**Migrating Legacy database system to AWS**

Contents

[1. Background: 2](#_Toc123526861)

[2. Motivation and Focus: 2](#_Toc123526862)

[3. High Level Architecture: 2](#_Toc123526863)

[3.1. Source: (MySQL database) 3](#_Toc123526864)

[For this use case we have considered the sources to be present in multiple legacy database systems like Oracle, MySQL, SQL Server and so on. 3](#_Toc123526865)

[3.2 Data Lake: (Amazon S3) 3](#_Toc123526866)

[3.3. Target Data model: 3](#_Toc123526867)

[4. Step 1: Migration of Legacy DB to AWS Data Lake in S3 4](#_Toc123526868)

[4.1. Create IAM user and role: 4](#_Toc123526869)

[4.2. Create an EC2 instance 5](#_Toc123526870)

[4.3. Source MySQL instance: 6](#_Toc123526871)

[4.4. Create Intermediate Data lake in S3: 7](#_Toc123526872)

[4.5. Create Database Migration Service: 8](#_Toc123526873)

[4.5.1. Create replication instance: 8](#_Toc123526874)

[4.5.2. Create source and target endpoints: 8](#_Toc123526875)

[4.5.3. Create a database migration task: 9](#_Toc123526876)

[4.5.4. Testing for migration of data from MySQL to s3 using DMS task 10](#_Toc123526877)

[4.6. CloudWatch metric: 11](#_Toc123526878)

[5. Step 2: Migration of data from S3 to AWS RDS, Athena 11](#_Toc123526879)

[5.1. AWS Services used: 12](#_Toc123526880)

[5.2. Migrating Amazon S3 to Amazon RDS MySQL using Glue: 12](#_Toc123526881)

[5.2.1. Create Target Amazon RDS: 12](#_Toc123526882)

[5.2.2. Secrets Manager: 13](#_Toc123526883)

[5.2.2. Create Target Tables in Amazon RDS: 14](#_Toc123526884)

[5.3. Setup AWS Glue 14](#_Toc123526885)

[5.3.1. Create Connection: 14](#_Toc123526886)

[5.3.2. Create Crawlers: 15](#_Toc123526887)

[5.3.3. Glue Target tables: 15](#_Toc123526888)

[5.3.4. AWS GLUE Job: 16](#_Toc123526889)

[6. Post Migration 17](#_Toc123526890)

[7. Next Steps: 18](#_Toc123526891)

[7.1. Alternate Design: 18](#_Toc123526892)

[7.2. Other tasks 18](#_Toc123526893)

[8. Cost Analysis: 18](#_Toc123526894)

[9. Learnings: 19](#_Toc123526895)

[10. References: 20](#_Toc123526896)

[11. Conclusion 20](#_Toc123526897)

# Background:

Aspen Capital is in the process of migrating legacy databases to AWS. The legacy databases are located on prem in a colocation facility. Aspen Capital needs to migrate the data in a cost effective way and be able to operate it with a small ops team. They need a pipeline that will sync the data to an AWS data lake and then ETL it into a datastore (e.g RDS, Athena, etc.) to provide the data source for the new applications being built. While the migration is happening the AWS and on prem data need to stay in sync (some delay is allowed. Part of the submission is to decide what delay makes sense).

# Motivation and Focus:

The main motivation of this migration is to minimize the operational pain points of the legacy database systems and migrate it to a more scalable, automated, low-cost environment using AWS services. In this use case, I have focused on making the data pipeline scalable keeping in mind the need of growing data in future, with tighter controls on data access. The main objective was to enable very fast availability of data in target Amazon RDS MySQL and easy exploration of data with faster response times using Athena and subsequently creating dashboards and reports using Amazon Quicksight. This would allow Aspen Capital to empower the business decision makers and managers to make data-driven decisions in a timely manner. In addition, I have also stressed on data security, classification, obfuscation, auditing, monitoring, logging, and compliance needs.

# High Level Architecture:

The below diagram shows a high level architecture of migrating the data from legacy Database system to AWS cloud services.

Diagram

Description automatically generated

Fig 1

On a high level, it consists of 2 steps:

Step 1: Migration of Legacy DB to AWS Data Lake in S3

Step 2: Migration of data from S3 to AWS services like RDS , Athena, etc.

