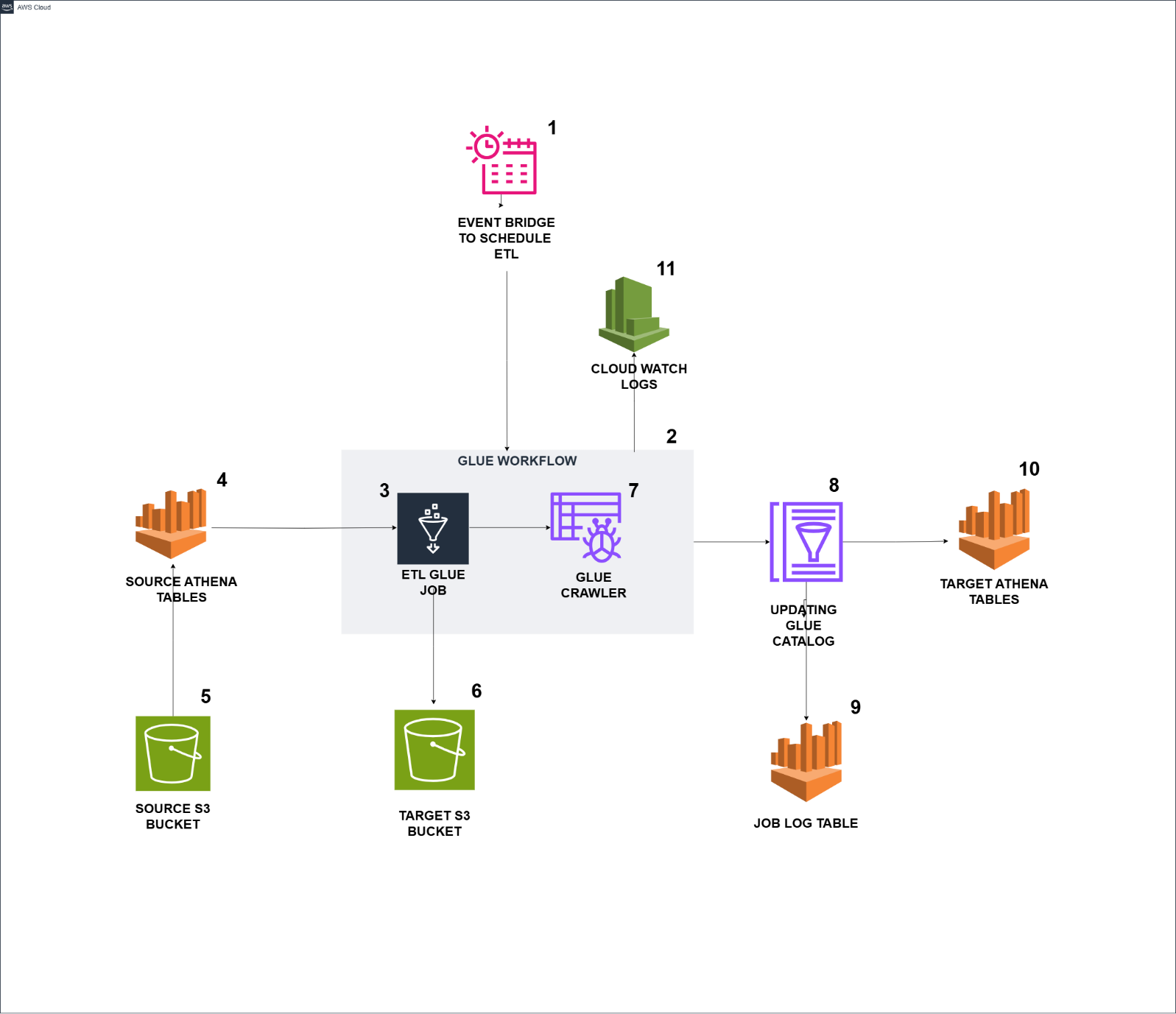
