

Lab 1: Setup Glue DataCatalog

Pre-requisites:

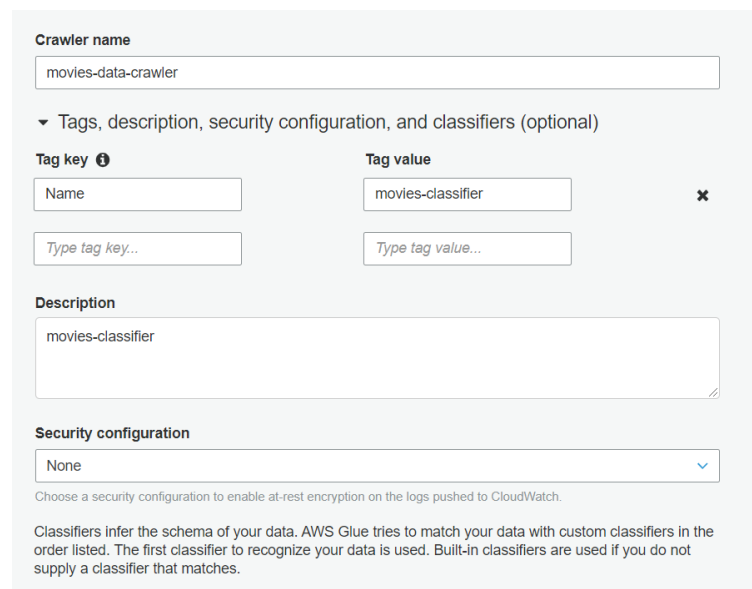
1. Data setup in S3
2. We will do this exercise in us-east-1 region.

TO DO:

1. Setup a crawler that crawls over the data in S3 which would then setup a table for us in AWS Glue Data Catalog

Steps:

1. Create a bucket **moviestabledata** in S3 and Upload **movies.json** file.
2. Navigate to Glue
3. Click on Crawlers. Click on Add Crawler.
 - a. Give Crawler a name **movies-data-crawler**. Expand Tags,Description,... and go to the bottom and add movies-classifier. Click **Next**



Crawler name

movies-data-crawler

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

Tag key ⓘ	Tag value
Name	movies-classifier
Type tag key...	Type tag value...

Description

movies-classifier

Security configuration

None

Choose a security configuration to enable at-rest encryption on the logs pushed to CloudWatch.

Classifiers infer the schema of your data. AWS Glue tries to match your data with custom classifiers in the order listed. The first classifier to recognize your data is used. Built-in classifiers are used if you do not supply a classifier that matches.

- b. Select the **Data Store** that we want this crawler to crawl. Click Next.
We have various options as indicated below. We can select JDB Connection, DynamoDB or Amazon Document DB which is a MongoDB managed service that AWS provides and we can also crawl a Mongo DB database.

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 all folders again with every subsequent crawl.
- ☐ 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

Choose a data store

S3

S3

JDBC

DynamoDB

Amazon DocumentDB

MongoDB

Crawl data in

- c. Select Data Store as S3. Select Crawl data in “Specified path in my account”.

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

Include path

s3://karthikglueinput

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.

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.

- Exclude patterns (optional)

Back

Next

- d. We can also exclude certain folders from crawling, expand “Exclude Patterns”. Just observe, we are not excluding anything for this lab exercise. Click Next.
- e. We can add more Data Stores. For this lab exercise its not needed. Select “No”. Click Next.
- f. We need to create movies-role - IAM Role. This is needed as we need to give read access to our bucket. Give the name as “Master” and Click on “Next”.

Choose an IAM role

☐ Update a policy in an IAM role
☐ Choose an existing IAM role
☒ Create an IAM role

IAM role ⓘ

AWSGlueServiceRole-

• s3://karthikglueinput

- g. Setup a schedule to specify when this crawler is going to run. We can setup hourly, weekly, monthly or choose days or you can setup a custom cron expression. For this lab exercise, select “**Run on Demand**”. And click next

Create a schedule for this crawler

Frequency

- Hourly
- Daily
- Choose days
- Weekly**
- Monthly
- Custom

- h. Setup a Database, this is where our tables will live. Click on Add Database. Give the name as movies-data-database. Click on Create. All the tables will now be created in this database within AWS Glue. Click Next.