## 3.1. Source: (MySQL database)

## For this use case we have considered the sources to be present in multiple legacy database systems like Oracle, MySQL, SQL Server and so on.

## 3.2 Data Lake: (Amazon S3)

The intermediate data lake is built on top of AWS S3 where we will store the raw data from the legacy systems as well as the transformed data to be fed into the target system.

## 3.3. Target Data model:

We have considered Amazon RDS MySQL as the final target database system. The target model is based on the following data model and I have created the Target tables in RDS keeping in mind the Referential integrity constraints of the transactional tables.

<https://dbdiagram.io/d/62268eff61d06e6eadbc43bc>

Although we will implement the pipeline and push the data in the Target tables in section 5, however, for visibility I have added the output of the target tables for your reference here:

Final target output data:



# Step 1: Migration of Legacy DB to AWS Data Lake in S3

The following diagram illustrates the AWS DMS replication process. In our use case the target is S3 instead of the database.

Diagram

Description automatically generated

## 4.1. Create IAM user and role:

* Created IAM user: arn: arn:aws:iam::055165295027:user/himadri
* Created IAM role: **DMStoS3**: Given S3 Full Access (Write and Delete access are necessary)

IAM user

Graphical user interface, text, application

Description automatically generated

IAM roles:

Graphical user interface, text, application, email

Description automatically generated

## 4.2. Create an EC2 instance

Graphical user interface, application

Description automatically generated

## Source MySQL instance:

For our use case, we have hosted a MariaDB MySQL instance as source database.

Refer below documents to create the MySQL instance:

<https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Source.MySQL.html#CHAP_Source.MySQL.CustomerManaged>

<https://mariadb.com/kb/en/mariadb-basics/>

<https://mariadb.com/kb/en/mysql-command-line-client/>

Root User ->

 ec2-user@ip-172-31-6-23 ~]$ mysql -uroot -p;

 Password – himadri

 User for DMS ->

$ mysql -u dmsuser -p;

 Password – dmstestuser

 Database Created – ASPENTEST

Graphical user interface, text, application, email

Description automatically generated

Created 2 tables borrower and role\_profiles under database: ASPENTEST

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

## Create Intermediate Data lake in S3:

Create a bucket in S3 and name it as : **aspendatalake**

Graphical user interface, text, website

Description automatically generated

Refer below link for setting up the S3 data lake: <https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Target.S3.html#CHAP_Target.S3.Configuring>

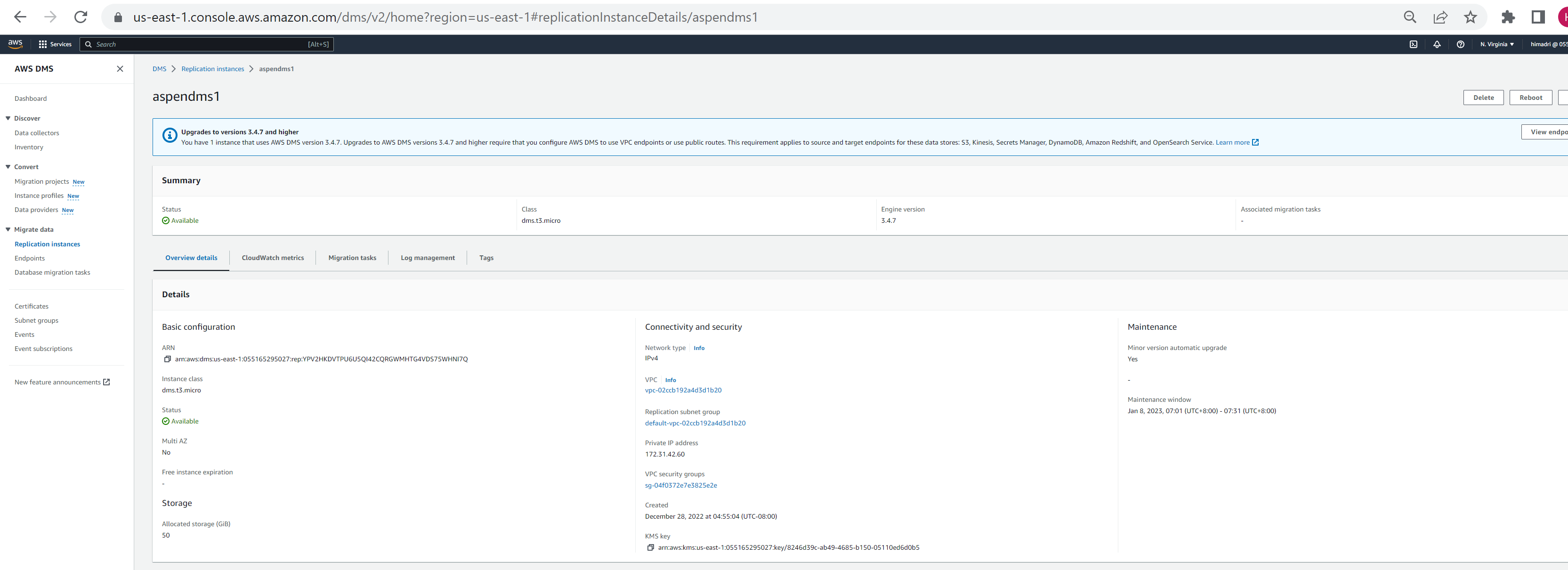
## Create Database Migration Service:

We will setup the DMS service to take care both historical and the CDC loads.

### Create replication instance:

Graphical user interface, text, application

Description automatically generated



### Create source and target endpoints:

Create the source and target endpoint for migration of data in database migration service.

Graphical user interface, text, application, email

Description automatically generated

Create source endpoint:

Graphical user interface, text, application, email

Description automatically generated

Create target endpoint and test: S3 data lake:

Graphical user interface, text, application, email

Description automatically generated

S3 format: For testing purpose we have used csv format for data storage. However it is recommended to use “Parquet” data format for production use to get the benefit of faster query execution and lesser data storage.

You may have to update the “Extra connection attributes” for modification.

Recommendation: Focus on below:

* Data storage format
* Partition if needed (for example on a daily basis, etc.)

### Create a database migration task:

Create a database migration task for migrating the existing data from source to target.

Graphical user interface, application

Description automatically generated

Graphical user interface, application, Teams

Description automatically generated

### Testing for migration of data from MySQL to s3 using DMS task

Start the data migration task by modifying the task created, once the replication is started it will replicate all data to s3 bucket created.

Graphical user interface, text, application, email

Description automatically generated

Once the DMS migration job executes successfully, it should look like below:

Graphical user interface, application

Description automatically generated

Check the s3 bucket where we can get the output as a database folder wise and tables in csv format.

Sample data in S3 query select:

Borrower table data in S3:  
Graphical user interface, text, application

Description automatically generated

Role profiles data in S3:

Graphical user interface, text, application, email

Description automatically generated

## CloudWatch metric:

Key CloudWatch metric at Task level that can be the guiding factor for cutover decision

Graphical user interface, application, email

Description automatically generated

# Step 2: Migration of data from S3 to AWS RDS, Athena

The architecture contains the following key components:

I have tried to de couple the storage and compute services and tried to access it independently.

## 5.1. AWS Services used:

**Data Storage:** Amazon S3 for raw data landing and transformed data staging. The raw datasets from the legacy data will be landed to this layer. Then the intermediate transformed data will be stored in a different bucket in S3.

[Amazon Simple Storage Service](https://aws.amazon.com/s3) : <https://aws.amazon.com/s3/>

Amazon RDS MySQL: In this use case we will consider AWS RDS MySQL as our final target database.

**Data Compute:** Data Processing and computation will be handled by AWS Glue. AWS Glue ETL jobs will be used to extract, transform, and load (ETL) data from the Amazon S3 landing zone to Amazon S3 transformed zone in optimal format (preferably parquet format) and layout. In addition, we have used AWS Glue crawler to update the AWS Glue Data Catalog.

AWS Glue: <https://aws.amazon.com/glue/>

**Monitoring and Notification:** AWS Cloudwatch and SNS for Email notification.

**Orchestration**: using AWS Lambda and AWS step functions for ETL job trigger and Glue Crawler trigger.

<https://aws.amazon.com/step-functions/>

**Data Consumption:**

We will use Amazon Athena and Quicksight for Data consumption.

Athena will be used for ad-hoc quick and extensive SQL data analysis and to build a logical layer on the landing zone.

<https://aws.amazon.com/athena/>

Quicksight will be used for reporting and creating dashboards.