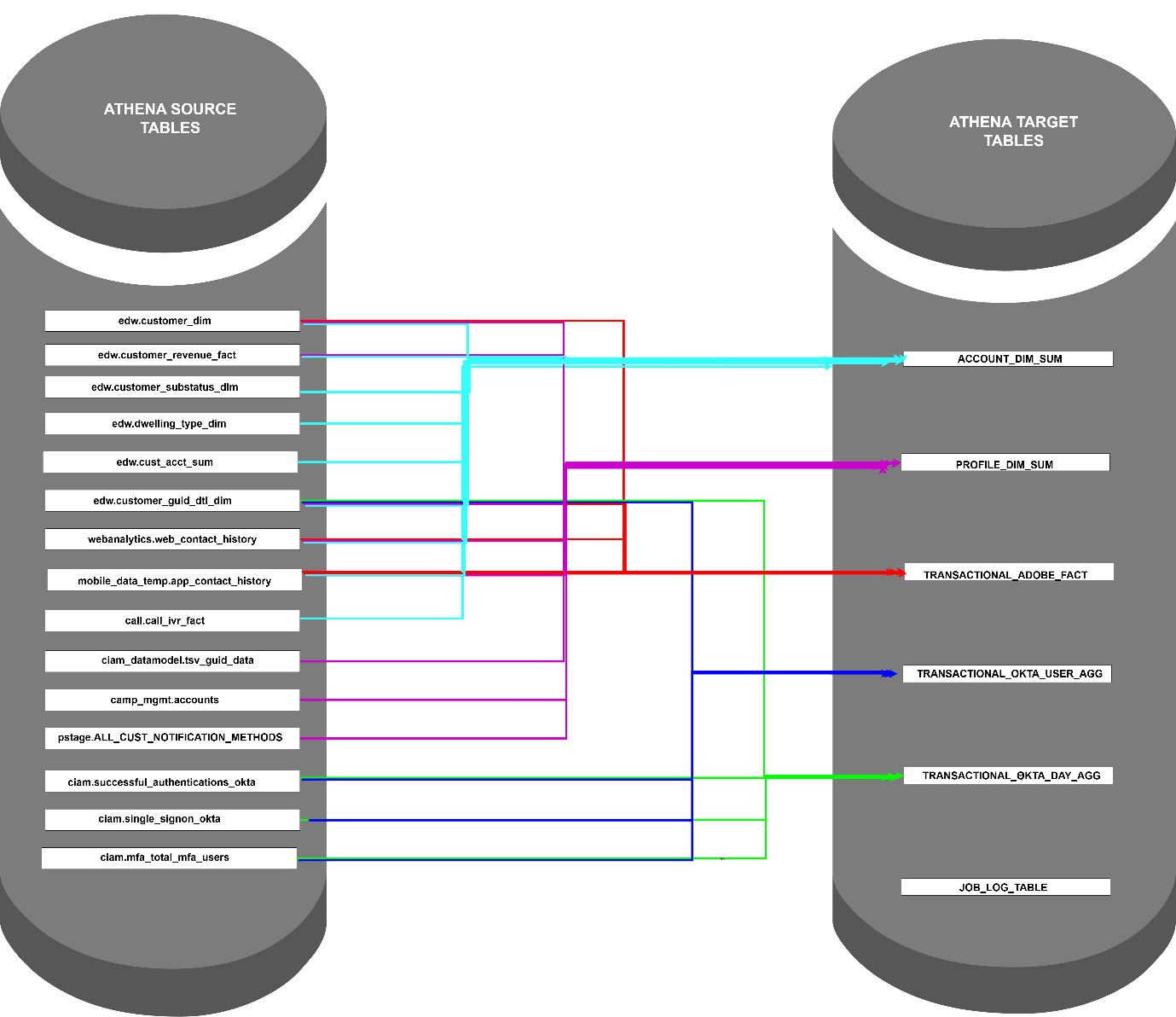
**ARICHITECTURE DIAGRAM FOR GLUE:**



