and password you provided when launching the CloudFormation template.

2. Make sure you select **us-east-1** region, from the dropdown list, select the **TicketTransactionStreamingData** as the target Kinesis stream, put Records per second as 1; click Template 2, and prepare to copy abnormal transaction data,


Region: us-east-1

Stream/delivery stream: TicketTransactionStreamingData

Records per second: Constant | Periodic

50

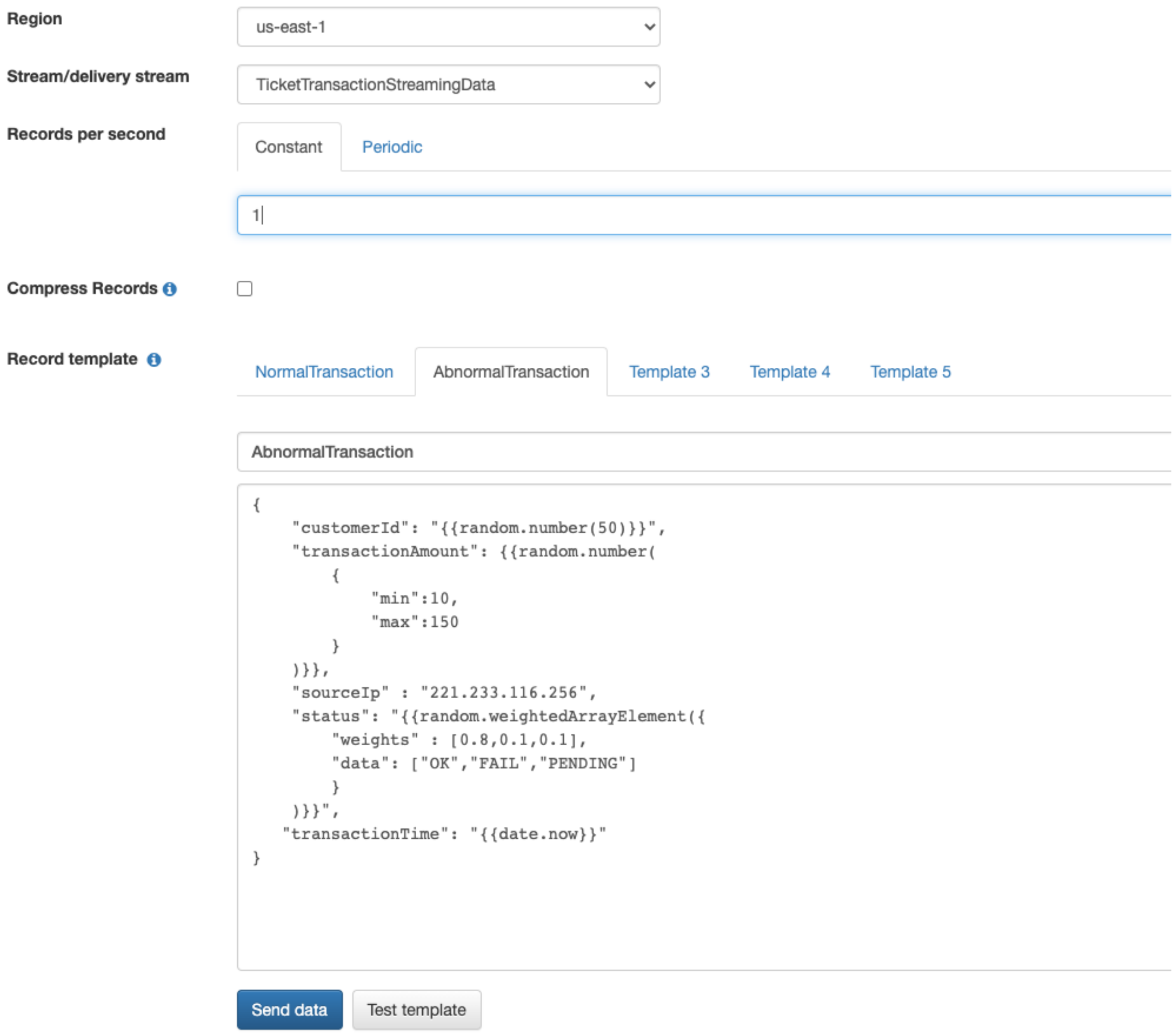
Compress Records ☐

Record template: NormalTransaction | Template 2 | Template 3 | Template 4 | Template 5

