TABLE OF CONTENTS

1. Introduction

In recent years, due to the exponential growth of data and growing demand for Data driven insights, interest in ETL (Extract, Transform, Load) have grown rapidly. ETL enables organizations to consolidate data from different sources, transform it into a structured format, and load it into a target database or data warehouse for analysis and reporting, a critical process in data integration.

ETL, which stands for extract, transform and load, is a data integration process that combines data from multiple data sources into a single, consistent data store that is loaded into a data warehouse or other target system.

As the databases grew in popularity in the 1970s, ETL was introduced as a process for integrating and loading data for computation and analysis, eventually becoming the primary method to process data for data warehousing projects.

ETL provides the foundation for data analytics and machine learning workstreams. Through a series of business rules, ETL cleanses and organizes data in a way which addresses specific business intelligence needs, like monthly reporting, but it can also tackle more advanced analytics, which can improve back-end processes or end user experiences. ETL is often used by an organization to:

- Extract data from legacy systems
- Cleanse the data to improve data quality and establish consistency.
- Load data into a target database.

How ETL works:

The easiest way to understand how ETL works is to understand what happens in each step of the process.

i. Extract

During data extraction, raw data is copied or exported from source locations to a staging area. Data management teams can extract data from a variety of data sources, which can be structured or unstructured. Those sources include but are not limited to:

- SQL or NoSQL servers
- CRM and ERP systems
- Flat files
- Email
- Web pages

ii. Transform

In the staging area, the raw data undergoes data processing. Here, the data is transformed and consolidated for its intended analytical use case. This phase can involve the following tasks:

- Filtering, cleansing, de-duplicating, validating, and authenticating the data.
- Performing calculations, translations, or summarizations based on the raw data. This can include changing row and column headers for consistency, converting currencies or other units of measurement, editing text strings, and more.
- Conducting audits to ensure data quality and compliance.
- Removing, encrypting, or protecting data governed by industry or governmental regulators.
- Formatting the data into tables or joined tables to match the schema of the target data warehouse.

iii. Load

In this last step, the transformed data is moved from the staging area into a target data warehouse. Typically, this involves an initial loading of all data, followed by periodic loading of incremental data changes and, less often, full refreshes to erase and replace data in the warehouse. For most organizations that use ETL, the process is automated, well-defined, continuous and batch driven. Typically, ETL takes place during off-hours when traffic on the source systems and the data warehouse is at its lowest.

Why is ETL important?

Organizations today have both structured and unstructured data from various sources including:

- Customer data from online payment and customer relationship management (CRM) systems
- Inventory and operations data from vendor systems
- Sensor data from Internet of Things (IoT) devices
- Marketing data from social media and customer feedback
- Employee data from internal human resources systems

By applying the process of ETL, individual raw datasets can be prepared in a format and structure that is more consumable for analytics purposes, resulting in more meaningful insights. For example, online retailers can analyze data from points of sale to forecast demand and manage inventory. Marketing teams can integrate CRM data with customer feedback on social media to study consumer behavior.

Amazon Redshift is a fully managed, petabyte-scale data warehouse service in the cloud that offers fast query performance using the same SQL-based tools and business intelligence applications that you use today. Many customers also like to use Amazon Redshift as an extract, transform, and load (ETL) engine to use existing SQL developer skillsets, to quickly migrate pre-existing SQL-based ETL scripts, and—because Amazon Redshift is fully ACID-compliant—as an efficient mechanism to merge change data from source data systems.

This project uses AWS Step Functions and AWS Glue Python Shell to orchestrate tasks for those Amazon Redshift-based ETL workflows in a completely serverless fashion. AWS Glue Python Shell is a Python runtime environment for running small to medium-sized ETL tasks, such as submitting SQL queries and waiting for a response. Step Functions coordinates multiple AWS services into workflows so you can easily run and monitor a series of ETL tasks.

Both AWS Glue Python Shell and Step Functions are serverless, which allows to automatically run and scale them in response to events you define, rather than requiring you to provision, scale, and manage servers.

While many traditional SQL-based workflows use internal database constructs like triggers and stored procedures, separating workflow orchestration, task, and compute engine components into standalone services allows you to develop, optimize, and even reuse each component independently. So, this project uses Amazon Redshift as an example, the aim is to more generally show how to orchestrate any SQL-based ETL.

2. Problem statement

In the dynamic and complex realm of urban transportation data analysis, the New York City (NYC) transportation dataset, which includes an extensive array of information about taxi trips, ride-sharing data, and other transportation metrics, presents a unique set of challenges and opportunities. The primary challenge lies in the efficient management and processing of this vast and intricate dataset to extract actionable insights into urban mobility, traffic patterns, and passenger behavior.

The development of an effective Extract, Transform, Load (ETL) pipeline is crucial for handling the complexities and scale of the NYC dataset. The goal of this project is to design and implement a robust, scalable, and reliable ETL process that leverages state-of-the-art cloud computing and data warehousing technologies.

Key components of the proposed solution include:

- Amazon Redshift: Utilized as the central data warehousing solution, Amazon Redshift will provide the storage and query capabilities needed to handle large-scale datasets efficiently. Its columnar storage and massively parallel processing (MPP) architecture make it particularly suited for aggregating and analyzing large volumes of data.

- AWS Step Functions: To orchestrate the ETL workflow, AWS Step Functions will be employed. This service will enable seamless coordination of the various ETL tasks, ensuring they are executed in an orderly, reliable, and efficient manner. Step Functions' ability to handle complex workflows makes it ideal for managing the dependencies and contingencies inherent in the ETL process.

- AWS Glue: This fully managed, serverless data integration service will be used for the data transformation part of the ETL process. AWS Glue will simplify the task of preparing and transforming the data for analysis, handling tasks such as data cleansing, normalization, and deduplication.

- QuickSight: For data visualization and business intelligence, QuickSight will be utilized. This tool will enable stakeholders to interact with the processed data through intuitive interfaces, creating dashboards and reports that provide insights into trends, patterns, and anomalies within the urban transportation domain.

The implementation of this ETL pipeline must prioritize high availability, scalability, and flexibility to adapt to the ever-changing nature of the NYC transportation dataset. The final

solution should not only handle the current data volume and complexity but also be scalable to accommodate future growth in data size and additional data sources. This approach will provide a comprehensive and nuanced understanding of New York City's transportation landscape, aiding in effective urban planning and policy-making.

3. Scenario

This ETL workflow utilizes the 'TLC Trip Record Data' dataset and encompasses two primary tasks, forming a streamlined ETL process:

- Task 1: Transfer a copy of the dataset, specifically focusing on yellow taxi trips, from its original Amazon S3 location to a designated table in Amazon Redshift.
- Task 2: Generate and export a series of output files to a different Amazon S3 location. These files will specifically highlight key metrics and patterns from the yellow taxi trip data, such as peak travel times, most frequented routes, and average fares, categorized by various dimensions like time, location, and trip characteristics. This detailed output will empower the analytics team to derive valuable insights into urban mobility and taxi service efficiency.

This dataset is publicly available via an Amazon Simple Storage Service (Amazon S3) bucket from Registry of Open Data on AWS.

4. Dataset

The New York City Taxi and Limousine Commission (NYC TLC) Yellow Trips dataset is a comprehensive collection of data that pertains to yellow taxi trips in New York City. This dataset is particularly valuable for a variety of analyses, including urban planning, transportation studies, and economic research.

Field Name	Description
VendorID	A code indicating the TPEP provider that provided the record. 1= Creative Mobile Technologies, LLC; 2= VeriFone Inc.
tpep_pickup_datetime	The date and time when the meter was engaged.
tpep_dropoff_datetime	The date and time when the meter was disengaged.
Passenger_count	The number of passengers in the vehicle. This is a driver-entered value.
Trip_distance	The elapsed trip distance in miles reported by the taximeter.
PULocationID	TLC Taxi Zone in which the taximeter was engaged
DOLocationID	TLC Taxi Zone in which the taximeter was disengaged

RateCodeID	The final rate code in effect at the end of the trip. 1= Standard rate 2=JFK 3=Newark 4=Nassau or Westchester 5=Negotiated fare 6=Group ride
Store_and_fwd_flag	This flag indicates whether the trip record was held in vehicle memory before sending to the vendor, aka “store and forward,” because the vehicle did not have a connection to the server. Y= store and forward trip N= not a store and forward trip
Payment_type	A numeric code signifying how the passenger paid for the trip. 1= Credit card 2= Cash 3= No charge 4= Dispute 5= Unknown 6= Voided trip
Fare_amount	The time-and-distance fare calculated by the meter.
Extra	Miscellaneous extras and surcharges. Currently, this only includes the $0.50 and $1 rush hour and overnight charges.
MTA_tax	$0.50 MTA tax that is automatically triggered based on the metered rate in use.
Improvement_surcharge	$0.30 improvement surcharge assessed trips at the flag drop. The improvement surcharge began being levied in 2015.
Tip_amount	Tip amount – This field is automatically populated for credit card tips. Cash tips are not included.
Tolls_amount	Total amount of all tolls paid in trip.
Total_amount	The total amount charged to passengers. Does not include cash tips.
Congestion_Surcharge	Total amount collected in trip for NYS congestion surcharge.
Airport_fee	$1.25 for pick up only at LaGuardia and John

Table 1. Data dictionary of yellow trips data

5. Approach

The steps in this process are as follows:

- The state machine launches a series of runs of an AWS Glue Python Shell job with parameters for retrieving database connection information from AWS Secrets Manager and an .sql file from S3.
- Each run of the AWS Glue Python Shell job uses the database connection information to connect to the Amazon Redshift cluster and submit the queries contained in the .sql file.
    - Task 1: The cluster utilizes Amazon Redshift Spectrum to read data from S3 and load it into an Amazon Redshift table. Amazon Redshift Spectrum is commonly used as a means for loading data to Amazon Redshift.
    - Task 2: The cluster executes an aggregation query and exports the results to another Amazon S3 location via UNLOAD.
- The state machine may send a notification to an Amazon Simple Notification Service (SNS) topic in the case of pipeline failure.
- Users can query the data from the cluster and/or retrieve report output files directly from S3 or use QuickSight for visualization and business intelligence.

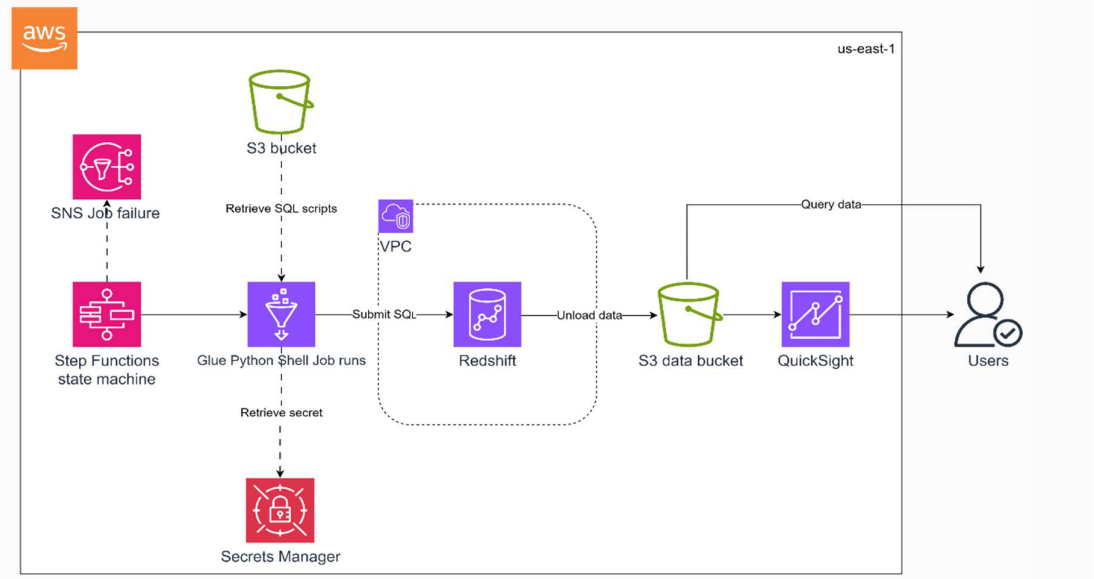


Fig. 1. Solutions architecture from end to end

6. Services Used

The following tools and services are used:

- Language: Python3, SQL.
- Services: Amazon Redshift, AWS Glue, AWS Step Function, VPC, S3, Secrets Manager, QuickSight, SNS.
- Libraries: boto3, sys.

