**[Instructions] Prerequisites for Glue Catalog Tables**

Let us go through the prerequisites for Glue Crawler to crawl the metadata and create Glue Catalog Tables..

* We need to have data in s3 or other supported data stores to use Glue Crawler to crawl the metadata to create tables.
* It is highly recommended to have structure to the data. If we are reading the data from text files, then the data should have a header.
* Glue should have appropriate permissions via IAM Role to access the s3 buckets.

**[Instructions] Steps for Creating Catalog Tables**

Let us go through 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
  + Configure Source

**[Instructions and Code] Download Data Set**

Let us download GitHub activity data so that we can upload to s3 to learn data engineering using AWS Analytics Services.

* You can use wget to download the data into a local machine.
* We will download 3 day’s worth of data. Here are the dates.
  + 2020-01-13
  + 2020-01-14
  + 2020-01-15
* We can download the files using instructions provided as part of [gharchive.org](https://www.google.com/url?q=https://www.gharchive.org&sa=D&source=editors&ust=1629529826548000&usg=AOvVaw2bRNpv3PNFcTv4-ZhEY5t8). I will be downloading these files using terminal on my Mac under ~/Downloads/ghactivity

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

* You can then use the s3 web console or upload using AWS CLI to upload the data.

**[Instructions and Code] Upload data to s3**

Let us upload GitHub activity data into s3. 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/**
* Upload files into the folder using AWS s3 Web Console

You can also use the below commands to upload the files using CLI. You need to have AWS CLI installed and configured against your AWS account with credentials which have write permissions to s3 bucket. I am using credentials which have write permissions on s3 bucket **itv-github**.

1. cd ~/Downloads/ghactivity
3. # Single thread
4. aws s3 cp . s3://itv-github/landing/ghactivity/ \
5. --recursive \
6. --profile itvgithub
8. # 3 threads in parallel using nohup
9. nohup aws s3 cp . s3://itv-github/landing/ghactivity/ \
10. --exclude "\*" \
11. --include "2021-01-13\*" \
12. --recursive &
13. nohup aws s3 cp . s3://itv-github/landing/ghactivity/ \
14. --exclude "\*" \
15. --include "2021-01-14\*" \
16. --recursive &
17. nohup aws s3 cp . s3://itv-github/landing/ghactivity/ \
18. --exclude "\*" \
19. --include "2021-01-15\*" \
20. --recursive &

**[Instructions] Create Glue Catalog Database - itvghlandingdb**

Let us create a Glue Catalog Database by name **itvghlandingdb**.

* Login into Glue Web Console.
* Under Glue Catalog Click on Databases.
* Click on Create Database and give the name **itvghlandingdb**.

**[Instructions] Create Glue Catalog Table - ghactivity**

Let us 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.
  + 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.

**[Instructions and Code] Running Queries using Athena - ghactivity**

Let us 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;

* Get the number of new repositories added.

1. SELECT count(1), count(distinct repo.id) FROM ghactivity
2. WHERE type = 'CreateEvent'
3. AND payload.ref\_type = 'repository';

* Preview repo related details using repo column of type struct.

1. SELECT repo.\* FROM ghactivity
2. WHERE type = 'CreateEvent'
3. AND payload.ref\_type = 'repository'
4. LIMIT 10;

* Get the number of repositories created for each of the 3 days.

1. SELECT substr(created\_at, 1, 10),
2. count(1),
3. count(distinct id)
4. FROM ghactivityWHERE type = 'CreateEvent'AND payload.ref\_type = 'repository'
5. GROUP BY substr(created\_at, 1, 10);

**[Instructions] Crawling Multiple Folders**

Let us see how 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](https://github.com/itversity/retail_db_json).
* 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.

**[Instructions and Code] Managing Glue Catalog using AWS CLI**

We can manage Glue Catalog using AWS Glue CLI. Let us understand how to trigger the crawler and also how to validate whether the tables are created or not using AWS Glue CLI commands.

* We need to make sure that the user has required permissions to manage Glue Catalog via CLI.
* We can assign Glue Service Role to the user whose credentials are configured as part of AWS CLI Profile.
* Here are the commands to get the details about a crawler.

1. aws glue list-crawlers --profile itvgithub
2. aws glue get-crawler --name "Retail Crawler" --profile itvgithub
3. aws glue start-crawler --name "Retail Crawler" --profile itvgithub

* Here are the commands to list the tables.

1. aws glue get-databases --profile itvgithub
2. aws glue get-tables --database-name retail\_db --profile itvgithub
3. aws glue get-table \
4. --database-name retail\_db \
5. --name orders \
6. --profile itvgithub

**[Instructions and Code] Managing Glue Catalog using Python Boto3**

We can manage Glue Catalog using Python boto3. Let us understand how to trigger the crawler and also how to validate whether the tables are created or not using AWS boto3 code.

* We need to make sure that the user has required permissions to manage Glue Catalog via CLI.
* We can assign Glue Service Role to the user whose credentials are configured as part of AWS CLI Profile.
* Here is the code to get the details about crawler.

1. import boto3
2. import os
3. os.environ.setdefault('AWS\_PROFILE', 'itvgithub')
4. os.environ.setdefault('AWS\_DEFAULT\_REGION', 'us-east-1')
6. glue\_client = boto3.client('glue')
7. crawler = glue\_client.list\_crawlers()['CrawlerNames'][3]
9. glue\_client.get\_crawler(
10. Name=crawler
11. )
13. glue\_client.start\_crawler(
14. Name=crawler
15. )
17. glue\_client.get\_crawler(
18. Name=crawler
19. )['Crawler']['State']

* Here is the code to get the details about databases.

1. import boto3
2. import os
3. os.environ.setdefault('AWS\_PROFILE', 'itvgithub')
4. os.environ.setdefault('AWS\_DEFAULT\_REGION', 'us-east-1')
6. glue\_client = boto3.client('glue')
8. for db in glue\_client.get\_databases()['DatabaseList']:
9. print(db['Name'])

* Here is the code to get the details about all the tables as well as a single table.

1. import boto3
2. import os
3. os.environ.setdefault('AWS\_PROFILE', 'itvgithub')
4. os.environ.setdefault('AWS\_DEFAULT\_REGION', 'us-east-1')
6. glue\_client = boto3.client('glue')
8. for table in glue\_client.get\_tables(
9. DatabaseName='retail\_db'
10. )['TableList']:
11. print(table['Name'])
13. glue\_client.get\_table(
14. DatabaseName='retail\_db',
15. Name='orders'
16. )