**DESCRIPTION:**

1. EventBridge triggers the Glue workflow based on the scheduled time.
2. The Glue workflow is triggered by EventBridge and initiates the run.
3. A Glue job starts running within the Glue workflow using an On-Demand trigger.
4. The Glue job executes Athena queries to read data from the source tables.
5. The Glue job fetches data from the source S3 path based on the queries executed in Athena.
6. The data from the source tables is loaded into the target S3 path as Parquet files, organized into partitioned folders.
7. Upon successful completion of the Glue job, the Glue crawler is triggered and starts running.
8. The job log table is created, containing runtime information and table counts.
9. The target tables are available in Athena, where the data can be queried.
10. CloudWatch stores logs for both the Glue jobs and the Glue crawler.

**DATA FLOW DIAGRAM :**



**DESCRIPTION :**

**Source Databases :** edw, webanalytics, mobile\_data\_temp,call,ciam, camp\_mgmt, pstage, ciam\_datamodel

**Source Tables :** edw.customer\_dim, edw.customer\_revenue\_fact, edw.customer\_substatus\_dim,edw.dwelling\_type\_dim, edw.cust\_acct\_sum, edw.customer\_guid\_dtl\_dim, webanalytics. web\_contact\_history, mobile\_data\_temp.app\_contact\_history, call.call\_ivr\_fact, ciam\_datamodel.tsv\_guid\_data, camp\_mgmt.accounts, pstage.ALL\_CUST\_NOTIFICATION\_METHODS, ciam.successful\_authentications\_okta, ciam.single\_signon\_okta, ciam.mfa\_total\_mfa\_users

**Target Database :** ciam\_datamodel

**Target Tables :** account\_dim\_sum, profile\_dim\_sum, transaction\_adobe\_fact, transaction\_okta\_use\_agg, transaction\_okta\_day\_agg, job\_log\_table

**GLUE JOB FUNCTIONALITIES:**

* When the job starts running, it checks if there is a JOB\_LOG\_TABLE in the target database.
* If the JOB\_LOG\_TABLE is not present or if it does not have an entry for 13\_MONTHS\_LOAD, the job will directly read the last 13 months of data from multiple source tables and load it into the target S3 path in Parquet file format. The data will be partitioned by a specific column for each target table.
* If the JOB\_LOG\_TABLE is present, the job will check the LOAD\_TYPE column. If the LOAD\_TYPE column contains an entry for 13\_MONTHS\_LOAD, the job will read only the latest current month’s data from the source tables and overwrite the corresponding partitioned folder in S3.
* Each time the job runs, it will create an entry in the JOB\_LOG\_TABLE that includes the table counts, start time, end time, and runtime details of the Glue job.
* After loading the data into the target S3 path, the Glue job will check if there is more than 14 months' worth of data. If so, it will delete the oldest month's data. Otherwise, it will skip the deletion.
* A Glue crawler has been created and configured with all the target S3 paths and the target database name.
* After the successful completion of the Glue job, the crawler will be triggered once to populate the tables in Athena.
* **ATHENA QUERIES :**

**PYSPARK SCRIPT : **

**GLUE JOB CONFIGURATION :**

Job Type: Spark

Glue Version: Glue 4.0

Language: Python 3

Worker Type: G.2X

Maximum No. of Workers: 10

Job Bookmark: enable

Job metrics: enable

Job observability metrics: enable

Continuous logging: enable

Maximum Concurrency : 1

**IAM ROLES & PERMISSIONS:**

* S3 read & write permission required
* Athena read permission required
* Cloud watch permission required to check the logs