6.1 AWS Redshift

Amazon Redshift is a data warehousing service provided by Amazon Web Services (AWS). It allows customers to store and analyze large amounts of data in a scalable and cost-effective way.

Redshift is based on a columnar data store, which means that it stores data in columns rather than rows, allowing for faster querying and data compression. It can handle petabyte-scale data warehouses and provides high performance for analytics and business intelligence applications.

Redshift is designed to work with a variety of data sources, including structured, semi-structured, and unstructured data. It can integrate with various AWS services such as Amazon S3, Amazon EMR, and Amazon Kinesis, as well as third-party tools such as Tableau, Power BI, and other BI tools.
Redshift is a fully managed service, meaning that AWS takes care of tasks such as provisioning, configuring, and monitoring the infrastructure, leaving the customer free to focus on their data analysis tasks. It also provides several security features, including encryption, access control, and compliance with industry standards such as HIPAA, PCI DSS, and SOC.

Here are some of the key features of Amazon Redshift:

- **Scalability:** Amazon Redshift is designed to scale to petabytes of data, allowing organizations to start small and grow their data warehouse as their needs increase.
- **Columnar Storage:** Redshift stores data in a columnar format, which provides faster query performance and better compression than traditional row-based storage.
- **Performance:** Redshift is optimized for high-performance analytics and can execute complex queries on large datasets in seconds.
- **Integration with other AWS services:** Redshift can easily integrate with other AWS services such as S3, EMR, and Kinesis, making it easier to ingest, transform, and load data from different sources.
- **Security:** Redshift provides several security features, including data encryption at rest and in transit, network isolation, and access control through AWS Identity and Access Management (IAM).
- **SQL support:** Redshift supports SQL, which makes it easier for organizations to leverage their existing BI tools and analytics platforms.
- **Cost-effective:** Redshift is cost-effective compared to traditional data warehousing solutions as it allows users to pay only for what they use, with no upfront costs or long-term commitments.
- **Automatic backups:** Redshift automatically backs up data to Amazon S3, ensuring that data is safe and easily recoverable in the event of a disaster or system failure.
- **Concurrency:** Redshift supports concurrent queries and multiple users, which means that users can run multiple queries at the same time without affecting the performance of the system.
- **Machine Learning integration:** Redshift integrates with AWS Machine Learning services, allowing users to use ML models to analyze data in their Redshift clusters.

6.2 AWS Glue

AWS Glue is a fully managed extract, transform, and load (ETL) service that makes it easy for customers to prepare and load their data for analytics. It is a serverless service that automatically provisions the infrastructure required to run ETL jobs, and scales resources up and down as needed to meet demand.

Glue provides a central metadata repository called the Glue Data Catalog, which contains metadata about data sources and targets, transformations, and jobs. This makes it easier to discover, catalog, and track data assets across an organization.

Glue supports various data sources such as Amazon S3, JDBC, and Amazon DynamoDB, and can extract data from these sources, transform it using built-in or custom transformations, and load it into data warehouses or other analytics platforms such as Amazon Redshift, Amazon EMR, or Amazon Athena.

Glue also provides a visual interface for building and testing ETL jobs, as well as APIs for programmatically creating, monitoring, and managing jobs. It integrates with AWS services such as AWS Lambda, AWS Step Functions, and Amazon CloudWatch for event-driven processing and monitoring.

Here are some of the key features of AWS Glue:

- **ETL Jobs:** AWS Glue simplifies the process of ETL by providing a fully-managed service that automates much of the infrastructure and provides a centralized metadata catalog.
- **Serverless:** Glue is serverless, which means that AWS provisions the infrastructure required to run ETL jobs automatically, and scales resources up and down as needed to meet demand.
- **Data Catalog:** Glue provides a central metadata repository called the Glue Data Catalog, which contains metadata about data sources and targets, transformations, and jobs. This makes it easier to discover, catalog, and track data assets across an organization.
- **Data Source Connectivity:** Glue supports various data sources such as Amazon S3, JDBC, and Amazon DynamoDB, and can extract data from these sources, transform it using built-in or custom transformations, and load it into data warehouses or other analytics platforms such as Amazon Redshift, Amazon EMR, or Amazon Athena.
- **Built-in Transformations:** Glue provides a library of built-in transformations that can be used to transform data, and also allows users to create custom transformations using Python or Scala.
- **Data Validation:** Glue can validate data during ETL, ensuring that the data conforms to the expected format and quality.
- **Visual Interface:** Glue provides a visual interface for building and testing ETL jobs, making it easier for users who are not proficient in programming.
- **APIs:** Glue provides APIs for programmatically creating, monitoring, and managing jobs.
- **Integration:** Glue integrates with AWS services such as AWS Lambda, AWS Step Functions, and Amazon CloudWatch for event-driven processing and monitoring.

6.3 AWS Step Functions

AWS Step Functions is a fully-managed service that makes it easy to coordinate the components of distributed applications and microservices using visual workflows. It enables developers to create complex workflows that integrate with various AWS services and custom applications, without having to worry about the underlying infrastructure.

Step Functions uses Amazon State Language (ASL) to define workflows as a series of states that represent tasks, decisions, and error handling. Each state in the workflow can invoke AWS services, Lambda functions, or custom applications, and the workflow transitions to the next state based on the results of the previous state.

Some of the key features of AWS Step Functions include:

- **Visual Workflows:** AWS Step Functions provides a visual interface for designing and visualizing workflows, making it easy for developers to understand and modify complex workflows.
- **Serverless:** Step Functions is a serverless service that automatically provisions the infrastructure required to run workflows, and scales resources up and down as needed to meet demand.
- **Built-in Error Handling:** Step Functions provides built-in error handling, allowing developers to define how the workflow should respond to errors or unexpected events.
- **Multiple Workflows:** AWS Step Functions can manage multiple workflows at the same time, making it easy to coordinate and manage distributed applications and microservices.
- **Integration:** Step Functions integrates with various AWS services such as AWS Lambda, Amazon SNS, Amazon SQS, and Amazon ECS, as well as custom applications.
- **Parallel Processing:** AWS Step Functions supports parallel processing, allowing developers to execute multiple branches of a workflow in parallel.
- **Task Retries:** AWS Step Functions provides built-in support for task retries, ensuring that failed tasks are automatically retried according to the specified retry policy.

6.4 AWS S3

Amazon S3 is an object storage service offering industry-leading scalability, data availability, security, and performance. This means customers of all sizes and industries can use it to store and protect any amount of data for a range of use cases, such as data lakes, websites, mobile applications, backup and restore, archive, enterprise applications, IoT devices, and big data analytics.

Key Features of Amazon S3:

- **Durability and Availability:** Amazon S3 provides high durability and availability by storing data across multiple globally dispersed data centers.
- **Security**: Offers sophisticated access controls, encryption at rest, and in transit, along with comprehensive auditing capabilities.
- **Scalability:** Easily manage large volumes of data, with the ability to scale up or down as needed.

- **Performance**: S3 is optimized for high-speed, low-latency access, making it ideal for various applications.
- **Data Management:** Provides features for data lifecycle management and automated tiering for cost-effective storage.
- **Integration:** Integrates seamlessly with a wide range of AWS and third-party services.

6.5 AWS VPC

Amazon Virtual Private Cloud (Amazon VPC) is a service that enables you to launch AWS resources into a virtual network that you have defined. It provides a logically isolated section of the AWS Cloud where you can launch Amazon EC2 instances, Amazon RDS databases, and other AWS resources.

With Amazon VPC, you have complete control over your virtual networking environment, including selection of your own IP address range, creation of subnets, and configuration of route tables and network gateways. You can also connect your VPC to your own data center using VPN or Direct Connect.

Here are some key features of Amazon VPC:

- **Isolation:** Amazon VPC provides a logically isolated section of the AWS Cloud, allowing you to launch AWS resources in a virtual network that you have defined.
- **Security:** Amazon VPC enables you to define security groups and network access control lists (ACLs) to control access to your resources.
- **Customizability:** Amazon VPC allows you to define your own IP address range, create subnets, and configure route tables and network gateways.
- **Connectivity:** Amazon VPC enables you to connect your VPC to your own data center using VPN or Direct Connect.
- **Scalability:** Amazon VPC allows you to launch and manage multiple instances of resources within a VPC, and scale the resources up or down as needed.
- **Integration:** Amazon VPC integrates with other AWS services such as Amazon EC2, Amazon RDS, and Amazon EMR.
- **Visibility:** Amazon VPC provides extensive visibility into your network traffic through the use of Amazon VPC Flow Logs.

6.6 AWS Secrets Manager

AWS Secrets Manager helps you protect access to your applications, services, and IT resources without the upfront cost and complexity of managing a secure infrastructure. This service enables you to easily rotate, manage, and retrieve database credentials, API keys, and other secrets throughout their lifecycle.

Key Features of AWS Secrets Manager:

- **Secret Rotation**: Automates the rotation of secrets safely without code changes to your applications.
- **Centralized Management:** Manage secrets centrally, eliminating the need to hard-code sensitive information in plain text.

- **Access Control**: Integrates with AWS Identity and Access Management (IAM) to control access to secrets.
- **Audit and Monitoring:** Integration with AWS CloudTrail for auditing secret changes and access.
- **Encryption:** Secrets are encrypted using encryption keys that you control through AWS Key Management Service (KMS).

6.7 AWS SNS

Amazon Simple Notification Service (AWS SNS) is a highly available, durable, secure, and fully managed pub/sub messaging service that enables you to decouple microservices, distributed systems, and serverless applications. AWS SNS provides a robust platform to send messages or notifications directly to users or distribute them to other services such as AWS Lambda, Amazon SQS, or HTTP/S endpoints.

Key Features of AWS SNS:

- **Flexibility**: AWS SNS supports a variety of subscribers, including web servers, email addresses, AWS Lambda functions, Amazon SQS queues, and SMS text messages.
- **Durability and High Availability**: Messages are stored redundantly across multiple availability zones, ensuring reliable message delivery.
- Scalability: SNS can handle a high throughput of messages, scaling automatically with the number of messages.
- **Filtering:** Message filtering allows subscribers to specify the messages of interest, reducing unnecessary network traffic and processing.
- **Security:** AWS SNS integrates with AWS Identity and Access Management (IAM), allowing you to control access to your topics and messages.
- **Integration:** Seamlessly integrates with other AWS services for comprehensive solutions.