×

Add database

Database name

movies-data-database

► Description and location (optional)

Resource link name

Enter resource link name

Shared database suggestions

Choose a database to autofill form

Shared database

Enter database to link to

Shared database owner account ID

Enter an AWS account ID

Create

- i. Review all the settings and click Finish.
 - j. Now our AWS Glue crawler *job-data-crawler* is ready.
4. Select the *movies-data-crawler* and click on Run Crawler.

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.

Add crawler

Run crawler

Action ▾

🔍

Filter by tags and attributes

Showing: 1 - 1 < > 🖨️ ⓘ

User preferences

<input type="checkbox"/>	Name	Schedule	Status	Logs	Last runtime	Median runtime	Tables updated	Tables added
<input type="checkbox"/>	movies-data-crawler		Stopping		1 min	1 min	0	1

You can see that crawler has finished running and it has added 1 table and it took runtime of 1 min.

5. Click on Databases. Select *movies-data-database* and click on it. Now, click on “Tables in movies-data-database”. Click on table **karthikglueinput** and have a look at details highlighted in yellow.

Classification	json				
Location	s3://karthikglueinput/				
Connection					
Deprecated	No				
Last updated	Fri Jul 30 14:50:42 GMT+530 2021				
Input format	org.apache.hadoop.mapred.TextInputFormat				
Output format	org.apache.hadoop.hive.qliio.HiveIgnoreKeyTextOutputFormat				
Serde serialization lib	org.openx.data.jsonserde.JsonSerDe				
Serde parameters	paths	cast,genres,title,year			
	sizeKey	10118	objectCount	1	UPDATED_BY_CRAWLER
				movies-data-crawler	CrawlerSchemaSerializerVersion
Table properties	recordCount	126	averageRecordSize	80	CrawlerSchemaDeserializerVersion
				1.0	compressionType
	typeOfData	file			

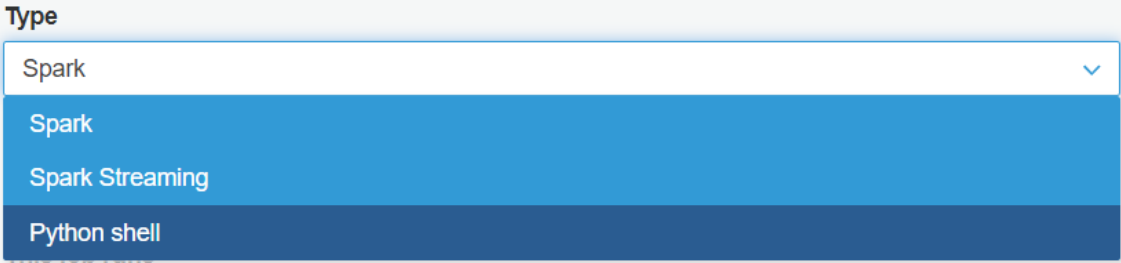
Schema

Showing: 1 - 4 of 4 < >

	Column name	Data type	Partition key	Comment
1	title	string		
2	year	int		
3	cast	array		
4	genres	array		

Lab2 : Setup Glue Job

1. Go to AWS Glue.
2. We have already setup Data Catalog from our previous lab. It is data related to movies from Wikipedia.
3. Click on the table and have a look at the schema.
4. Click on Jobs under ETL category. Click on Add Job.
5. Give the job name – *movies-data-job*
6. Select IAM Role that we created in previous lab – *AWSGlueServiceRole-MasterKarthik*
7. Select the processing environment, we can select Spark out of the below mentioned options.



Type

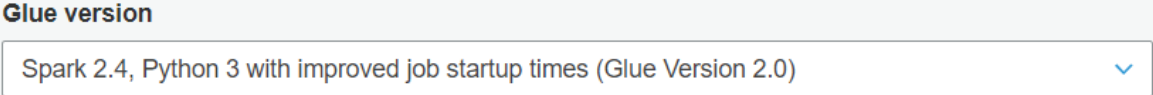
Spark

Spark

Spark Streaming

Python shell

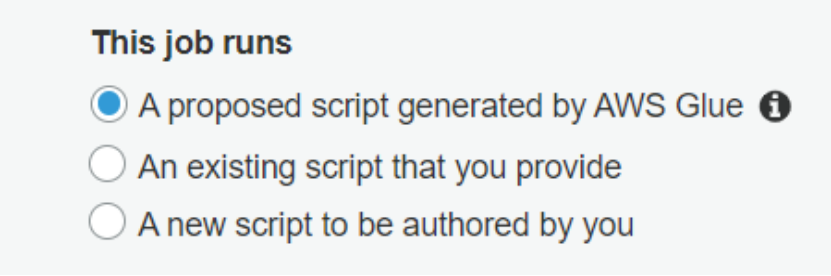
8. Select appropriate Glue Version. For this exercise, let's go ahead with below mentioned version:



Glue version

Spark 2.4, Python 3 with improved job startup times (Glue Version 2.0)

9. Select how do you want to run the job. Let's go ahead with the first option wherein Glue will construct a script for us.



This job runs

☒ A proposed script generated by AWS Glue ⓘ

☐ An existing script that you provide

☐ A new script to be authored by you

10. Examine the name of the script and where the scripts are going to get stored on S3.
11. Go to Advanced Properties, we will talk about job bookmarks later. Leave it Disabled.
12. Go to Monitoring options and we would like to monitor Job Metrics. This will load all of the job metrics into CloudWatch. Also, select continuous logging.

▼ Advanced properties

Job bookmark ⓘ

Disable

▼ Monitoring options

☐ Job metrics ⓘ

☒ Continuous logging

Log filtering ⓘ

☒ Standard filter ☐ No filter

☐ Spark UI ⓘ

13. Go to security configurations and select the worker type and number of worker nodes. Worker type can be any one of the below mentioned options. Select G.1X for our exercise.

Worker type ⓘ

G.1X (Recommended for memory-intensive jobs)

Standard

G.1X (Recommended for memory-intensive jobs)

G.2X (Recommended for jobs with ML transforms)

14. No Of workers or DPU's is 10 by default for spark jobs. Leave it to 10 for our exercise.
15. Click on Next.
16. We need to now select the data source. We have already setup the metadata catalog. Select **"moviestabledata"** and click on Next.

Job properties
movies-data-job

Data source

Transform type

Data target

Schema

Choose a data source

Filter by attributes or search by keyword

Showing: 1 - 1 < >

Name	Database	Location	Classification
vinayinputdataforglue	movies-data-database	s3://vinayinputdataforglue/	json

17. You will see that ML is not supported with Glue 2.0. If you want to use ML then you need to go ahead with Glue 1.0. Select Change Schema and click on Next. This will create a new schema for the data and load it to a target dataset.
18. We can now update our Glue Data Catalog or we can create tables in target dataset. Select "Create tables in your data target" and select as shown in below image. Ensure that you have created **"transformeddata"** folder in the same S3 bucket –. Mention **transformeddata** as the target path where data in CSV zipped format will be stored on S3.

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
 CSV

Compression type
 gzip

Connection
 - Select one -

[Add connection](#)

Target path
 s3://moviestransformeddata/transformed data

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

19. Click on Next. JSON data will now be converted into CSV format and You will now land up as shown below:

Output Schema Definition

Verify the mappings created by AWS Glue. Change mappings by choosing other columns with **Map to target**. You can **Clear** all mappings and **Reset** to default AWS Glue mappings. AWS Glue generates your script with the defined mappings.

Source

Column name	Data type	Map to target
title	string	title
year	int	year
cast	array	cast
genres	array	genres

Target

Column name	Data type			
title	string	x	↓	↑
year	int	x	↓	↑
cast	string	x	↓	↑
genres	string	x	↓	↑

[Add column](#)
[Clear](#)
[Reset](#)

