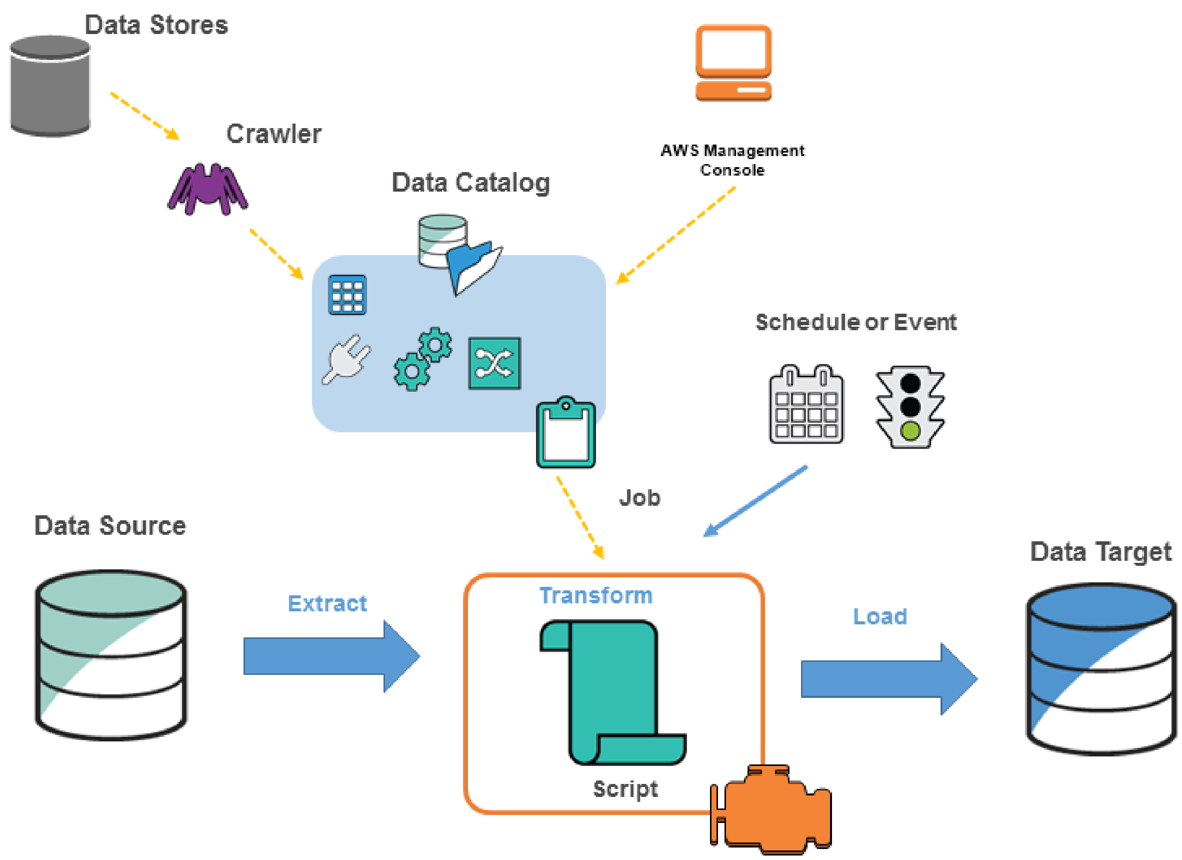
#13 Task: Walkthrough on an ETL (Extract, Load & Transform) Job using PySpark framework on AWS Glue with S3 Bucket, Glue Crawlers, and required IAM Roles. Also, Encrypt the data of the S3 bucket using a KMS CMK (Customer Managed Key) with a strong Cryptographic Algorithm.



1. Set S3 Buckets for source and destination data.

Glue can read data from a database or S3 bucket. For example, I have created an S3 bucket called glue-bucket-cloudspikes. Create two folders from the S3 console and name them to read and write. Now create a text file with the following data and upload it to the read folder of the S3 bucket.

```
rank,movie_title,year,rating
1,The Shawshank Redemption,1994,9.2
2,The Godfather,1972,9.2
3,The Godfather: Part II,1974,9.0
4,The Dark Knight,2008,9.0
5,12 Angry Men,1957,8.9
6,Schindler’s List,1993,8.9
7,The Lord of the Rings: The Return of the King,2003,8.9
8,Pulp Fiction,1994,8.9
9,The Lord of the Rings: The Fellowship of the Ring,2001,8.8
10,Fight Club,1999,8.8
```

2. Crawl the data source to the data catalog.

In this step, we will create a crawler. The crawler will catalog all files in the specified S3 bucket and prefix. All the files should have the same schema. In Glue crawler terminology the file format is known as a classifier. The crawler identifies the most common classifiers automatically including CSV, JSON, and parquet. Our sample file is in CSV format and will be recognized automatically.