6.8 AWS QuickSight

Amazon QuickSight is a fully managed business intelligence (BI) service that allows users to create interactive dashboards, perform ad-hoc analysis, and create data visualizations from a wide range of data sources. It is a cloud-based service that allows users to quickly and easily access the data and perform analysis without having to worry about the underlying infrastructure.

Some of the key features of Amazon QuickSight include:

- **Easy Data Visualization:** Amazon QuickSight provides an easy-to-use interface for creating and customizing data visualizations such as charts, graphs, and tables.
- **Access to Multiple Data Sources:** Amazon QuickSight can access a wide range of data sources including AWS services such as Amazon S3, Amazon RDS, and Amazon Redshift, as well as non-AWS sources such as Salesforce, MySQL, and Oracle.
- **Interactive Dashboards:** Amazon QuickSight enables users to create interactive dashboards that allow them to explore their data in real-time.

- **Integration with Other AWS Services:** Amazon QuickSight integrates with other AWS services such as Amazon S3, Amazon Redshift, and AWS Glue for easy data access and management.
- **Machine Learning-Powered Insights:** Amazon QuickSight provides machine learning-powered insights that help users understand their data better and gain new insights.
- **Advanced Analytics:** Amazon QuickSight supports advanced analytics such as forecasting, trend analysis, and anomaly detection.
- **Customizable:** Amazon QuickSight is highly customizable, allowing users to create their own themes and templates for their dashboards.

7. Prerequisites

- Create an AWS account
- Download the dataset.

Ways to interact with the AWS services:

- Console - Using the Web-based User Interface (UI).
- Common line interface (CLI) - Using the AWS CLI tool in Terminal/CMD.
- Cloud Development Kit (CDK)- Using infrastructure-as-a-code, e.g., Cloud Formation.
- Software Development Kit (SDK) - Programmatically, e.g., 'boto3' in python.

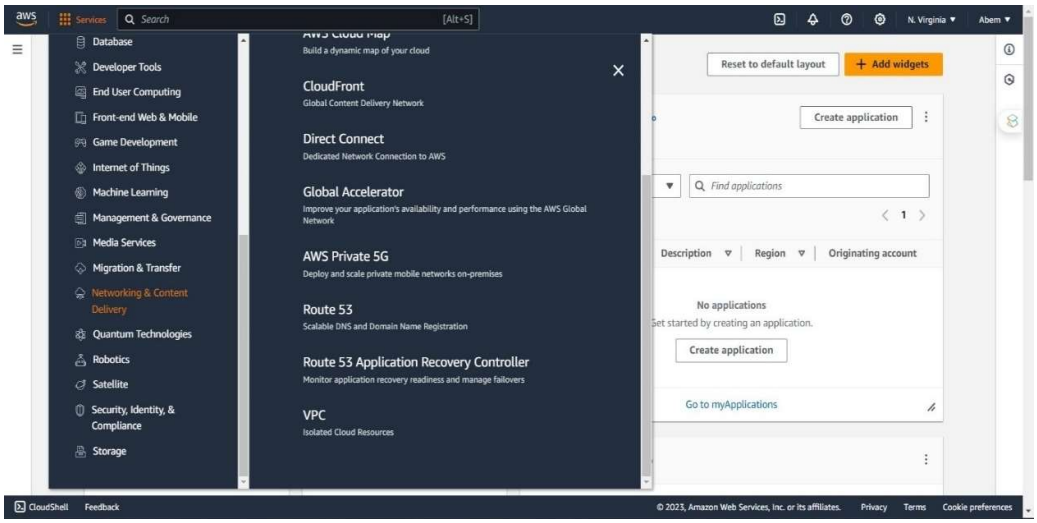
For this project, the CDK (CloudFormation) will not be used and instead the rest of the three methods will be used for execution.

8. Setup

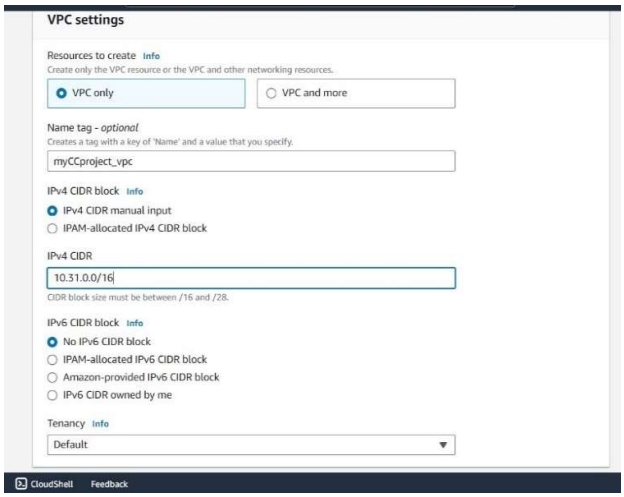
First, choose a region near to the S3 bucktes if you are using a public S3 or choose a nearest region.

8.1 Create VPC

i. Navigate to services > Networking and Content Delivery > VPC.

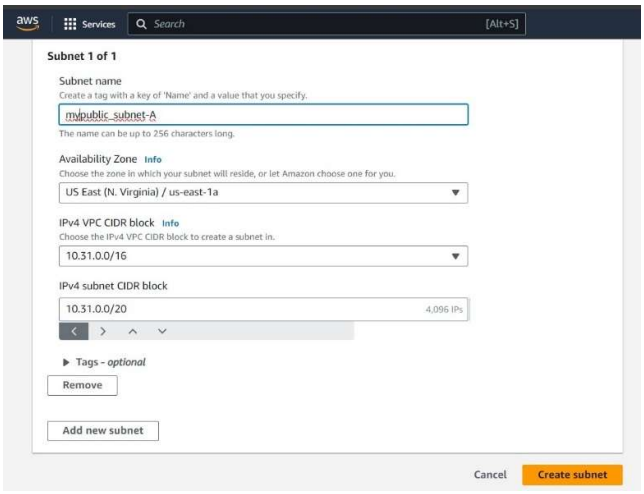


ii. Create a VPC "myCCProjectVPC" in us-east-1 region with IPV4 CIDR range: 10.31.0.0/16

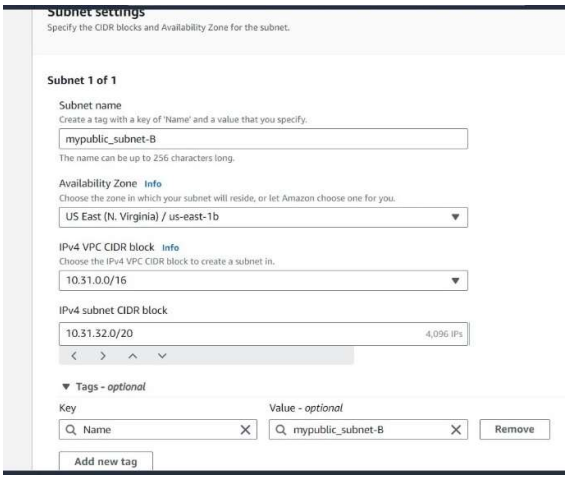


iii. Create the following subnets:

a. Public Subnet A - 10.31.0.0/20

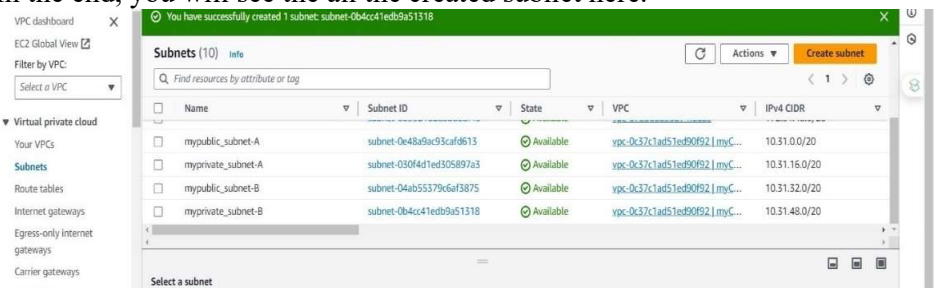


b. Private Subnet A - 10.31.16.0/20: Follow similar steps.
c. Public Subnet B - 10.31.32.0/20: Select a different availability zone.



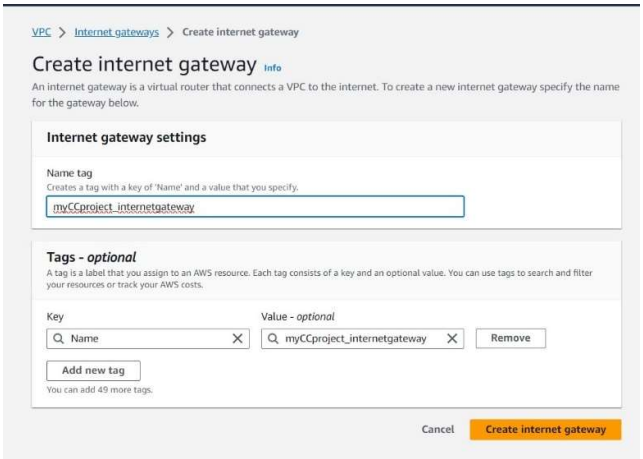
d. Private Subnet B - 10.31.48.0/20

In the end, you will see the all the created subnet here.

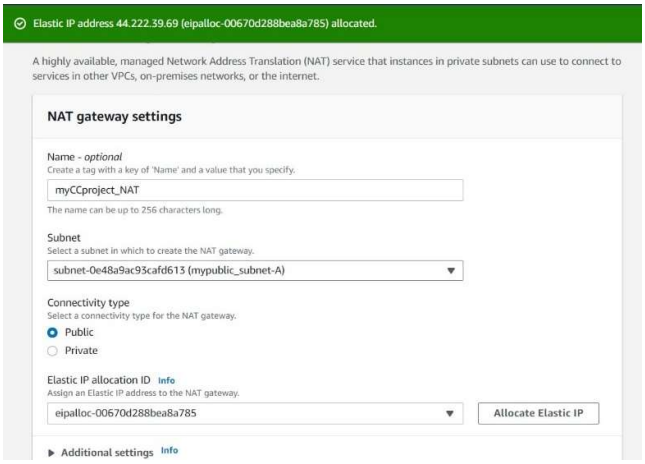


iv. Now Navigate to Internet gateway and create Internet Gateway-> "myprojectinternetgateway" and assign it to Public subnets A & B

v.

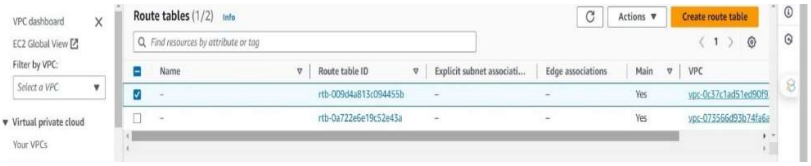


vi. Navigate to NAT gateway and allocate a NAT Gateway-> "myprojectNAT" and allocate an Elastic IP address.

