

Steps for Creating Catalog Tables:

The step-by-step process to define catalog tables using s3 as source.

Upload data to s3. We will be using JSON Data.

Create Crawler

Provide Name

Configure IAM Role

I download GitHub activity data so that we can upload to s3 to learn data engineering using AWS Analytics Services. We can download the files using instructions provided as part of gharchive.org.

```
mkdir ~/Downloads/ghactivity
cd ~/Downloads/ghactivity
wget https://data.gharchive.org/2021-01-13-{0..23}.json.gz
wget https://data.gharchive.org/2021-01-14-{0..23}.json.gz
wget https://data.gharchive.org/2021-01-15-{0..23}.json.gz
```

You can use s3 web console to create a bucket and then copy the data into the bucket. Based up on the bandwidth, this action will take a considerable amount of time.

Make sure to have a bucket by name itv-github

Make sure to create folder by name landing/ghactivity/

Create Glue Catalog Table – ghactivity:

create Glue Crawler as well as Glue Catalog Table for GitHub Activity data under itvghlandingdb.

We will give the name as GHActivity Landing Crawler.

Create a new role by name AWSGlueServiceRole-GitHub. It will create a role and attach policy to provide access to relevant s3 buckets.

Data is in s3 in this location - s3://itv-github/landing/ghactivity/

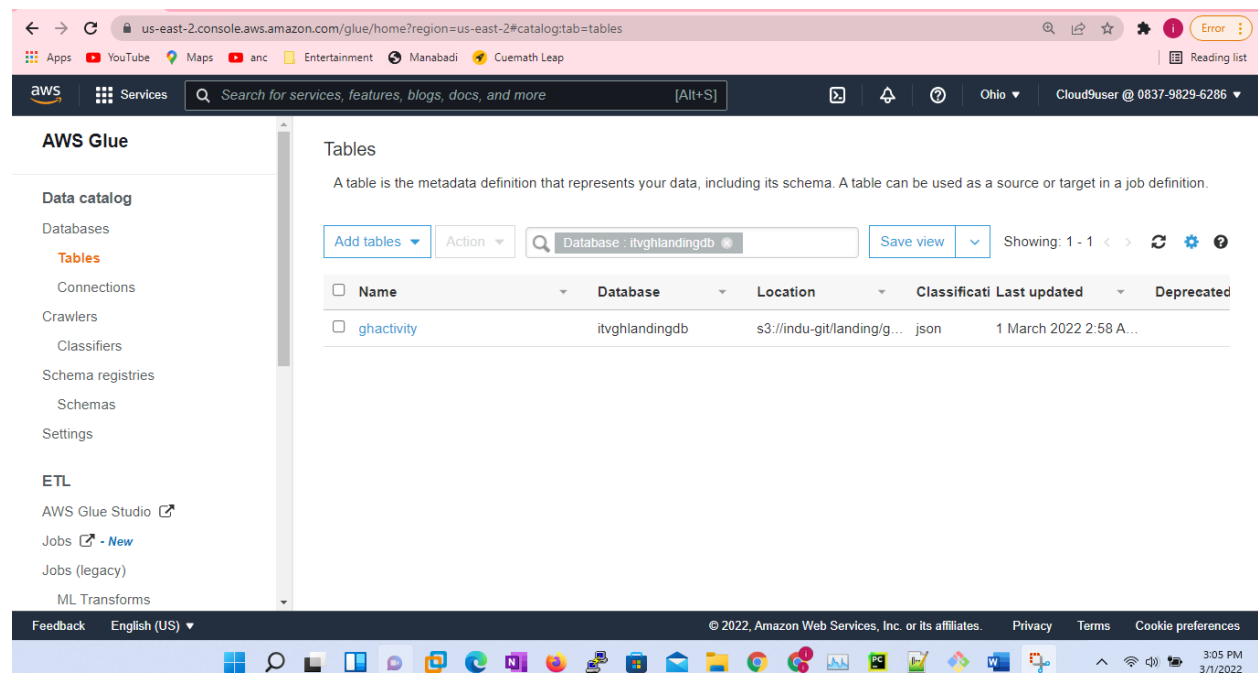
When we run crawler it will create a table with name ghactivity in itvghlandingdb.