**GLUE JOB RUNTIME** :

*For 13 months - 1 hr 58 mins*

*For latest current months - 42 mins*

**COST PRICING:**

**GLUE JOB costs:** 0.44$ per DPU per hour

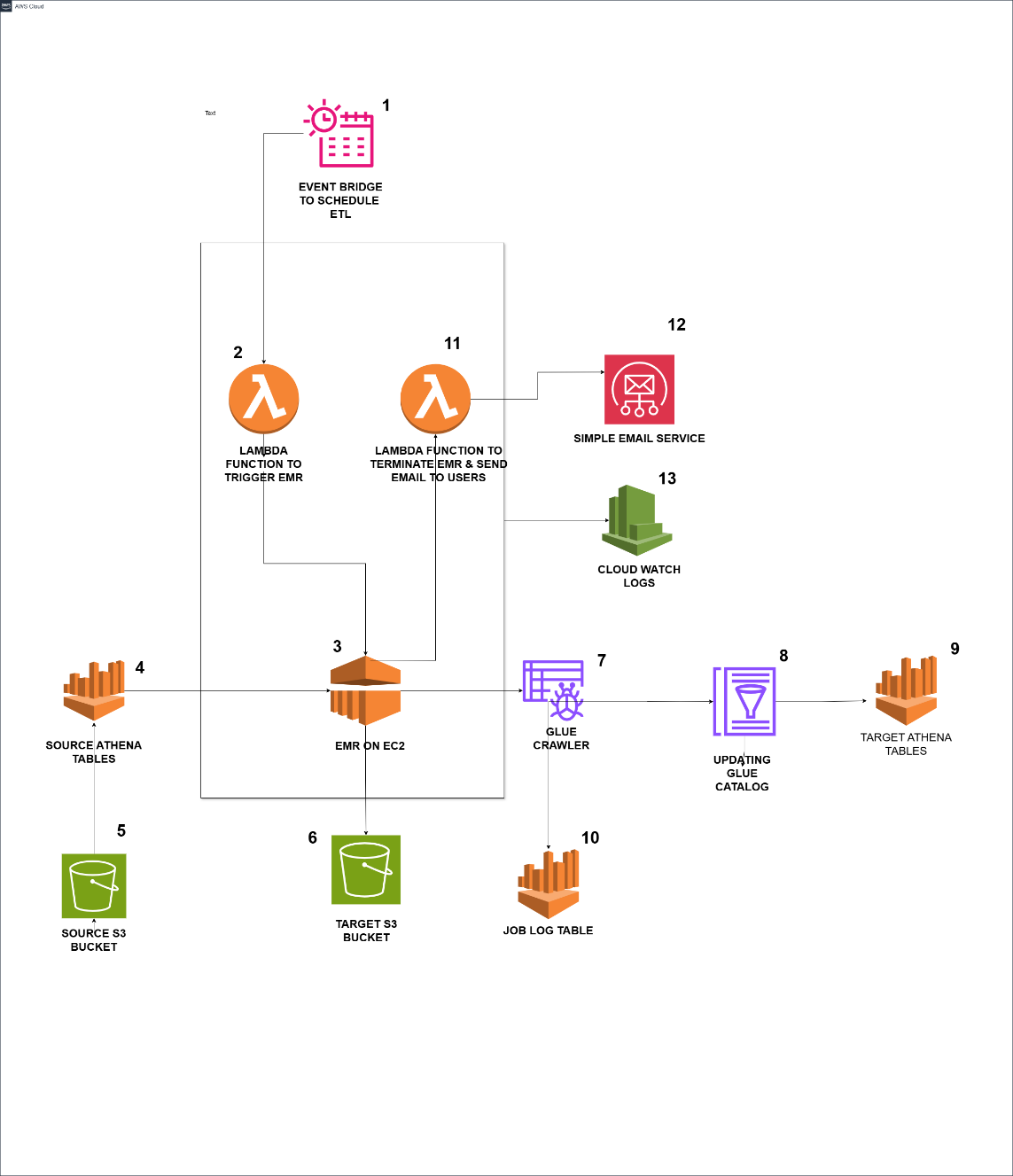
**EXISTING GLUE JOB CONFIGURATION**: G.8X 100 worker