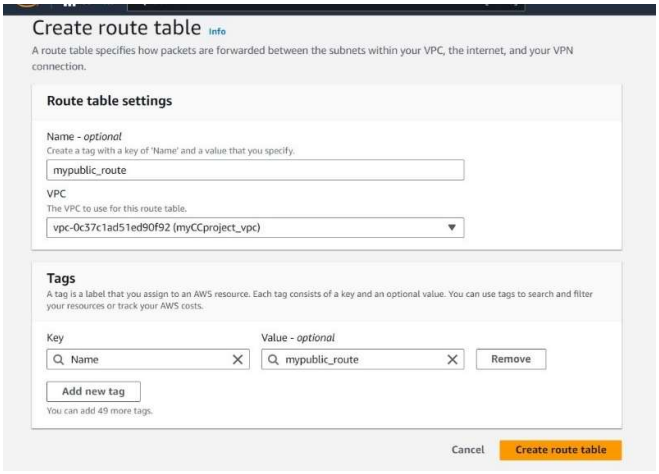


vii. Naviagte to route tables and create 4 route tables, one for each subnet.

a. First, select your project VPC.

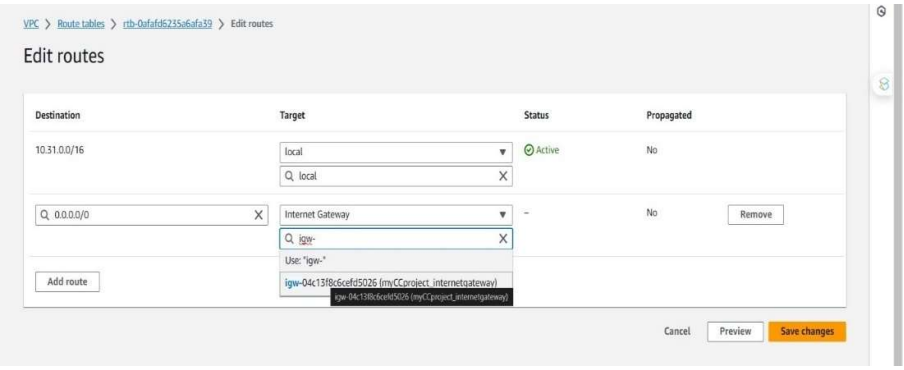


b. 2 Public subnets (A&B) will have Internet Gateway referred if traffic is routed for 0.0.0.0/0

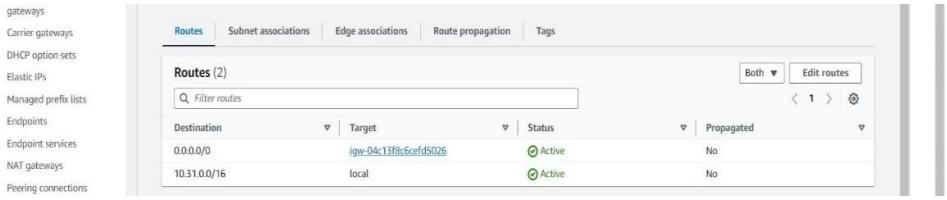


Please do not forget to attach the internet gateway to the VPC.

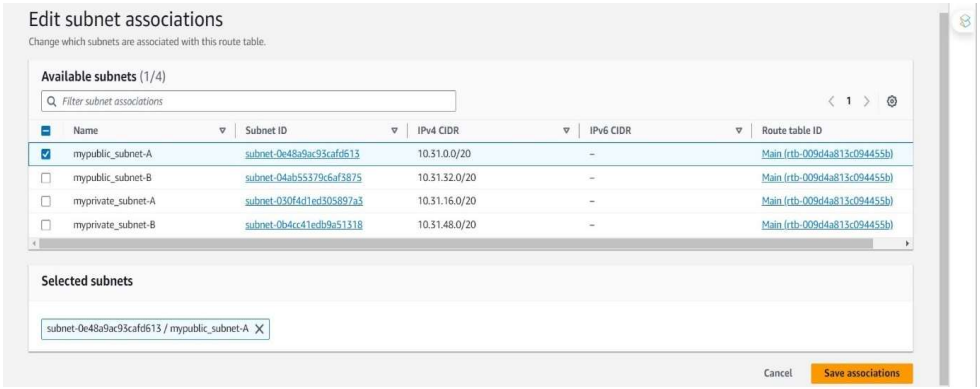
Now, provide the internet gateway for public route.



Now, you can see the routes destinations.



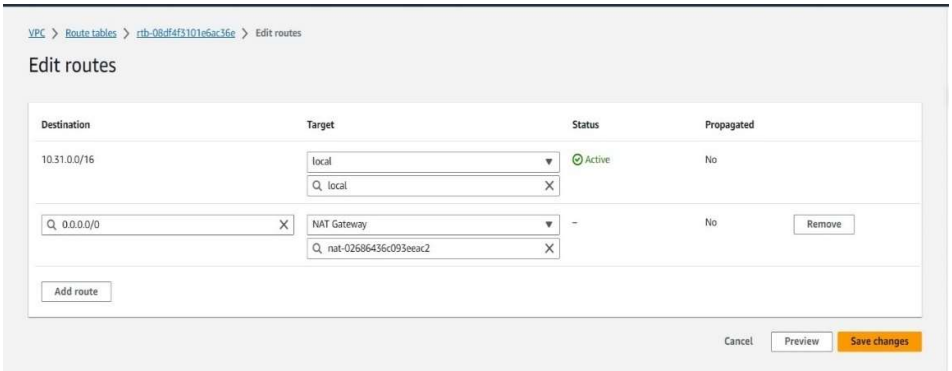
Next, is to associate subnet with the route table.



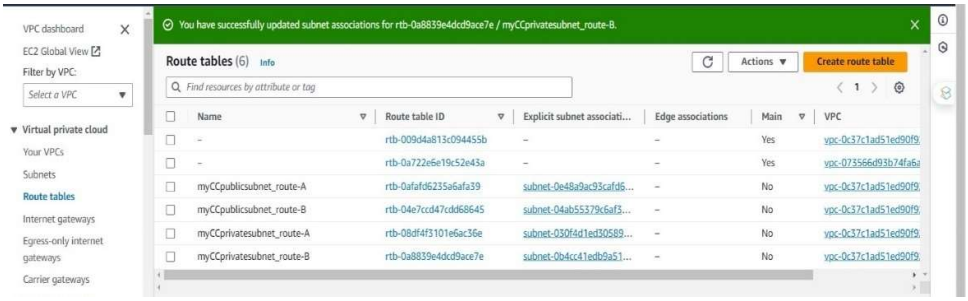
Follow, the same steps for route for public subnet B

c. 2 Private subnets(A&B) will have NAT Gateway referred if traffic is routed for 0.0.0.0/0

Follow same steps as that in public routes but use NAT gateway.



Now, the final route table is created.

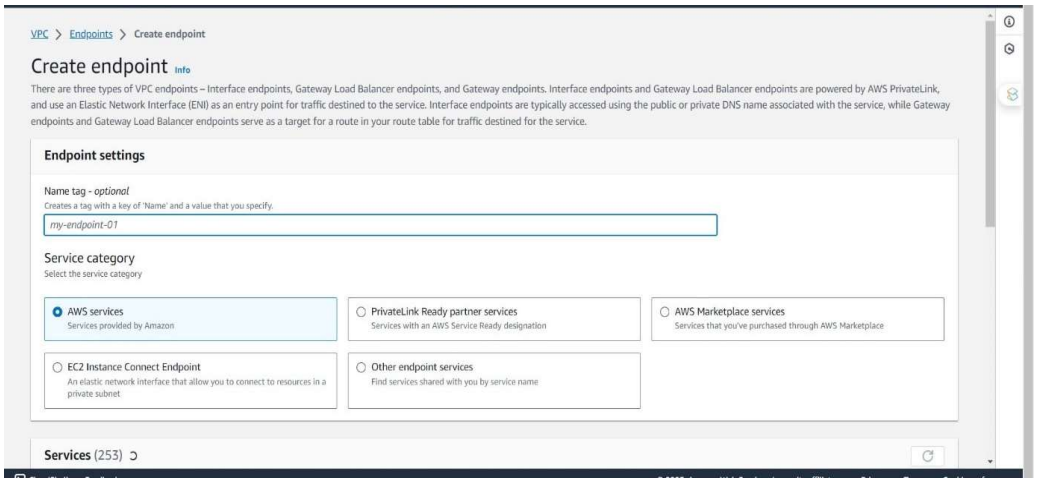


viii. Security groups and ACL:

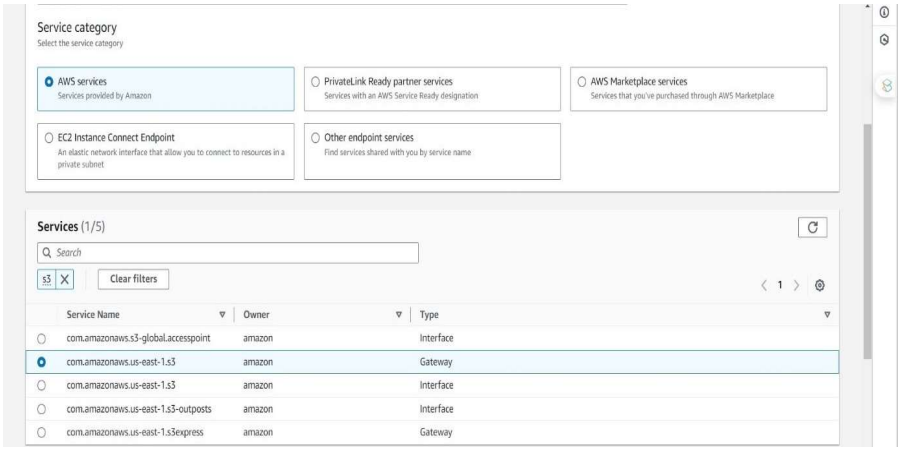
Use the default ones created.

ix. Navigate to end points and create the following:

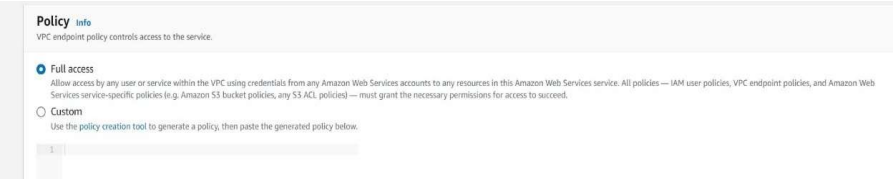
a. S3 endpoint



a. Select from the services.



b. For policy, allow full access or can change according to requirements.