Crawls the files and sample data from the files.

Infer schema using folder names as well as attributes from our JSON files.

Creates a table in the configured database with the schema inferred from the data and folders.

Here are the instructions to validate the table in Glue Catalog.



The screenshot shows the AWS Glue console interface. On the left is a navigation menu with options like 'Data catalog', 'Databases', 'Tables', 'Connections', 'Crawlers', 'Classifiers', 'Schema registries', 'Schemas', 'Settings', 'ETL', 'AWS Glue Studio', 'Jobs', and 'ML Transforms'. The main area displays the 'Tables' page for the 'itvghlandingdb' database. It includes a table with columns: Name, Database, Location, Classification, Last updated, and Deprecated. One table, 'ghactivity', is listed with location 's3://indu-git/landing/g...' and classification 'json'. The last updated time is '1 March 2022 2:58 A...'. The bottom of the image shows a Windows taskbar with various application icons and a system clock indicating 3:05 PM on 3/1/2022.

Name	Database	Location	Classification	Last updated	Deprecated
ghactivity	itvghlandingdb	s3://indu-git/landing/g...	json	1 March 2022 2:58 A...	

Figure 1 Created the catalog tabel

Go to the database and check if the table is created or not.

Click on the table and check the columns as well as their data types.

Once the table is created, we can run queries using services like Athena or process data using services like Glue Jobs, EMR or even third party services like Databricks.

Upload files into the folder using AWS s3 Web Console

Running Queries using Athena – ghactivity:

run following queries using Athena to ensure that data is copied and tables can be queried. Get the number of records from the table.

SELECT count(1) FROM ghactivity;

The screenshot shows the AWS Athena console interface. On the left, there's a sidebar with 'Tables (1)' and 'Views (0)'. The main area displays the query editor with the SQL query: `SELECT count(1) FROM ghactivity;`. Below the query editor, the execution status is 'Completed' with a message 'Completed'. The execution details show 'Time in queue: 0.151 sec', 'Run time: 27.167 sec', and 'Data scanned: 4.17 GB'. The results section shows a single row with the value 8237729.

#	_col0
1	8237729

Get the number of new repositories added.

SELECT count(1), count(distinct repo.id) FROM ghactivity

WHERE type = 'CreateEvent'

AND payload.ref_type = 'repository';

Crawling Multiple Folders:

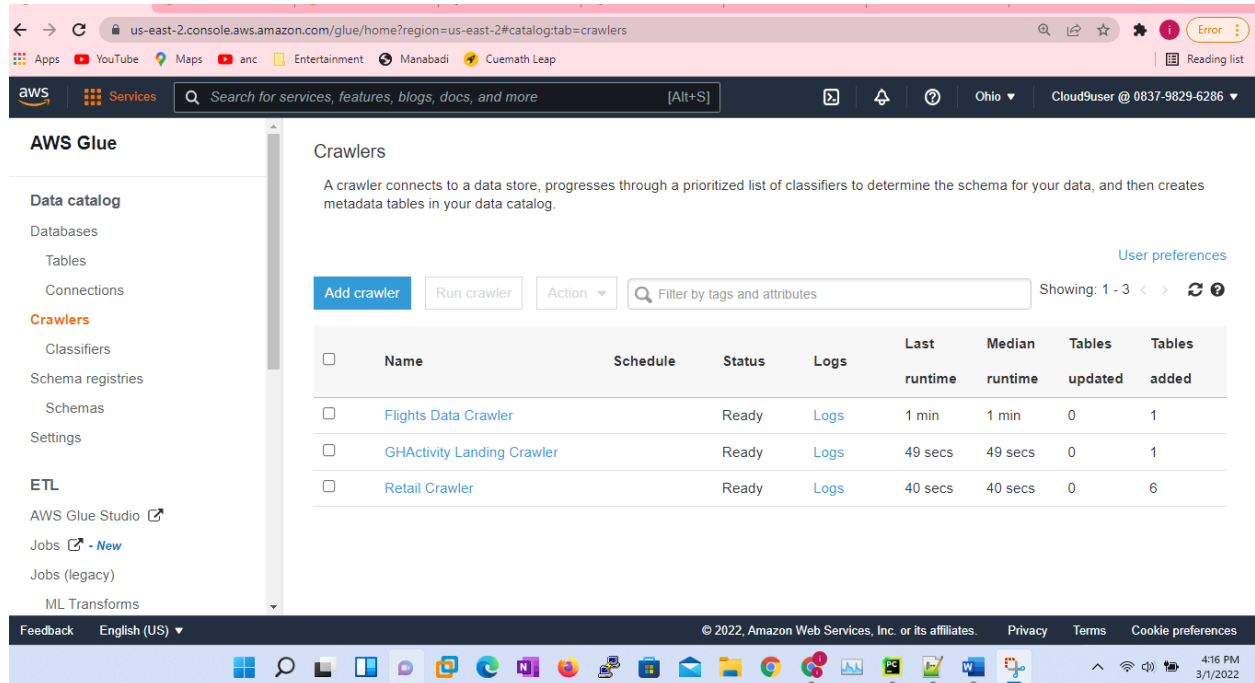
we can crawl multiple folders and create multiple tables using Glue Crawler.