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Usage Type** | **Run Time** | **Days** | **Calculation** | **Cost** |
| Development Phase | Approx 6 hours a day | 3 weeks | 800 DPU \* 0.44 \* 6 hour = $2112 per day \* 21 days = $1222.2 USD | **$44352 USD** |
| Testing Phase | Approx 6 hours a day | 1 week | 800 DPU \* 0.44 \* 6 hour = $2112 per day \* 7 days = $407.4 USD | **$14784 USD** |
| Production | Approx 1 hour a day | Per Month | 800 DPU \* $0.44 \* 1 hour = $352 per day \* 30 days = $264 USD | **$10560 USD** |

**RECOMMENDED GLUE JOB CONFIGURATION**: G.2X 10 worker

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Usage Type** | **Run Time** | **Days** | **Calculation** | **Cost** |
| Development Phase | Approx 6 hours a day | 3 weeks | 20 DPU \* 0.44 \* 6 hour = $58.2 per day \* 21 days = $1222.2 USD | **$1222.2 USD** |
| Testing Phase | Approx 6 hours a day | 1 week | 20 DPU \* 0.44 \* 6 hour = $58.2 per day \* 7 days = $407.4 USD | **$407.4 USD** |
| Production | Approx 1 hour a day | Per Month | 20 DPU \* $0.44 \* 1 hour = $8.8 per day \* 30 days = $264 USD | **$264 USD** |

**ARCHITECTURE DIAGRAM FOR EMR ON EC2:**



**DESCRIPTION:**

1. Based on the specified schedule, EventBridge triggers the Lambda Function, initiating the process.
2. The Lambda Function then triggers the EMR cluster, which starts running.
3. The EMR cluster launches the required EC2 instances and starts processing.
4. PySpark script is executed on the EMR cluster, which reads data from the Athena tables.
5. The PySpark script fetches data from the source S3 path using Athena queries.
6. The data retrieved from Athena is loaded into the target S3 path as parquet files, organized in partitioned folders.
7. Once the PySpark script completes execution on EMR, a Glue Crawler is triggered to run.
8. The Glue Crawler populates the tables in the AWS Glue Data Catalog.
9. The newly populated target tables in Glue can now be queried using Athena.
10. The PySpark script on EMR generates a Job Log Table containing runtime information and table counts.
11. The Lambda Function terminates the EC2 instances & EMR cluster after successful execution of the PySpark script.
12. Through SES, Lambda Function sends an email to the users, notifying them of the job status and providing the job log table details.
13. All logs for the Lambda Function and EMR are stored in CloudWatch for monitoring and troubleshooting.