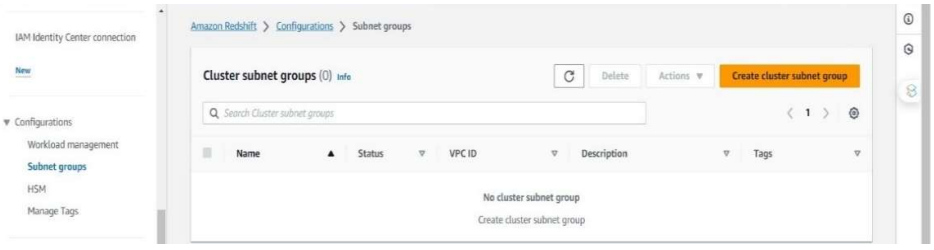
c. And attach routes tables to the S3 endpoint according to the requirements.

b. AWS manager endpoint

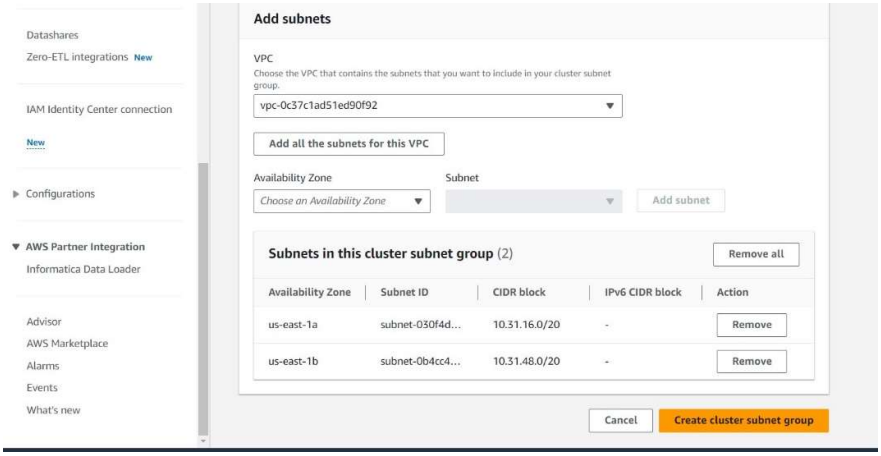
d. Follow the same steps as above and give the service as secret manager.

8.2 Create Redshift clusters

i. Navigate to Redshift from services.
ii. From the left panel, select subnets.

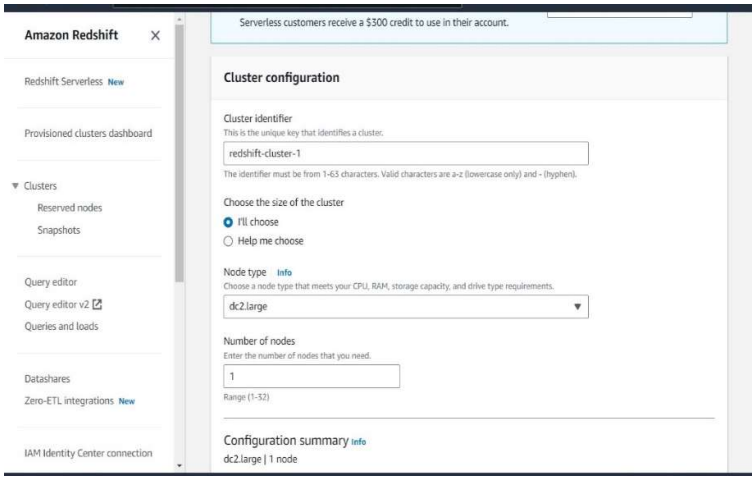


Subnet group should hold 2 private subnets (in 2 different regions) from VPC created.

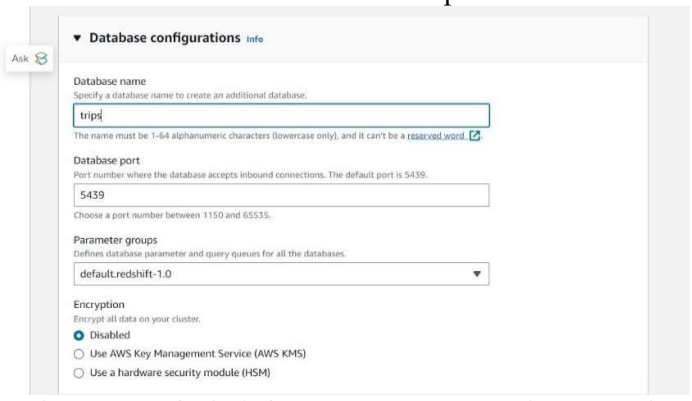


iii. Now, navigate to clusters and create clusters.

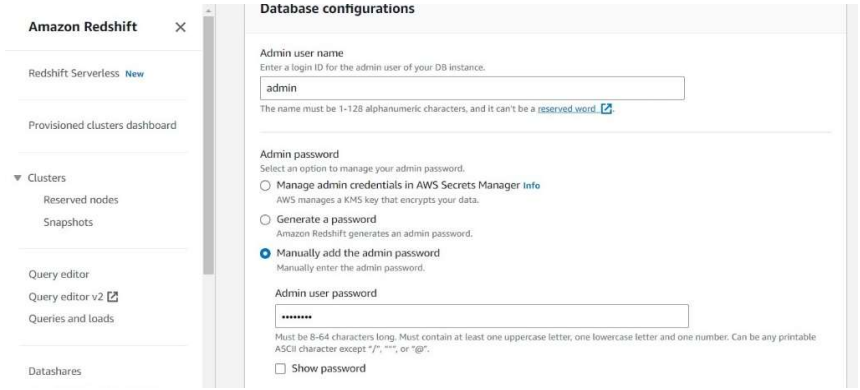
a. Select node type according to the requirement.



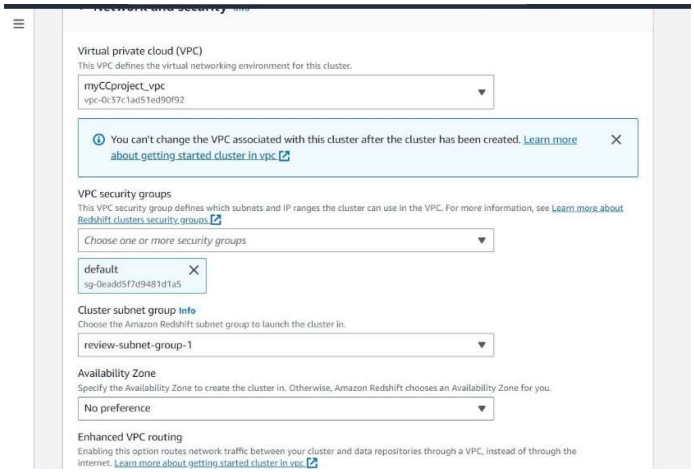
b. Choose default database name as “trips” instead of “dev”



c. Make a note of admin/master user name and password.

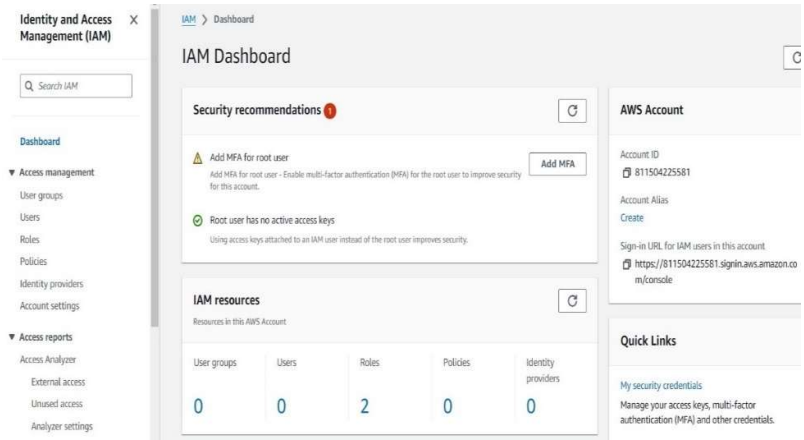


d. Select the default security groups.

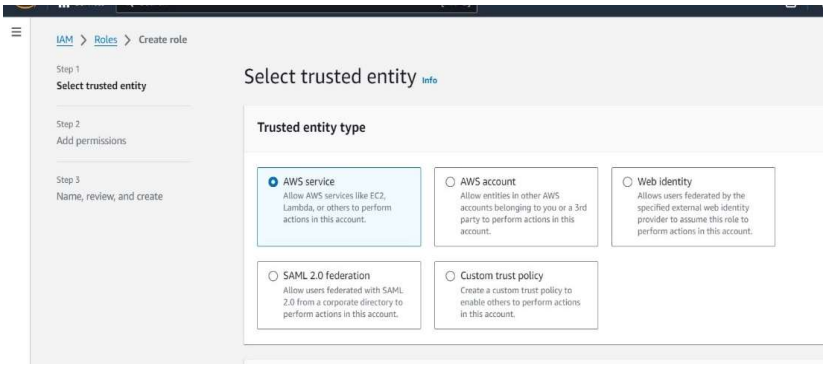


e. Now create and associate IAM roles.

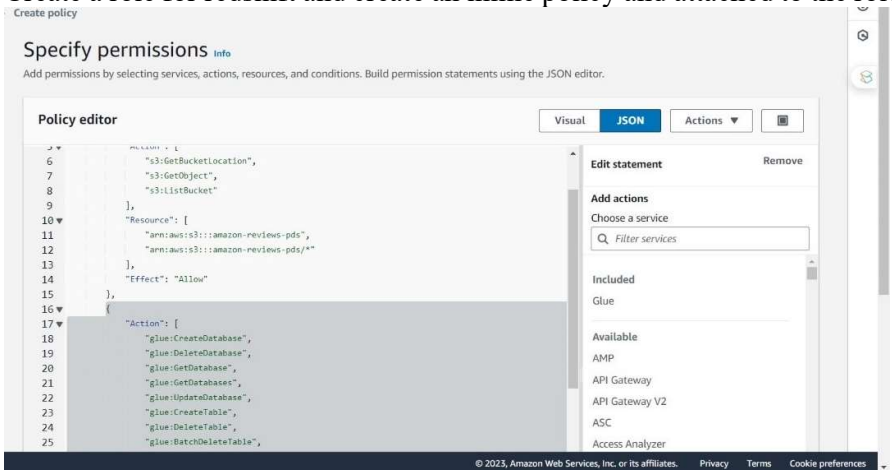
- Now, navigate to IAM dashboard.

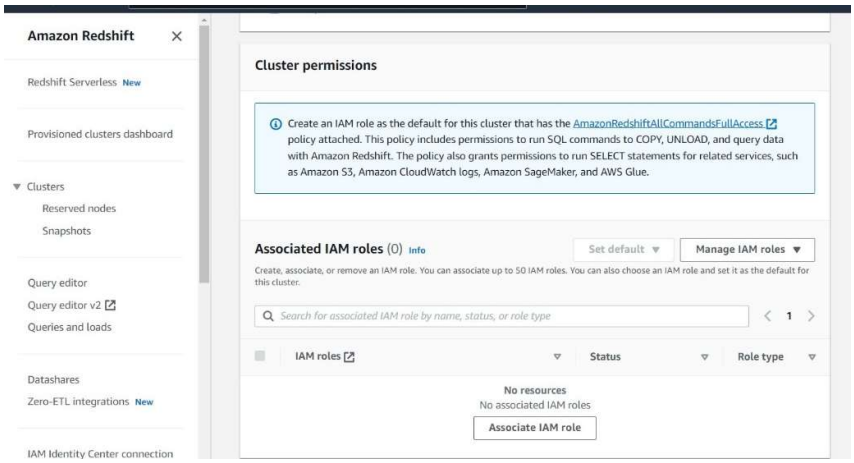


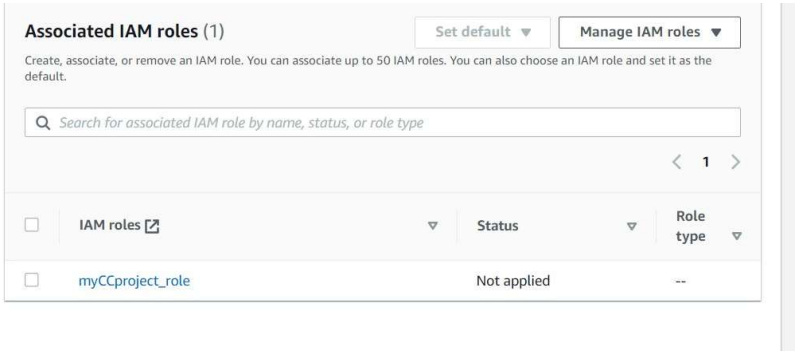
- Select the entity > redshift.