- In the left panel of the Glue management console click Crawlers.
- Click the blue Add crawler button.
- Give the crawler a name such as glue-demo-cloudspikes-crawler.
- In Add a data store menu choose S3 and select the bucket you created. Drill down to select the read folder.
- In Choose an IAM role create a new one. Name the role for example glue-demo-cloudspikes-iam-role.
- In Configure the crawler’s output add a database called glue-demo-cloudspikes-db.
- When you are back in the list of all crawlers, tick the crawler that you created. Click **Run** Crawler.

3. Query the captured data from S3 Bucket on Athena.

Go to the Database created and click on Table data and hit proceed to navigate to the Athena Query editor tool.

First, go to the Settings tab and set the Query result location as the destination S3 bucket with write specific folder. Then go back to the Editor tab in Athena and hit on the Run button.

This will execute the Athena query which looks like a SQL query to fetch the data from the source location which is Glue Database Table. This will also generate the data in the Query result location S3 bucket that you selected before as per the format you chose (JSON, CSV, etc).

4. Setup a Glue Job to perform ETL Operations.

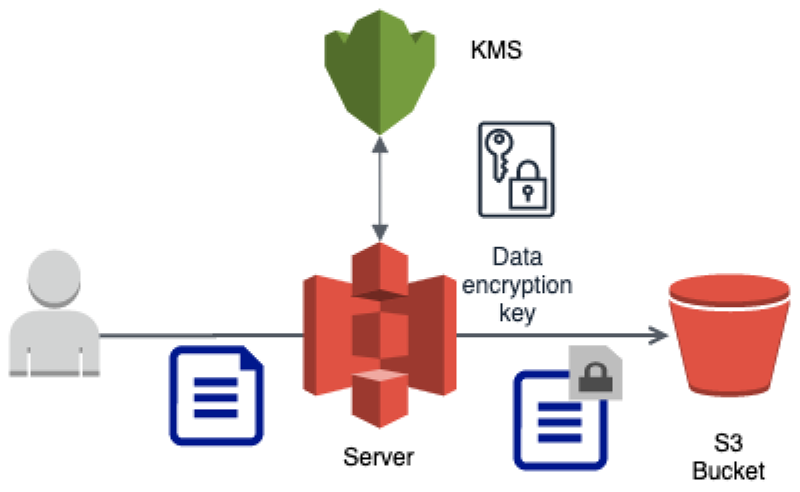
Go back to the Glue console and hit on Create button under the ETL Jobs → Visual ETL section. Name your job and configure the Data source and Data target S3 buckets as the same bucket but at different locations by clicking on the Visual canvas. For Data source make sure you select the read folder and for Data target select the write folder under glue-bucket-cloudspikes S3 Bucket.

Now, go to the Job details tab and select IAM Role as an existing role or create a new role that has full access to all the services used in this lab work. Go down and open the **Advanced Properties** section and name the Script filename as per your choice.

Once everything is done, you can hit on Run button by selecting the specific newly created Job. On the left-hand tab, you can navigate to the Job run monitoring tab and have a look at the Job status along with the CloudWatch Log Group screen that has Glue Job execution logs.

If the Glue job is successfully executed, you can see the generated data on the target S3 bucket under the write folder as per the configurations we set.

1. Encrypt data on S3 Bucket via Permission Policy.



In order to enforce object encryption, create an S3 bucket policy that denies any S3 Put request that does not include the x-amz-server-side-encryption header. There are two possible values for the x-amz-server-side-encryption header: AES256, which tells S3 to use S3-managed keys, and aws:kms, which tells S3 to use AWS KMS–managed keys. Bucket Policy will look like this

```
{
    "Version": "2012-10-17",
    "Id": "PutObjPolicy",
    "Statement": [
        {
            "Sid": "DenyIncorrectEncryptionHeader",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::<bucket_name>/*",
            "Condition": {
                  "StringNotEquals": {
                       "s3:x-amz-server-side-encryption": "AES256"
                  }
            }
        },
        {
            "Sid": "DenyUnEncryptedObjectUploads",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::<bucket_name>/*",
            "Condition": {
                "Null": {
                    "s3:x-amz-server-side-encryption": true
                }
            }
        }
    ]
}
```

Now if we try to upload the object to S3 bucket without encryption it should fail:

```
$ aws s3 cp testingbucketencryption s3://mytestbuclet-198232055
upload failed: ./testingbucketencryption to
s3://mytestbuclet-198232055/testingbucketencryption An error occurred
(AccessDenied) when calling the PutObject operation: Access Denied
```

Let try it with encryption enabled:

```
$ aws s3 cp testingbucketencryption s3://mytestbuclet-198232055 --sse AES256
upload: ./testingbucketencryption to
s3://mytestbuclet-198232055/testingbucketencryption
```