**EMR ON EC2 INSTANCE:**

**RECOMMENDED CONFIGURATION:**

EC2 instance Type**:** m5.xlarge = $0.192 per hour

Master Node: 1 \* m5.xlarge

Core Node: 3 \* m5.xlarge

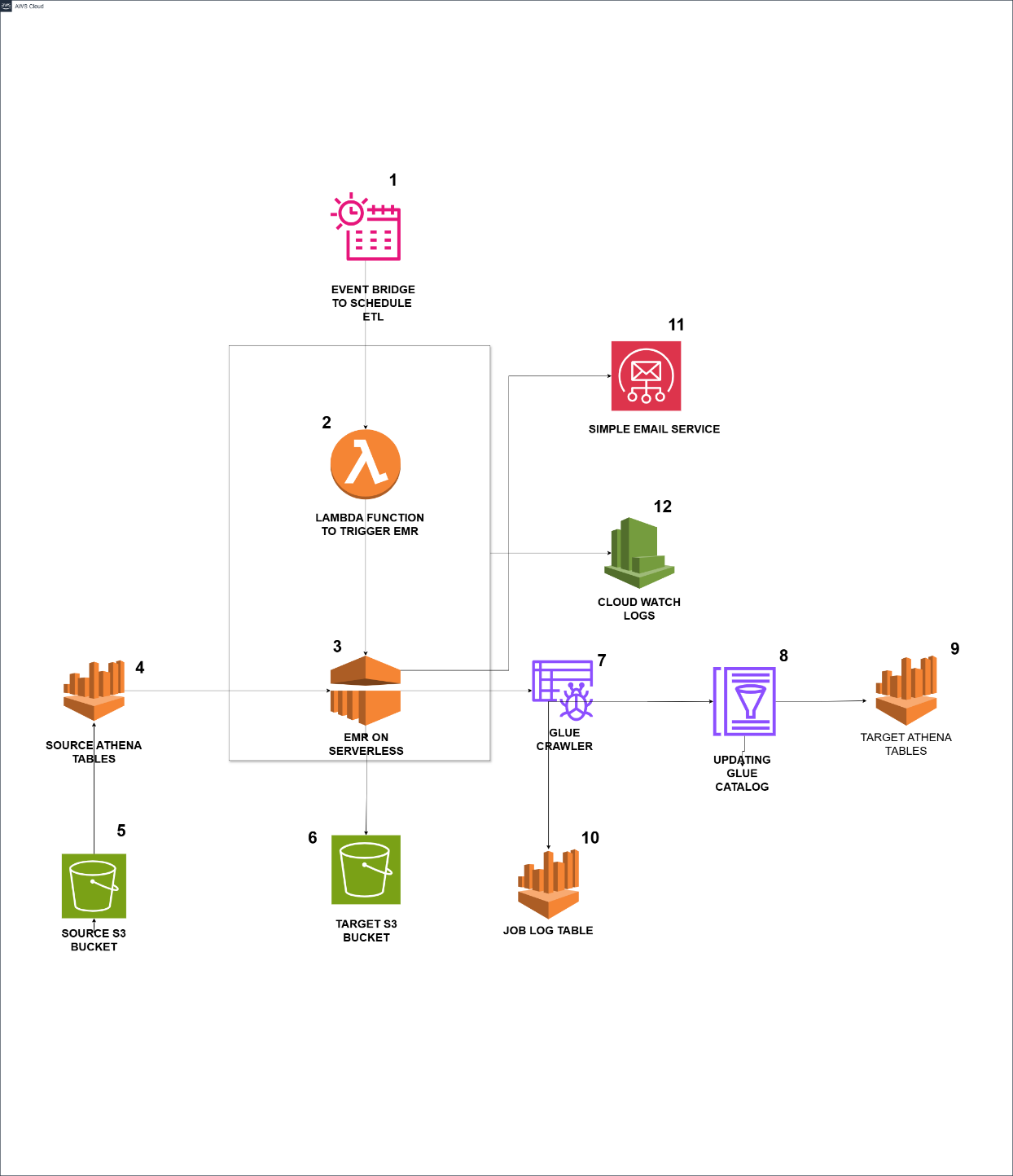
Task Node: 0

Total Nodes: 4 Nodes

**COST PRICING DETAILS :**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Usage Type** | **Run Time** | **Days** | **Calculation** | **Cost** |
| Development Phase | Approx 6 hours a day | 3 weeks | 4 Nodes \* 0.192 \* 6 hour = $4.608 per day \* 21 days = $96.768 USD | **$96.768 USD** |
| Testing Phase | Approx 6 hours a day | 1 week | 4 Nodes \* 0.192 \* 6 hour = $4.608 per day \* 7 days = $32.256 USD | **$32.256 USD** |
| Production | Approx 1 hour a day | Per Month | 4 Nodes \* 0.192 \* 2 hour = $1.536 per day \* 30 Days = $46.08 | **$46.08 USD** |