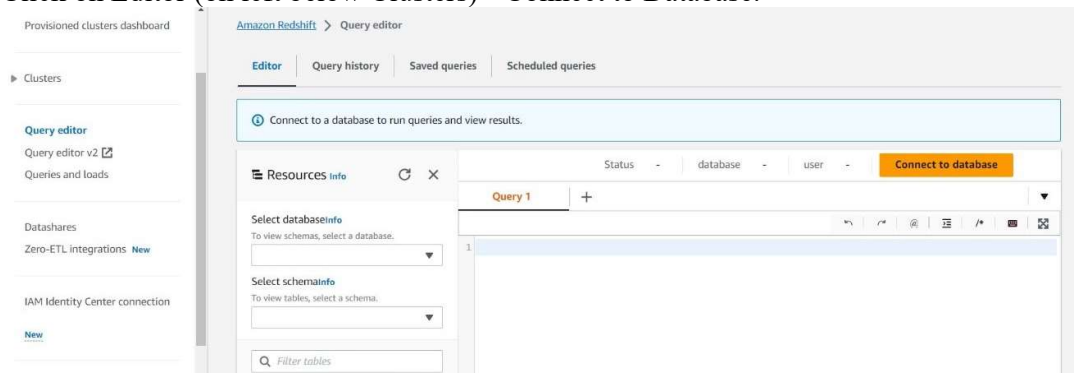
- Create a role for redshift and create an inline policy and attached to the role.



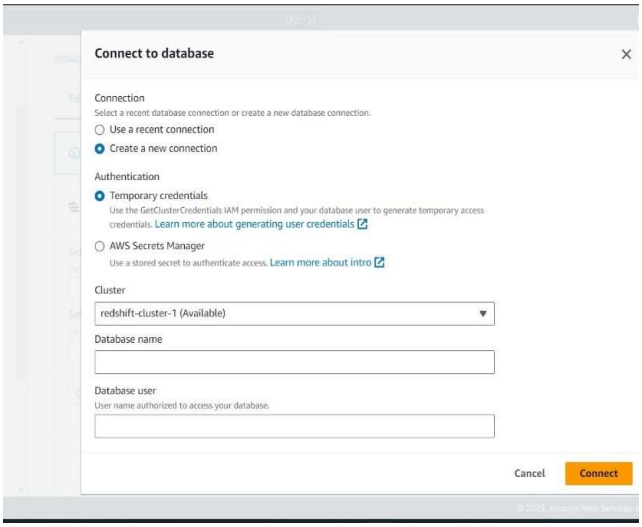




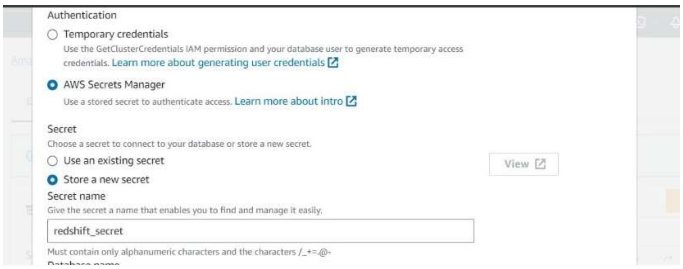
iv. Click on Editor (on left below Clusters)->Connect to Database.



v. Create New Connection->Navigate to Secrets Manager dashboard and create a secret credentials for the redshift connection.



vi. Store the secret in Secrets Manager->key in your cluster details->Give a name for the secret. And give the database name ‘trips’ and type user and password.

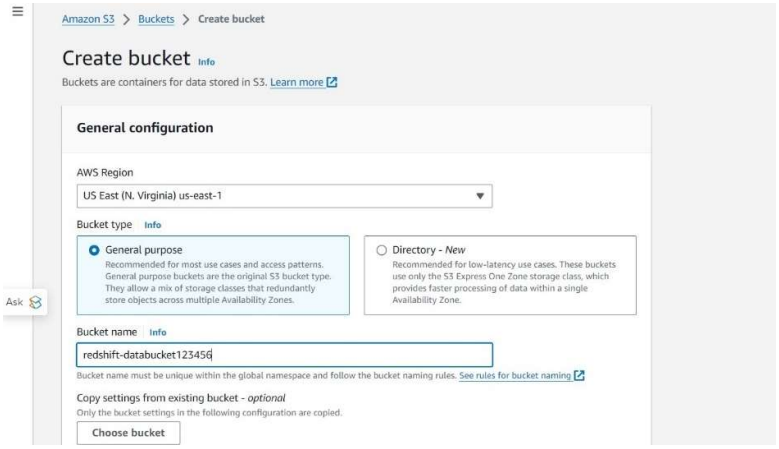


vii. Check the secrets manager for redshift secret. Use this secret as "db_creds" parameter in glue job.

8.3 Create S3 buckets

1. Using Console

   i. Create a new S3 bucket "redshift-databucket<junk numbers> to hold the extract from redshift (this is the source for quicksight).



   ii. Create a new S3 bucket "redshift-scriptbucket<junk numbers> and create 2 folders: python, sql

   iii. Upload following files from my project S3 folder in the same hierarchy:
      - python
        - rs_query.py
        - redshift_module-0.1-py3.6.egg
      - sql
        - tripschema.sql
        - final.sql
        - etl.sql

2. By using CLI:

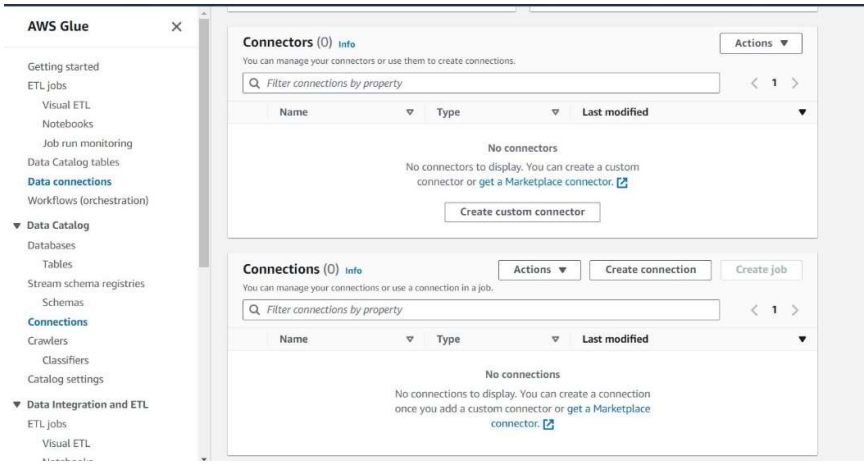
   i. Download MSI CLI installer available in AWS site. https://aws.amazon.com/blogs/developer/aws-cli-v2-installers/

   ii. Open cmd and type aws –version

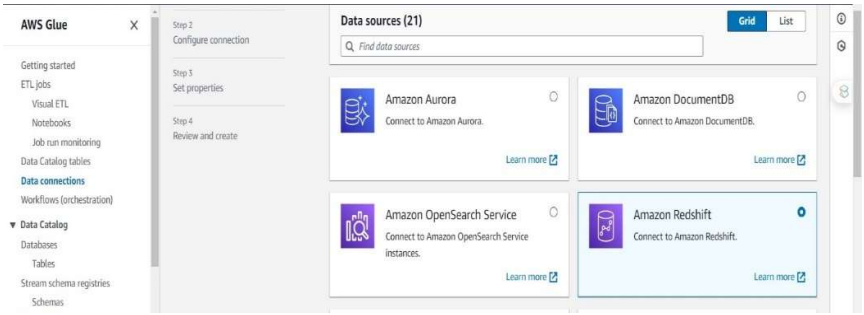
iii. Navigate to IAM dashboard and create user for the CLI and attach AdministratorAccess.
iv. Then in cmd, type aws configure and fill in the credentials.
v. Now create S3 bucket using the command line
aws s3 mb s3://your-unique-bucket-name --region your-preferred-region

8.4 Create Glue job

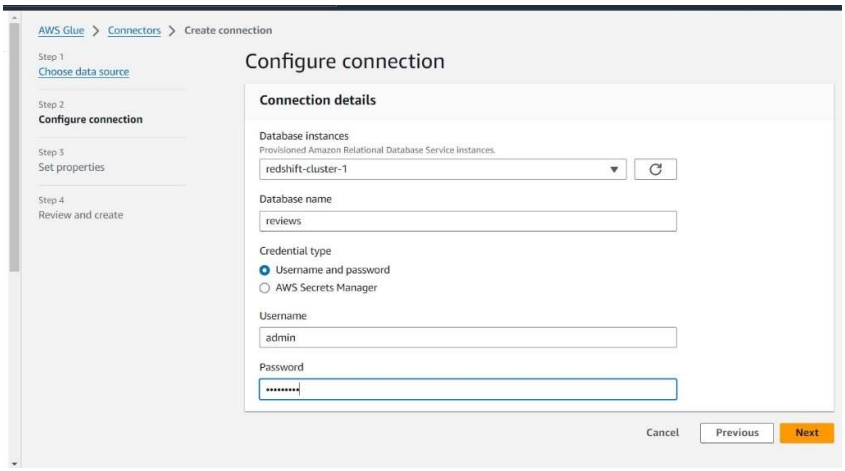
i. First create a connection in Glue Console pointing to "myprojectcluster" and test the connection using "myproject_gluerole" role. Check the policies if the connection fails.



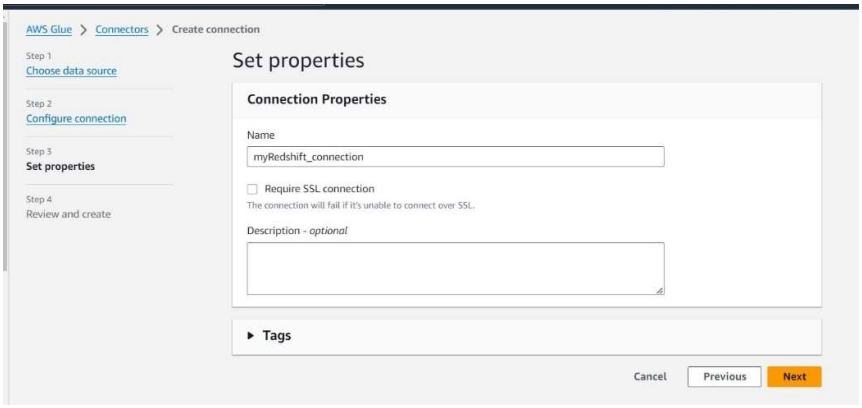
- Select the service required.