<https://aws.amazon.com/quicksight/>

**Deployment and IaaS:**

The entire infrastructure and the deployment can be automated using [AWS CloudFormation](https://aws.amazon.com/cloudformation).

However, due to shortage of time for this use case, I have not been able to automate the deployment.

<https://aws.amazon.com/cloudformation/>

## 5.2. Migrating Amazon S3 to Amazon RDS MySQL using Glue:

In this section we will focus on creating a pipeline to move data from S3 transformed layer to Amazon RDS using Glue.

### 5.2.1. Create Target Amazon RDS:

Target RDS instance: **aspentargetrds**

Graphical user interface, text, application

Description automatically generated

Create VPC, security groups to be used here. Security groups: helps to set inbound and outbound rules

* DB instance identifierInfo: aspentargetrds
* Security Groups: AspenRDSSG
* Master username/pwd: admin/ N\_EN\*5\_2m:.~PaGt1eOo]4?E0jo6

### 5.2.2. Secrets Manager:

Note that the password has been created using Secrets Manager.

Graphical user interface, application

Description automatically generated

Once AWS RDS target is created, you can setup and check the connection using MySQL Workbench.

Graphical user interface, text, application, email

Description automatically generated

Once connection is successful you can create the Target tables based on the Create table DDl statements as attached below.

### 5.2.2. Create Target Tables in Amazon RDS:



## Setup AWS Glue

### Create Connection:

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

### Create Crawlers:

We need to create 2 crawlers one for source and another for target

Graphical user interface, application

Description automatically generated

### Glue Target tables:

Glue Target Tables in RDS:

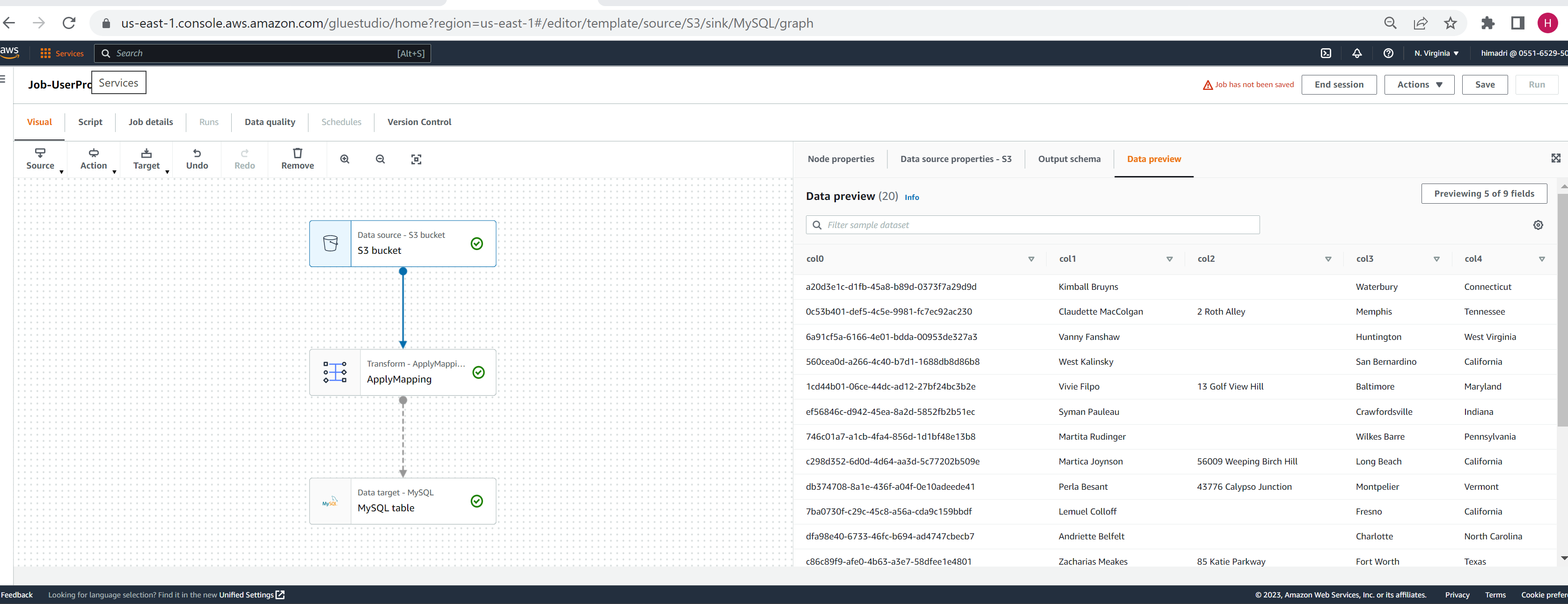
Graphical user interface, application

Description automatically generated

### AWS GLUE Job:

Source data in S3:

Source sample data for User profile:



Target sample data in AWS RSD MySQL database:

Graphical user interface, text, application, email

Description automatically generated

# Post Migration

Once the migration is complete and the system is up and running and data is validated, we need to understand the business requirement so that we can deprecate the legacy database system and point out application services directly to the AWS RDS instance. However, the below architecture shows a sample option to switch connection to the new RDS instance after migration.

Diagram

Description automatically generated

# Next Steps:

In the interest of time, I was not able to focus on few open ends. However we can focus on below areas once we get more time:

## 7.1. Alternate Design:

1. Instead of Glue we can directly write Pyspark code or SparkSQL code on a “Spark on EMR” cluster and can maintain the orchestration using Airflow. However this is a topic under discussion and need more inputs from the business.
2. The data can directly be fed from source MySQL to target Amazon RDS MySQL using DMS and then we can create a data lake in S3. However we need more input to design this data flow and setup the pipeline.

On a high level it looks like:

**Source legacy DB -> DMS (transformations directly in DMS)-> Amazon RDS -> DMS -> S3(Data Lake) -> Athena -> Quicksight**

## 7.2. Other tasks

1. Optimization of the Data pipeline (using optimized low cost EC2 instances, using proper RDS cluster, etc.) and using best practices.
2. Data Monitoring and setup Cloud Watch metrics
3. Deployment using Cloudformation
4. Understand the business logic to finalize the exact tools like RDS, Aurora, etc.
5. Machine learning and Data Science related task: If we need to partner with the Data Science and Machine learning teams and feed in the data to their model, we will have to understand their requirement and create feature sets in S3 and can pull data using Sagemaker.
6. Data Privacy: We need to keep in mind the Data Privacy compliances after the recent implementation of GDPR, Coppa, CCPA, etc. and need to make sure that we are not exposing our data to unintended users.

# 8. Cost Analysis:

Pricing is a very important factor for data migration engagements. Not only we need to consider the one time migration and CDC loads, but also we will have to analyze the ongoing day to day cost for running the AWS services. Note that all services are not free even while testing using the AWS Free tier.

Check cost of running the AWS services: <https://calculator.aws/#/addService/DMS>

Few best practices to consider:

If you are not using the EC2 instance, shut it down. To avoid incurring ongoing charges, check and stop all paid services after the demo.

Graphical user interface, text, application, email

Description automatically generated

# Learnings:

Error faced and resolution:

Note: My SQL workbench RDS connects through Internet gateway.

Graphical user interface, application

Description automatically generated

However for AWS Glue, you need to connect though NAT Gateway:

Graphical user interface, application

Description automatically generated

# References:

1. Using Amazon S3 as a target for AWS Database Migration Service: <https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Target.S3.html>

* DMS: <https://docs.aws.amazon.com/dms/latest/sbs/dms-sbs-welcome.html>
* DMS: <https://docs.aws.amazon.com/dms/?id=docs_gateway>
* Setting global variables at database level for S3 access: <https://dev.mysql.com/doc/refman/5.7/en/set-variable.html>

# Conclusion

All in all, it was fun working on this use case. Although, this use case is not very complicated, however, it is very challenging to implement, deploy and test the entire end to end pipeline within the stringent timeline. Also, I have made few assumptions since the requirement was given on a high level and need to discuss with the business team to understand the requirement in a more detailed level.

The first step to break down data silos and running analytics to gain insights from all the data is to build a data lake. In this engagement, I was able to set up a sustainable working framework on AWS infrastructure using MySQL, DMS, S3, RDS, IAM, Secrets Manager, EC2, Glue to name a few without compromising the quality. We can consider this as a base framework and can move forward to implement and fine tune the end-to-end framework after getting more details form the business.

I am happy to discuss with the Aspen Capital team to understand the business requirement in a more detailed manner and propose and recommend the best architecture on an AWS based cloud environment. Thanks.