



# **Amazon Web Services Data Engineering Immersion Day**

---

Lab 1. Hydrating the Data Lake with Glue Streaming ETL  
*July 2021*

## Table of Contents

<b><i>Introduction</i></b>	<b>2</b>
<b><i>Go To Lab Environment</i></b>	<b>3</b>
<b><i>Setup Streaming Data Generator</i></b>	<b>5</b>
Create Kinesis Data Stream	5
Create Table for Kinesis Stream Source in Glue Data Catalog	6
Create and trigger Glue Stream job	8
Trigger stream data from KDG	11
Verify the Glue stream job	13
Create Glue Crawler for the transformed data	14
Trigger abnormal transaction data from KDG	18
Detect Abnormal Transactions using Ad-Hoc query from Athena	20

## 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:

Who are you?

**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 you 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 derivative 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.

✓ Accept Terms & Login

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:

## Lab 1. Hydrating the Data Lake with Glue Streaming ETL

- S3 Bucket, IAM roles etc

Modules

### Environment Setup Readme

**Outputs:**

**S3 Bucket name**  
mod-3fccddd609114925-dmslabs3bucket-1ngcgzccnd15u

**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** Data Engineering Immersion Day - Test

Team Name: \_

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 "us-east-1" as your region, unless otherwise directed by the event operator.

**Login Link**

[Open AWS Console](#) [Copy Login Link](#)

**Credentials / CLI Shell scripts**

Mac / Linux Windows

Mac or Linux

```
export AWS_DEFAULT_REGION=us-east-1
export AWS_ACCESS_KEY_ID=AKIAI44QH8D8DFK1L53K9
export AWS_SECRET_ACCESS_KEY=wJalrXU3Fgm4VbRqIzLqTvKlvu9
export AWS_SESSION_TOKEN=...
```

**How to 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 and make sure you are in the correct AWS region.
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

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.

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.

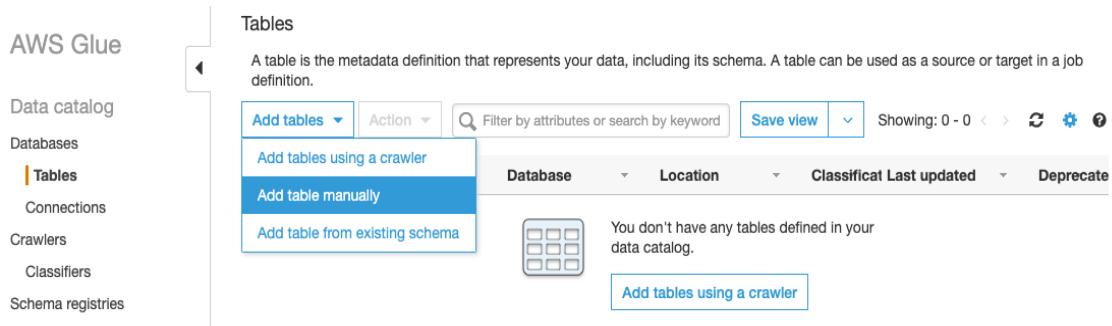
Write  
2 MiB/second, 2000 Data records/second

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 and make sure you are in the correct AWS region
2. On the AWS Glue menu, select Tables



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**

▸ Description and location (optional)

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

**Set up your table's properties**

**Table name**

**Database** ⓘ

**Add database**

▸ Description (optional)

**Next**

6. Select **Kinesis** as the source, select **Stream in my account** for Select a kinesis data stream, select the appropriate AWS region where you have created the stream, select the stream name as **TicketTransactionStreamingData** from the dropdown, and click **Next**.

## Lab 1. Hydrating the Data Lake with Glue Streaming ETL

### ✔ Table properties

Name:  
TicketTransactionStreamData  
Database:  
tickettransactiondatabase

### ○ Data store

○ Data format

○ Schema

○ Partition indices

○ Review

### Add a data store

**Select the type of source**

S3  
 Kinesis  
 Kafka

**Select a kinesis data stream**

Stream in my account  
 Stream in another account

**Region**

US East (N. Virginia) us-east-1

**Kinesis stream name**

TicketTransactionStreamingData

**Sample size (optional)**

Enter an integer between 1 and 249.

This field sets the number of files in each leaf folder to be crawled. If not set, all the files are crawled.

[Back](#) [Next](#)

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

### Choose a data format

**Classification**

CSV  
 JSON  
 ORC  
 Parquet  
 Avro  
 Grok

Choose the format of the data in your table.

[Back](#) [Next](#)

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



## Lab 1. Hydrating the Data Lake with Glue Streaming ETL

**Add table** ✕

Define a schema

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			

Back Next

9. In the Add partition indices page, without adding anything, click **Next**. 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**

Schema registries

Schemas

Settings

ETL

AWS Glue Studio

New

Workflows

**Jobs**

ML Transforms

Add job Action Filter by tags and attributes Showing: 0 - 0 < > ↻

Name	Type	ETL language	Script location	Last modified	Job bookmark
You don't have any jobs defined yet.					