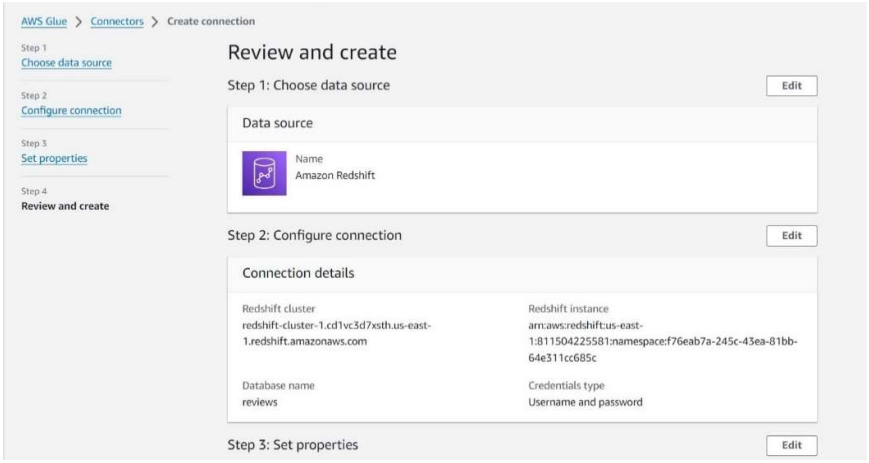
- Configure connection.



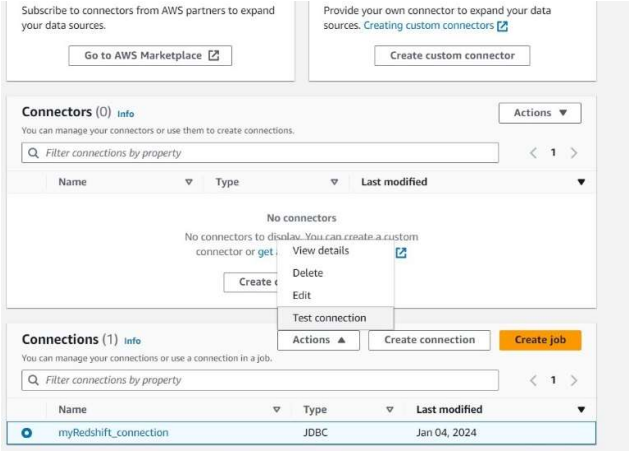
- Set properties.



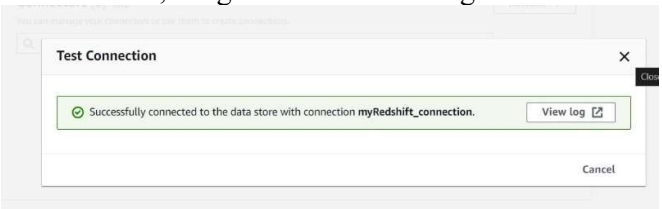
- Review and create.



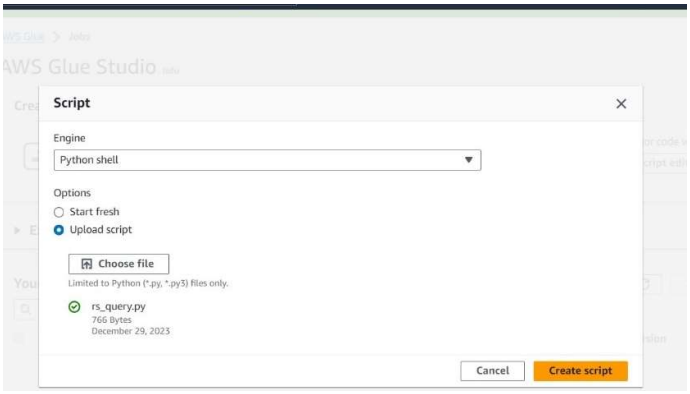
- Test the connection.



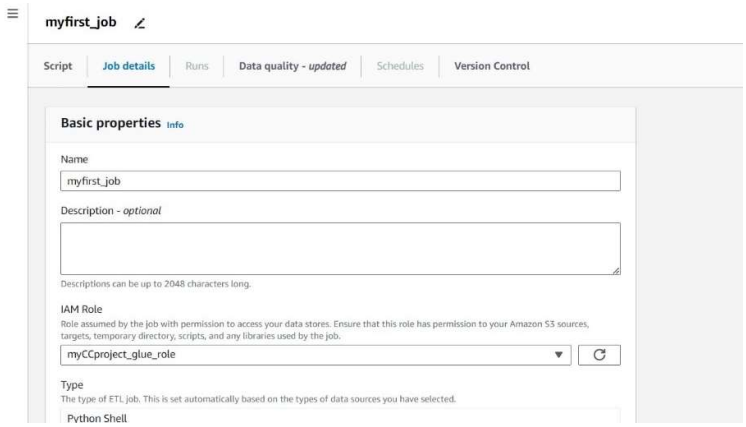
- If it succeeds, we get the below message.



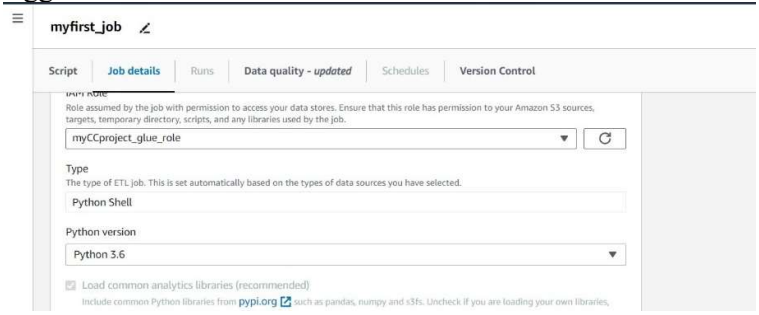
ii. Create a new job using "Python Shell" option. Choose the script from the local machine or create the script.



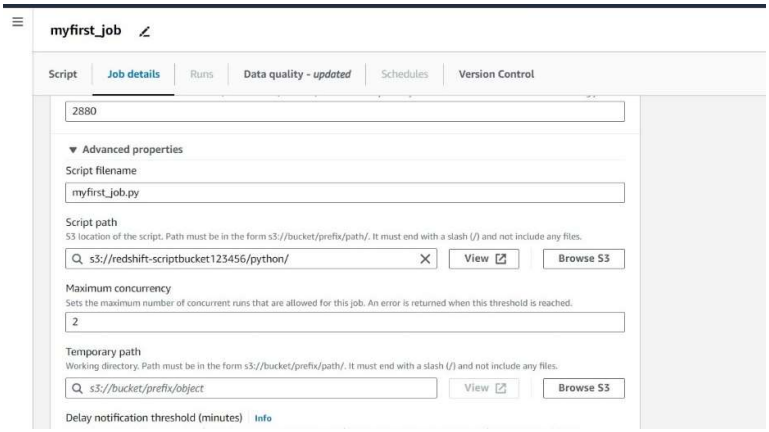
- Go to job details.
- Create glue role from IAM and attach the policy similar to the steps before.
- Select glue role. And attach policy AWSGlueServiceRole and create an inline policy (from IAM folder in my project).



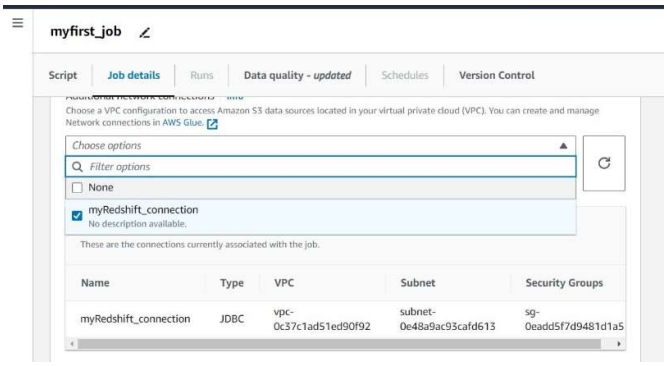
- Select the shell as Python 3.6 (Note: in order to make it compatible with the .egg file later.



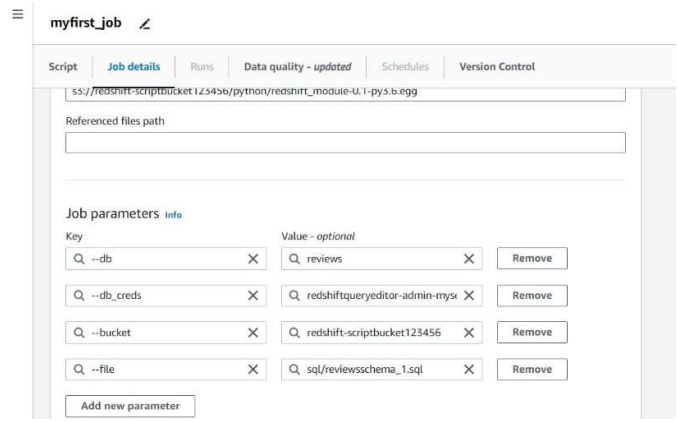
- And choose rs_query.py (from /python) if script not provided and add the path.



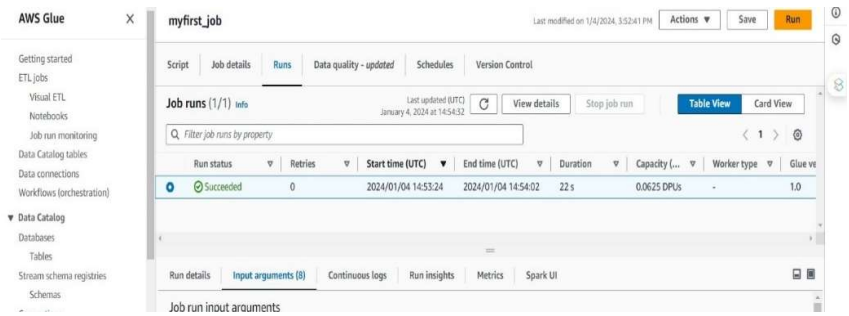
- Select the connection created before.



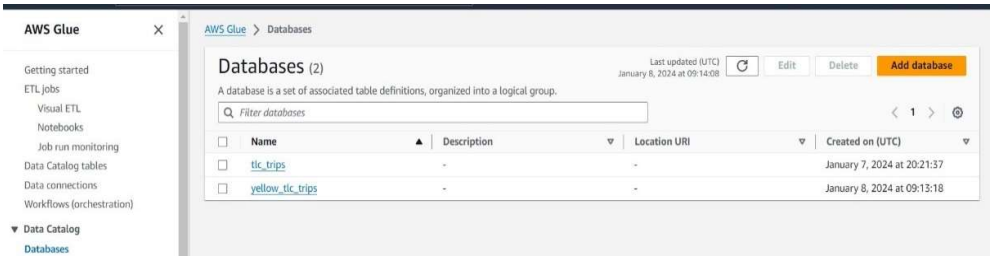
- And give the *.egg file (as python dependency file) and the use the following parameters:
    - --file **sql/tripsschema.sql (from S3)**
    - –db_creds **redshiftqueryeditor-admin-myCCproject_secret (from secrets manager)**
    - --db **trips (database name of Redshift)**
    - –bucket **redshift-scriptbucket<junk numbers> (bucket name S3)**



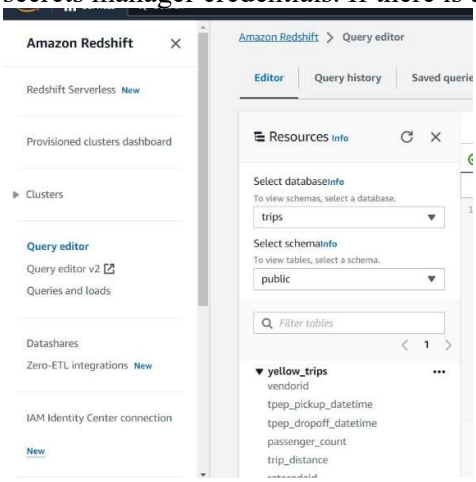
viii. Run the job. And go to the runtime link provided. If it succeeds if will be as below. If it fails, navigate to error logs from runtime and check the error encountered. For any issues. Ensure to re-check the parameters for any naming issue.



ix. Now navigate to Data Catalog > Database, the newly created external schema can be seen.