We will use retail_db data using JSON format. You can get the files from our GitHub Repository.

It contains 6 folders with 1 file each.

We need to add a data source for each table as part of the crawler definition.

Once we create the crawler and run it, it will take care of creating 6 tables



The screenshot displays the AWS Glue console interface. The left-hand navigation pane is open, showing the 'Crawlers' section under the 'Data catalog' tab. The main content area, titled 'Crawlers', provides a description of the crawler's function and includes buttons for 'Add crawler', 'Run crawler', and 'Action'. A search bar and a 'Showing: 1 - 3' indicator are also present. Below this, a table lists the existing crawlers:

<input type="checkbox"/>	Name	Schedule	Status	Logs	Last runtime	Median runtime	Tables updated	Tables added
<input type="checkbox"/>	Flights Data Crawler		Ready	Logs	1 min	1 min	0	1
<input type="checkbox"/>	GHActivity Landing Crawler		Ready	Logs	49 secs	49 secs	0	1
<input type="checkbox"/>	Retail Crawler		Ready	Logs	40 secs	40 secs	0	6

Generate baseline Glue Job:

Let us generate a baseline Glue Job. We will go through all the steps that are involved in generating a baseline Glue Job.

- Give a name to the job.
- Assign appropriate role with all permissions to read and write the data.
- Configure source and target s3 locations.
- Configure locations related to logs.

AWS Glue

Data catalog

- Databases
- Tables
- Connections
- Crawlers
- Classifiers
- Schema registries
- Schemas
- Settings

ETL

- AWS Glue Studio
- Jobs [- New](#)
- Jobs (legacy)**
- ML Transforms

Jobs

A job is your business logic required to perform extract, transform and load (ETL) work. Job runs are initiated by triggers which can be scheduled or driven by events.

[Add job](#) [Action](#) Showing: 1 - 2

<input type="checkbox"/>	Name	Type	ETL language	Script location	Last modified	Job bookmark
<input type="checkbox"/>	flights_csv_to_parquet	Spark	python	s3://aws-g...	1 March 2022 10...	Disable
<input checked="" type="checkbox"/>	github_json_to_parquet	Spark	python	s3://aws-g...	1 March 2022 11...	Disable

[View run metrics](#) [Rewind job bookmark](#) Showing: 1 - 1

Start

Let us run the baseline Glue Job before customizing it. It will prove that all the permissions are working as expected.

- Monitor the job to track progress and troubleshoot if there are any errors.
- Once the job is completed, make sure that data is copied into the target location.
- We can crawl the target location to create a Glue Catalog table and query using Athena.

Make sure to delete the data in the target location as we would like to customize the job to partition the data.