Add job

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**;

## Lab 1. Hydrating the Data Lake with Glue Streaming ETL

### Configure the job properties

**Name**

**IAM role** ⓘ

mod-[redacted]-GlueLabRole-[redacted] ↕ ↻

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 ↕

- Spark
- Spark Streaming**
- Python shell

A proposed script generated by AWS Glue ⓘ

An existing script that you provide

A new script to be authored by you

**Script file name**

**S3 path where the script is stored**

 📁

**Temporary directory** ⓘ

 📁

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
<input checked="" type="radio"/> tickettransactionstreamdata	tickettransactiondatabase	TicketTransactionStreaming...	json

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

## Lab 1. Hydrating the Data Lake with Glue Streaming ETL

- ✔ 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 [Folder](#)

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-...-us-east-1
- aws-glue-temporary-466746666767-us-east-1
- mod-3...-dmslabs3bucket-1...
- ...
- ...

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

- ✔ 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**  
dmslabs3bucket-...s/TicketTransactionStream [Folder](#)

[Back](#) [Next](#)

## Lab 1. Hydrating the Data Lake with Glue Streaming ETL

- 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.

[Back](#) [Save job and edit script](#)

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

Job: TicketTransactionStreamingJob [Action](#) [Save](#) [Run job](#)

```
1 import sys
2 from awsglue.transforms import *
3 from awsglue.utils import getResolvedOptions
4 from pyspark.context import SparkContext
5 from awsglue.context import GlueContext
6 from awsglue.job import Job
7 from pyspark.sql import DataFrame, Row
8 import datetime
9 from awsglue import DynamicFrame
10
11 ## @params: [JOB_NAME]
12 args = getResolvedOptions(sys.argv, ['JOB_NAME'])
13
14 sc = SparkContext()
15 glueContext = GlueContext(sc)
16 spark = glueContext.spark_session
17 job = Job(glueContext)
18 job.init(args['JOB_NAME'], args)
19 ## @type: DataSource
```

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

[Add Job](#) [Action](#)

Name	Type	ETL language	Script location
<input checked="" type="checkbox"/> TicketT...	Spark Streaming	python	s3://aws-g...

- Run job
- Stop job run
- Choose job triggers
- Delete
- Edit job
- Edit script
- Reset job bookmark
- Create development endpoint

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.

Run job

## 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 the appropriate 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}}"
}
```

## Lab 1. Hydrating the Data Lake with Glue Streaming ETL

**Region**

**Stream/delivery stream**

**Records per second**

**Compress Records**

**Record template**

```
{
  "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** ×

---

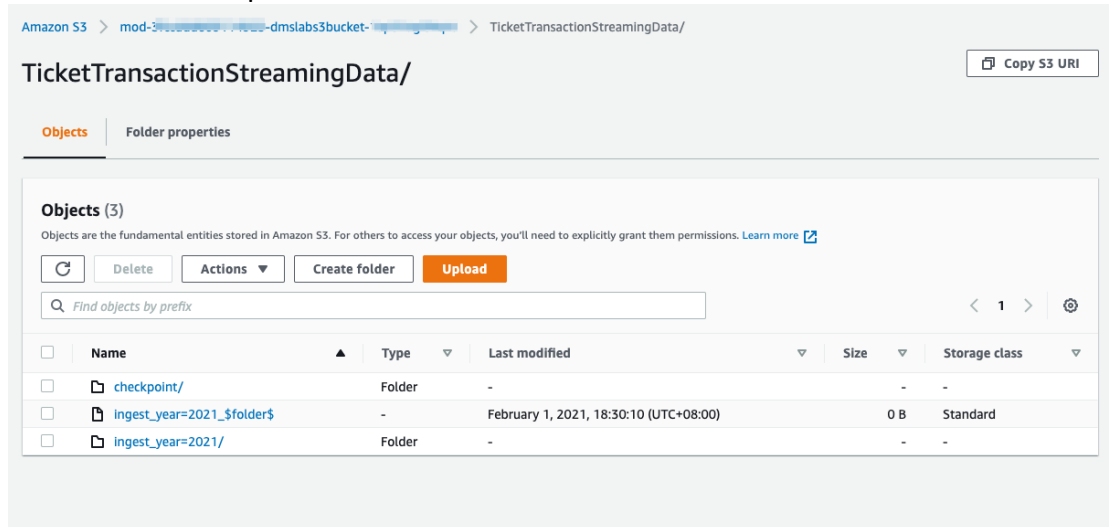
 600 records sent to Kinesis.