**ARCHITECTURE DIAGRAM FOR EMR SERVERLESS:**

****

**DESCRIPTION:**

1. Based on the specified schedule, EventBridge triggers the Lambda Function, initiating the process.
2. The Lambda Function then triggers the EMR Serverless, which starts running.
3. The EMR serverless gets triggered from lambda and it starts executing the PySpark Script.
4. PySpark script is executed on the EMR cluster, which reads data from the Athena tables.
5. The PySpark script fetches data from the source S3 path using Athena queries.
6. The data retrieved from Athena is loaded into the target S3 path as parquet files, organized in partitioned folders.
7. Once the PySpark script completes execution on EMR, a Glue Crawler is triggered to run.
8. The Glue Crawler populates the tables in the AWS Glue Data Catalog.
9. The newly populated target tables in Glue can now be queried using Athena.
10. The PySpark script on EMR generates a Job Log Table containing runtime information and table counts.
11. Through SES, PySpark script from EMR Serverless sends an email to the users, notifying them the job status and providing the job log table details.
12. All logs for the Lambda Function and EMR Serverless are stored in CloudWatch for monitoring and troubleshooting.

**EMR ON SERVERLESS:**

**DEFAULT CONFIGURATION:**

Spark Driver: 1

Spark Executor: 2

Size of driver: 4 vCPUs and 16 GB of memory, 20 GB disk

Size of Executor: 4 vCPUs and 16 GB of memory, 20 GB disk

Maximum Limit: 400 vCPU, 3000 GB Memory, 20000 GB disk

**COST CALCULATION FOR THE DEFAULT CONFIGURATION:**

per vCPU per hour: $0.052624

per GB per hour: $0.0057785

Compute vCPUs: Driver: 4 vCPUs \* $0.052 \* 1 hour = 4 vCPU-hours

Executor: 2 Executor \* 4 vCPUs \* 1 hour = 8 vCPU-hours

Total vCPU-hours:4+8 = 12 vCPU-hours

Compute Memory-Hours: Driver: 16 GB memory \* 1 hour = 16 GB-hours

Executors:2 executors \* 16 GB memory \* 1 hour = 32 GB-hours

Total Memory-Hours :16 + 32 = 48 GB-hours

vCPU cost: 12 vCPU-hours \* 0.05264 = $0.63168

Memory cost: 48 GB-hours \* $0.00577 = $0.27696

Total Cost per hour: $0.63168 + $0.27696 = $0.90864

**COST PRICING DETAILS :**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Usage Type** | **Run Time** | **Days** | **Calculation** | **Cost** |
| Development Phase | Approx 6 hours a day | 3 weeks | $0.90864 \* 6 hour = $5.451 per day \* 21 days = $114.471 USD | **$114.471 USD** |
| Testing Phase | Approx 6 hours a day | 1 week | $0.90864 \* 6 hour = $5.451 per day \* 7 days = $38.162 USD | **$38.162 USD** |
| Production | Approx 2 hour a day | Per Month | $0.90864 \* 2 hour = $1.817 per day \* 30 Days = $54.51 | **$54.51 USD** |

**OTHER SERVICES COST DETAILS :**

**S3** – S3 cost is based on how much data we stored in s3, we’re loading and overwriting 20GB data approximately every run.

For all Development phase/ Testing Phase/ Production deployment, it only overwrites approx. 20 GB every run

Per day: ~$0.015

Per month: ~$0.46 for 20 GB of overwritten data.

**Lambda Function:** we’re invoking lambda function 2 times for 5 min runtime for every run

Monthly Cost: $0.38 USD per month

**Cloud Watch Logs:**  Cloud watch logs costs based on volume of logs storing in cloud watch from EMR or glue or lambda function.

Per GB: approx. $0.50 USD

**SES Simple Email Service :** If we are sending 300 emails per month approximately

300 emails Per month: $0.03 USD