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

20. If any of the columns are not important then you can drop them. Click on “Save job and edit script”.

21. You will see that AWS Glue has setup a script for us. It will be a PySpark code. This code will connect to the datasource, apply transformation and store the output in the target.



```

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
8 ## @params: [JOB_NAME]
9 args = getResolvedOptions(sys.argv, ['JOB_NAME'])
10
11 sc = SparkContext()
12 glueContext = GlueContext(sc)
13 spark = glueContext.spark_session
14 job = Job(glueContext)
15 job.init(args['JOB_NAME'], args)
16
17 ## @type: DataSource
18 ## @args: [database = "movies-data-database", table_name = "vinayinputdataforglue", transformation_ctx = "datasource0"]
19 ## @return: DataSource
20 ## @inputs: []
21
22 datasource0 = glueContext.create_dynamic_frame.from_catalog(database = "movies-data-database", table_name = "vinayinputdataforglue", transformation_ctx = "datasource0")
23
24 ## @type: ApplyMapping
25 ## @args: [mapping = [{"title", "string", "title", "string"}, {"year", "int", "year", "int"}, {"cast", "array", "cast", "string"}, {"genres", "array", "genres", "string"}]]
26 ## @return: ApplyMapping
27 ## @inputs: [frame = datasource0]
28
29 applymapping1 = ApplyMapping.apply(frame = datasource0, mappings = [{"title", "string", "title", "string"}, {"year", "int", "year", "int"}, {"cast", "array", "cast", "string"}, {"genres", "array", "genres", "string"}])
30
31 ## @type: DataSink
32 ## @args: [connection_type = "s3", connection_options = {"path": "s3://vinayinputdataforglue/transformedata", "compression": "gzip", "format": "csv", "transformation_ctx = "datasink2"}]]
33 ## @return: DataSink
34 ## @inputs: [frame = applymapping1]
35
36 datasink2 = glueContext.write_dynamic_frame.from_options(frame = applymapping1, connection_type = "s3", connection_options = {"path": "s3://vinayinputdataforglue/transformedata", "compression": "gzip", "format": "csv", "transformation_ctx = "datasink2"}])
37
38 job.commit()

```

On the left hand side, select the item and the code that does that job will be selected automatically.

22. Select “Transform” at the top-right-hand-side. It will show all the transformations that u can do.

Add transform

Name	Description
<input type="radio"/> ApplyMapping	Apply mapping to a DynamicFrame
<input type="radio"/> DropFields	Drop fields from a DynamicFrame
<input type="radio"/> DropNullFields	DynamicFrame without null fields.
<input type="radio"/> Filter	Builds a new DynamicFrame by selecting records from the input frame that satisfy the predicate function
<input type="radio"/> FindMatches	Builds a new DynamicFrame that detects records that refer to the same real-world entity based on your trained ML Transform
<input type="radio"/> Join	Join two DynamicFrames
<input type="radio"/> Map	Builds a new DynamicFrame by applying a function to all records in the input DynamicFrame
<input type="radio"/> MapToCollection	Apply a transform to each DynamicFrame in this
Create	

23. Select DropNullFields and you will see that code will be inserted into the script automatically. Get rid of that line of code. For our exercise, we don’t need.

24. Click on save and Run Job. It will allow you to override any parameters that you must have set earlier. You can do those changes and click on Run Job. For our exercise no changes are required at this step.

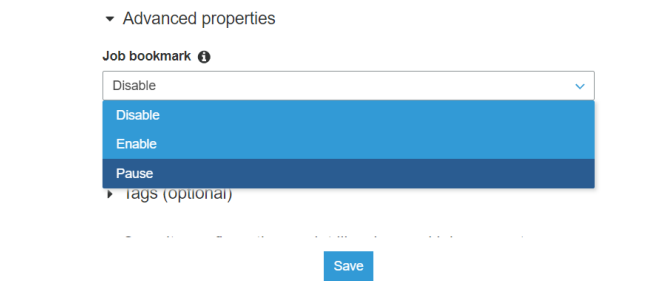
25. You can see the progress in “Continuous Logs” tab below.

26. What will happen as part of Running the job? Glue will spin up a cluster servers which will run this PySpark code against our input data, will make the transformations, it is going to compress the data and store that compressed data on S3 in our target folder.
27. Examine the target folder. You should have .gz file created. Download it. Unzip using 7-Zip and open the unzipped file using Notepad++. Observe the column headings and comma separated data. It will be a csv file.

Lab 3: Job Bookmarks (observation)

1. Go to AWS Glue -> Jobs -> select movies-data-job. Go to Actions and click on Edit Job
2. Expand Advanced Properties. You can select either Disable (by default), Enable or Pause. Select Pause and click on save.

Edit job: movies-data-job



3. In case of Pause, you need to specify from-value and to-value. To specify this, Go to Actions and click on Run Job, go to advanced properties and select pause again. Now you will be able to specify from and to values based on the run id's of jobs.