---

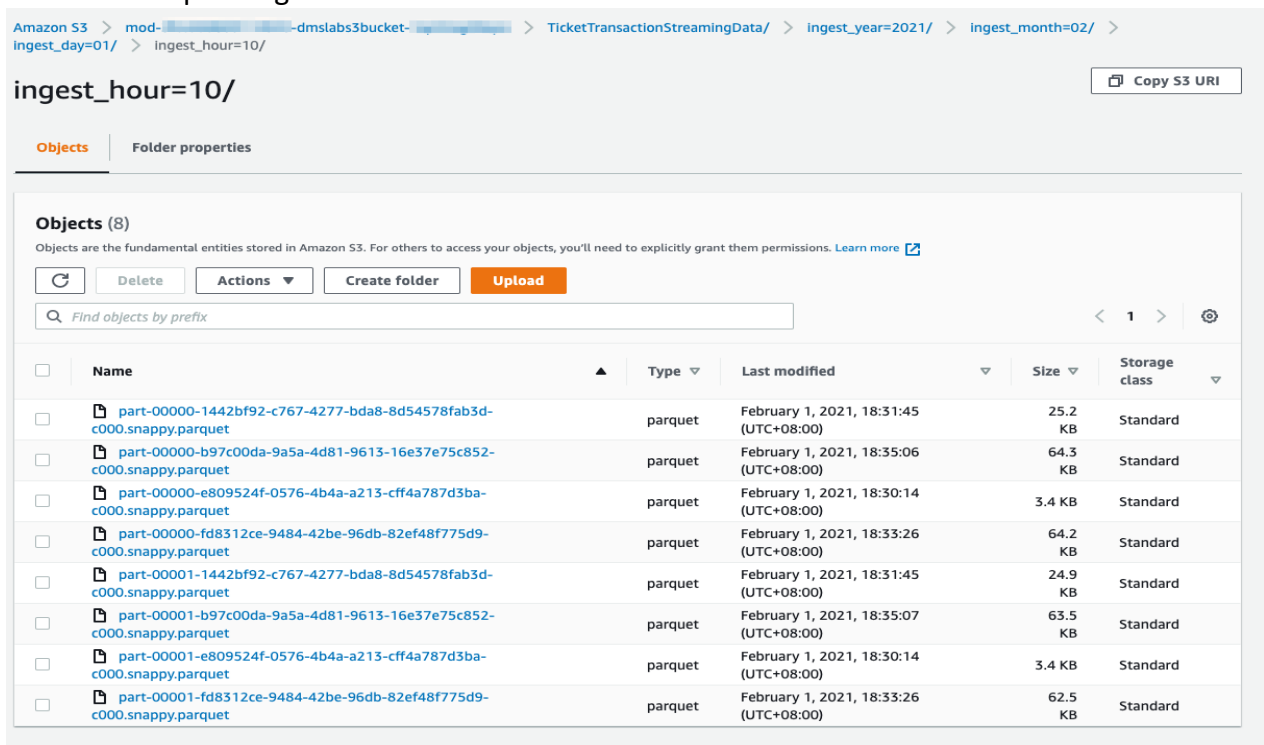
Verify the Glue stream job

## Lab 1. Hydrating the Data Lake with Glue Streaming ETL

1. Navigate to **Amazon S3** console by using this link <https://s3.console.aws.amazon.com/s3/home>
2. Navigate to the S3 bucket path we've set as Glue Stream Job target, note the folder structure of the processed data.

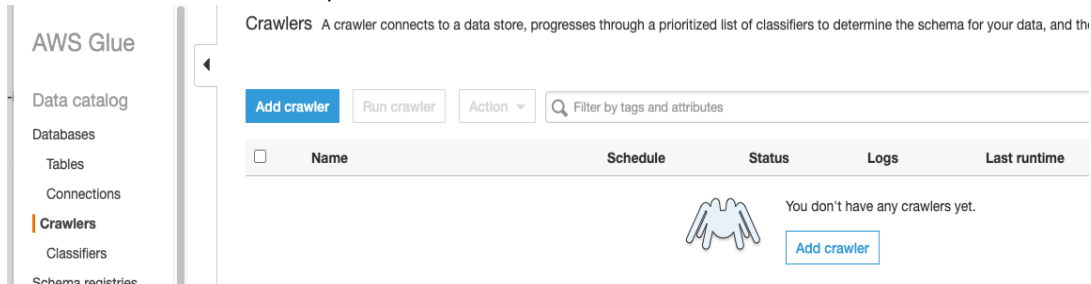


3. Check the folder content using current date and time as the folder name. Verify that the streaming data has been transformed into parquet format and persisted into corresponding folders.



## Create Glue Crawler for the transformed data

1. Navigate to [AWS Glue console](#) and make sure that you are in the correct AWS region
2. On the AWS Glue menu, select **Crawlers** and click **Add crawler**.