Template 2

3. for the record template, type in **NormalTransaction** as the payload name, and copy the template payload as below:

```
{
  "customerId": "{{random.number(50)}}",
  "transactionAmount": {{random.number(
    {
      "min":10,
      "max":150
    }
  )}},
  "sourceIp" : "221.233.116.256",
  "status": "{{random.weightedArrayElement({
    "weights" : [0.8,0.1,0.1],
    "data": ["OK","FAIL","PENDING"]
    }
  )}}",
 "transactionTime": "{{date.now}}"
}
```



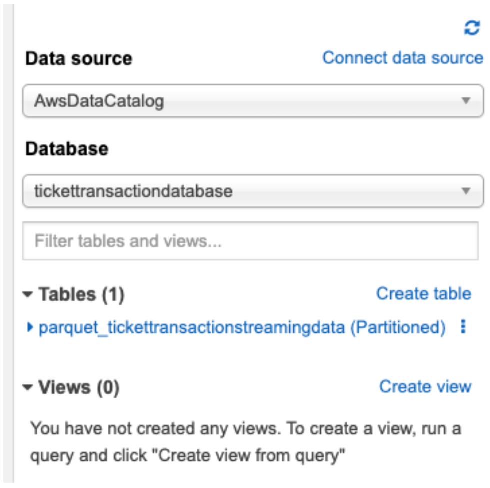
```
{
    "customerId": "{{random.number(50)}}",
    "transactionAmount": {{random.number(
        {
            "min":10,
            "max":150
        }
    )}},
    "sourceIp" : "221.233.116.256",
    "status": "{{random.weightedArrayElement({
        "weights" : [0.8,0.1,0.1],
        "data": ["OK","FAIL","PENDING"]
        }
    )}}",
   "transactionTime": "{{date.now}}"
}
```

4. Click Send data to simulate abnormal transactions (1 transaction per second all from the same source IP address).

Detect Abnormal Transactions using Ad-Hoc query from Athena

1. Navigate to **AWS Athena** console by using this link https://console.aws.amazon.com/athena/home?region=us-east-1

2. Make sure you select **AwsDataCatalog** as Data source and **tickettransactiondatabase** as the database, refresh to make sure the **parquet_tickettransactionstreamingdata** is showing in the table list.