Amazon S3

Buckets

- Access Points
- Object Lambda Access Points
- Multi-Region Access Points
- Batch Operations
- Access analyzer for S3

Storage Lens

- Dashboards
- AWS Organizations settings

Feature spotlight

ghactivity/

[S3 URI copied](#) [Copy S3 URI](#)

Objects (71)

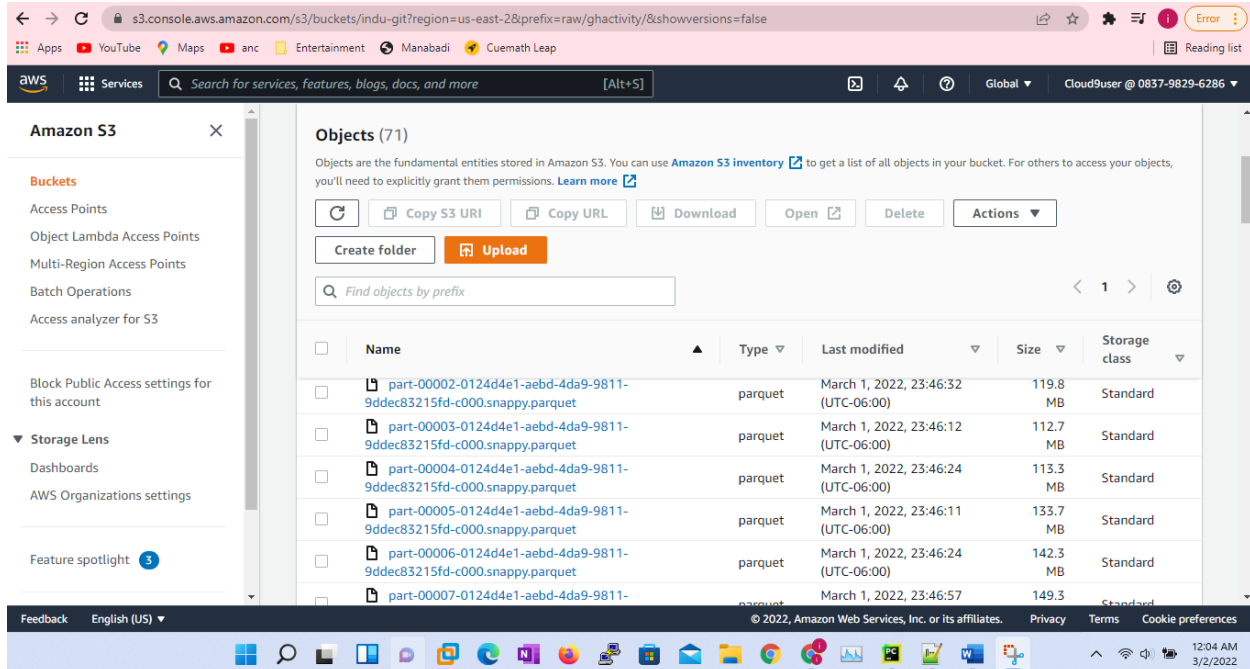
Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

[Copy S3 URI](#) [Copy URL](#) [Download](#) [Open](#) [Delete](#) [Actions](#)

[Create folder](#) [Upload](#)

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	part-00000-0124d4e1-aebd-4da9-9811-9ddec83215fd-c000.snappy.parquet	parquet	March 1, 2022, 23:45:46 (UTC-06:00)	78.8 MB	Standard

Figure 2 Data is copied into the target location.



Glue Script for Partitioning Data:

```
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from pyspark.sql.functions import date_format, substring
from awsglue.context import GlueContext
from awsglue.job import Job
from awsglue.dynamicframe 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)

datasource0 = glueContext.\
    create_dynamic_frame.\
    from_catalog(\
        database = "itvghlandingdb",\
        table_name = "ghactivity",\
        transformation_ctx = "datasource0"\
    )

df = datasource0.\
    toDF().\
    withColumn('year', date_format(substring('created_at', 1, 10), 'yyyy')).\
    withColumn('month', date_format(substring('created_at', 1, 10), 'MM')).\
    withColumn('day', date_format(substring('created_at', 1, 10), 'dd'))

dyf = DynamicFrame.fromDF(dataframe=df, glue_ctx=glueContext, name="dyf")

datasink4 = glueContext.\
    write_dynamic_frame.\
```

```
from_options(frame=dyf,  
             connection_type="s3",  
             connection_options={"path": "s3://itv-github/raw/ghactivity/",  
                                "compression": "snappy",  
                                "partitionKeys": ["year", "month", "day"]},  
             format="glueparquet",  
             transformation_ctx="datasink4")
```

```
job.commit()
```