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

Add information about your crawler

**Crawler name**

▸ 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.

**Back**   **Next**

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



## Lab 1. Hydrating the Data Lake with Glue Streaming ETL

**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).

Add connection

**Crawl data in**

Specified path in my account

Specified path in another account

- 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

- [blurred]
- [blurred]
- [blurred]
- [blurred]
- [blurred]
- [blurred]
- mod-[blurred]-dmslabs3bucket-1
- TicketTransactionStreamingData
- mod-[blurred]
- mod-[blurred]

**Select**

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



## Lab 1. Hydrating the Data Lake with Glue Streaming ETL

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

Create a schedule for this crawler

Frequency

Start Minute

[Back](#) [Next](#)

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

[Add database](#)

Prefix added to tables (optional)

▸ 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.

[User preferences](#)

[Add crawler](#) [Run crawler](#) [Action](#)  Showing: 1 - 1

<input checked="" type="checkbox"/>	Name	Schedule	Status	Logs	Last runtime	Median runtime	Tables updated	Tables added
<input checked="" type="checkbox"/>	<a href="#">TicketTransactionParquetDataCrawler</a>	At 00 minutes past ...	Ready		0 secs	0 secs	0	0

14. When the crawler job has stopped, go to Glue Data catalog, under Tables, verify that **parquet\_tickettransactionstreamingdata** table is listed.

<input type="checkbox"/>	Name	Database	Location	Classification
<input type="checkbox"/>	<a href="#">parquet_tickettransactionstreamingdata</a>	tickettransactiondatabase	s3://mod-3fccddd609114925-d...	parquet
<input type="checkbox"/>	<a href="#">tickettransactionstreamdata</a>	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.

## Lab 1. Hydrating the Data Lake with Glue Streaming ETL

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-XXXXXXXXXXXX-dmslabs3bucket-XXXXXXXXXXXX/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

sizeKey 1121888 objectCount 28 UPDATED\_BY\_CRAWLER TicketTransactionParquetDataCrawler CrawlerSchemaSerializer/Version 1.0

recordCount 85300 averageRecordSize 20

**Table properties**

exclusions ["s3://mod-XXXXXXXXXXXX-dmslabs3bucket-XXXXXXXXXXXX/TicketTransactionStreamingData/checkpoint/\*\*"]

CrawlerSchemaDeserializer/Version 1.0 compressionType none typeOfData file

Schema Showing: 1 - 9 of 9 < >

	Column name	Data type	Partition key	Comment
1	customerid	string		
2	sourcecp	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 the appropriate 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,

## Lab 1. Hydrating the Data Lake with Glue Streaming ETL

Region

Stream/delivery stream

Records per second

Constant  Periodic

Compress Records

Record template  NormalTransaction  Template 2  Template 3  Template 4  Template 5

- for the record template, type in **AbnormalTransaction** 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}}"
}
```

## Lab 1. Hydrating the Data Lake with Glue Streaming ETL

**Region**

**Stream/delivery stream**

**Records per second**  Constant  Periodic

**Compress Records**

**Record template**  NormalTransaction  AbnormalTransaction  Template 3  Template 4  Template 5

**AbnormalTransaction**

```
{
  "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>
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.

## Lab 1. Hydrating the Data Lake with Glue Streaming ETL

**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"

- 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;
```

```
1 SELECT count(*) as numberOfTransactions, sourceip
2 FROM "tickettransactiondatabase"."parquet_tickettransactionstreamingdata"
3 WHERE ingest_year='2021'
4 AND cast(ingest_year as bigint)=year(now())
5 AND cast(ingest_month as bigint)=month(now())
6 AND cast(ingest_day as bigint)=day_of_month(now())
7 AND cast(ingest_hour as bigint)=hour(now())
8 GROUP BY sourceip
9 Order by numberOfTransactions DESC;
10
```

[Run query](#) [Save as](#) [Create](#) (Run time: 2.47 seconds, Data scanned: 1.14 MB) [Format query](#) [Clear](#)

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

## Lab 1. Hydrating the Data Lake with Glue Streaming ETL

- 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;
```

```
1 SELECT *
2 FROM "tickettransactiondatabase"."parquet_tickettransactionstreamingdata"
3 WHERE ingest_year='2021'
4 AND cast(ingest_year as bigint)=year(now())
5 AND cast(ingest_month as bigint)=month(now())
6 AND cast(ingest_day as bigint)=day_of_month(now())
7 AND cast(ingest_hour as bigint)=hour(now())
8 AND sourceip='221.233.116.256'
9 limit 100;
```

[Run query](#) [Save as](#) [Create](#) (Run time: 3.49 seconds, Data scanned: 650.97 KB) [Format query](#) [Clear](#)

Use Ctrl + Enter to run query, Ctrl + Space to autocomplete [Athena engine version 1](#) [Release versions](#)

### 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