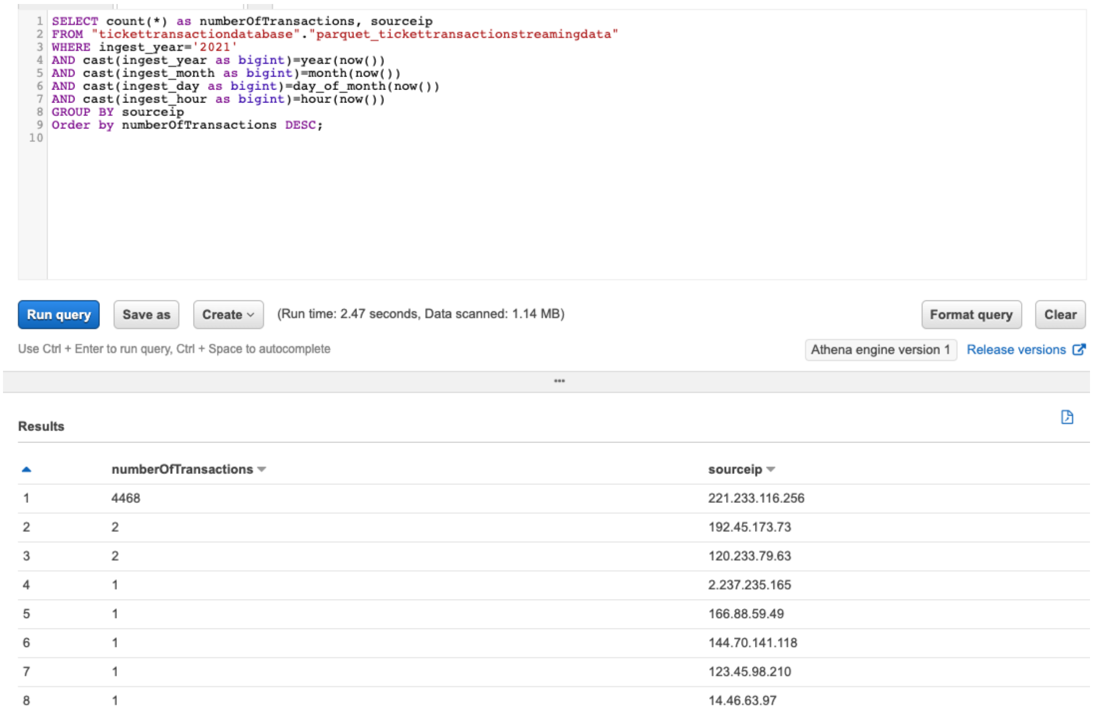


Data source
Connect data source
AwsDataCatalog
Database
tickettransactiondatabase
Filter tables and views...
Tables (1)
Create table
parquet_tickettransactionstreamingdata (Partitioned)
Views (0)
Create view

You have not created any views. To create a view, run a query and click "Create view from query"

3. Copy query as below, this is to query last hour the number of transactions by sourceip. You should see there's large number of transactions from the same sourceip.

```
SELECT count(*) as numberOfTransactions, sourceip
FROM "tickettransactiondatabase"."parquet_tickettransactionstreamingdata"
WHERE ingest_year='2021'
AND cast(ingest_year as bigint)=year(now())
AND cast(ingest_month as bigint)=month(now())
AND cast(ingest_day as bigint)=day_of_month(now())
AND cast(ingest_hour as bigint)=hour(now())
GROUP BY sourceip
Order by numberOfTransactions DESC;
```



```
SELECT count(*) as numberOfTransactions, sourceip
FROM "tickettransactiondatabase"."parquet_tickettransactionstreamingdata"
WHERE ingest_year='2021'
AND cast(ingest_year as bigint)=year(now())
AND cast(ingest_month as bigint)=month(now())
AND cast(ingest_day as bigint)=day_of_month(now())
AND cast(ingest_hour as bigint)=hour(now())
GROUP BY sourceip
Order by numberOfTransactions DESC;
```

(Run time: 2.47 seconds, Data scanned: 1.14 MB)

Use Ctrl + Enter to run query, Ctrl + Space to autocomplete

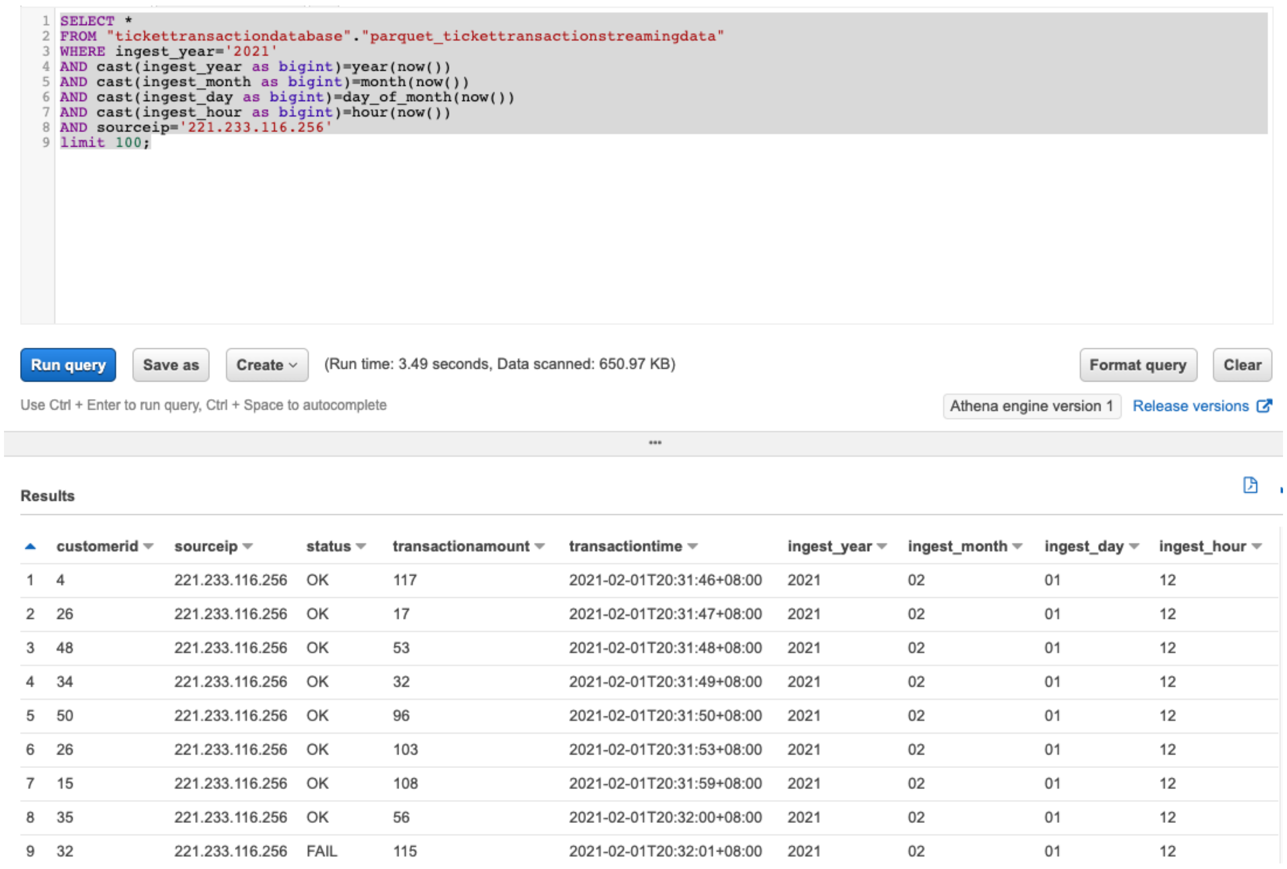
Athena engine version 1 Release versions

Results

	numberOfTransactions	sourceip
1	4468	221.233.116.256
2	2	192.45.173.73
3	2	120.233.79.63
4	1	2.237.235.165
5	1	166.88.59.49
6	1	144.70.141.118
7	1	123.45.98.210
8	1	14.46.63.97

4. Copy query as below, this is to further check if the transaction details from the same source IP. The query verified that the request is coming from same IP but with different customer id, so it's verified as abnormal transactions.

```
SELECT *
FROM "tickettransactiondatabase"."parquet_tickettransactionstreamingdata"
WHERE ingest_year='2021'
AND cast(ingest_year as bigint)=year(now())
AND cast(ingest_month as bigint)=month(now())
AND cast(ingest_day as bigint)=day_of_month(now())
AND cast(ingest_hour as bigint)=hour(now())
AND sourceip='221.233.116.256'
limit 100;
```



```
SELECT *
FROM "tickettransactiondatabase"."parquet_tickettransactionstreamingdata"
WHERE ingest_year='2021'
AND cast(ingest_year as bigint)=year(now())
AND cast(ingest_month as bigint)=month(now())
AND cast(ingest_day as bigint)=day_of_month(now())
AND cast(ingest_hour as bigint)=hour(now())
AND sourceip='221.233.116.256'
limit 100;
```

(Run time: 3.49 seconds, Data scanned: 650.97 KB)

Use Ctrl + Enter to run query, Ctrl + Space to autocomplete

Results

	customerid	sourceip	status	transactionamount	transactiontime	ingest_year	ingest_month	ingest_day	ingest_hour
1	4	221.233.116.256	OK	117	2021-02-01T20:31:46+08:00	2021	02	01	12
2	26	221.233.116.256	OK	17	2021-02-01T20:31:47+08:00	2021	02	01	12
3	48	221.233.116.256	OK	53	2021-02-01T20:31:48+08:00	2021	02	01	12
4	34	221.233.116.256	OK	32	2021-02-01T20:31:49+08:00	2021	02	01	12
5	50	221.233.116.256	OK	96	2021-02-01T20:31:50+08:00	2021	02	01	12
6	26	221.233.116.256	OK	103	2021-02-01T20:31:53+08:00	2021	02	01	12
7	15	221.233.116.256	OK	108	2021-02-01T20:31:59+08:00	2021	02	01	12
8	35	221.233.116.256	OK	56	2021-02-01T20:32:00+08:00	2021	02	01	12
9	32	221.233.116.256	FAIL	115	2021-02-01T20:32:01+08:00	2021	02	01	12