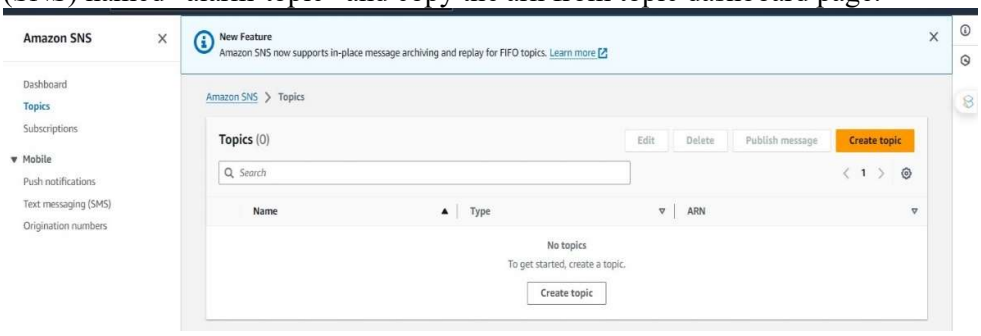


x. Now navigate to Redshift and navigate to query editor and connect the database via the secrets manager credentials. If there is a newly created table, then this is successful.

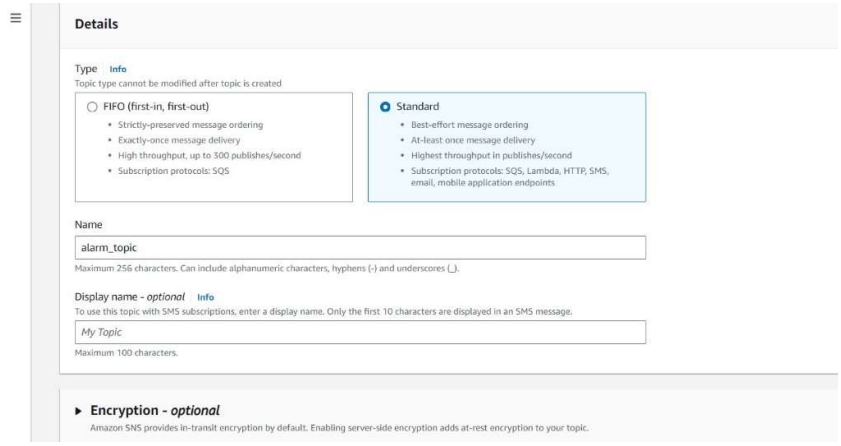


8.5 Create SNS notification

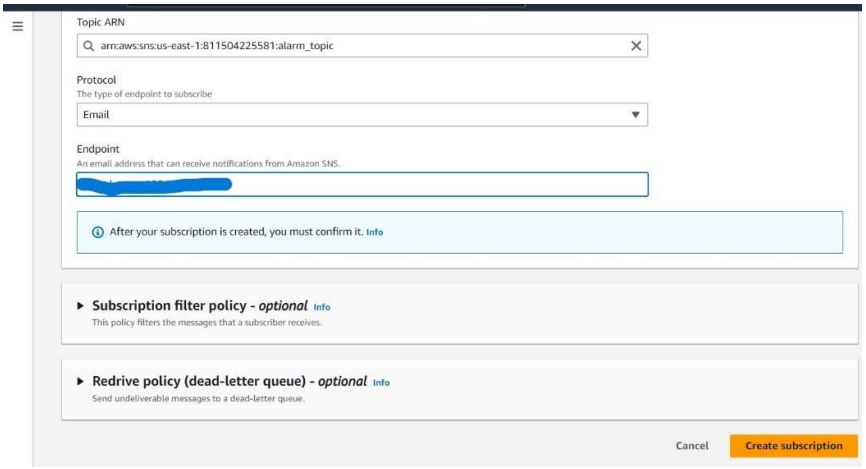
i. Navigate to SNS from services and create a **Standard** Simple Notification Service (SNS) named “alarm-topic” and copy the arn from topic dashboard page.



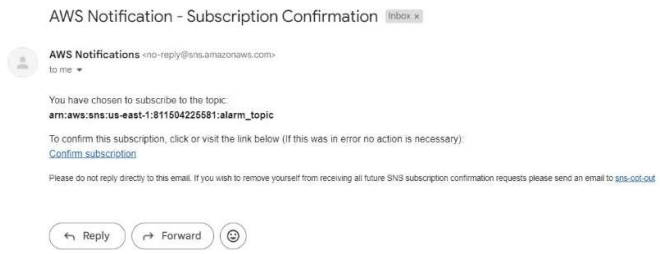
Select standard from details.



ii. Create a subscription to SNS topic by adding your email-id and "EMAIL" as option.

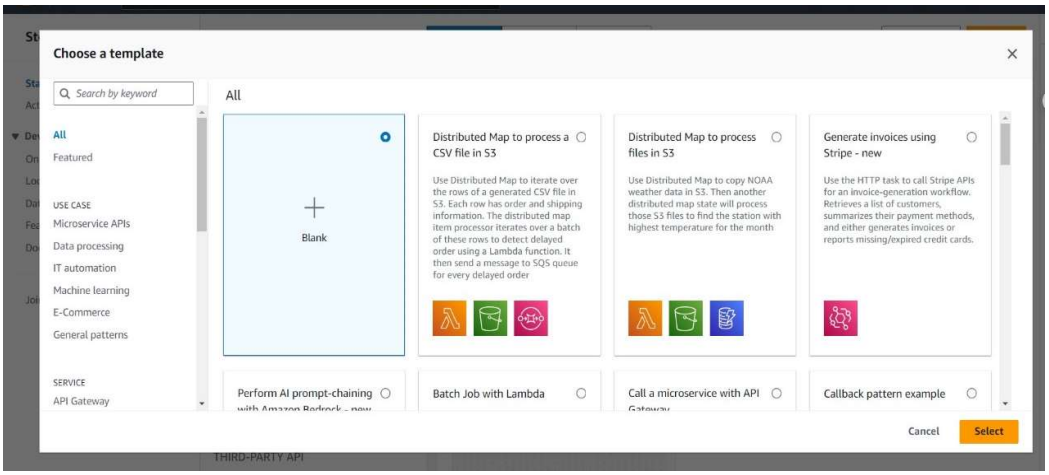


iii. Accept subscription request by logging into your email account. Only then you will start receiving messages.

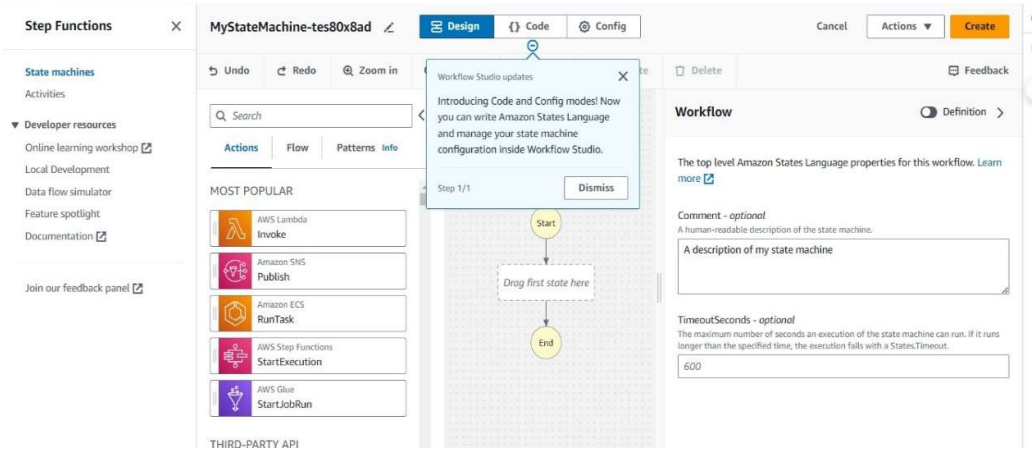


8.6 Create Step Function

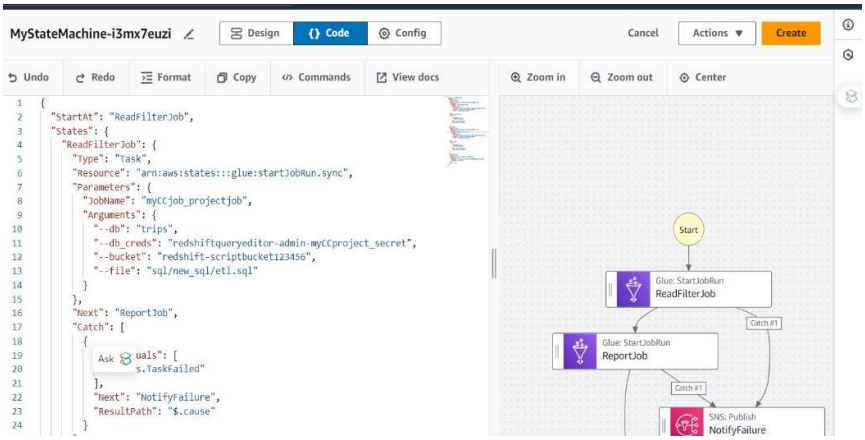
i. Navigate to step function from services and create a state machine.



ii. Go to the Code options and use the json file (from /stepfunction folder).



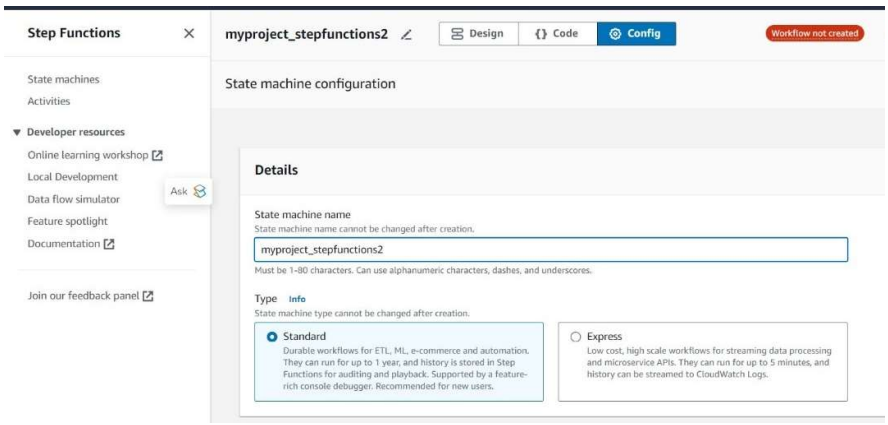
- Update the definition with the parameter values similar to the succeeded Glue job except for “file” parameter.
  - ReadFilterjob” select etl.sql file for inserting into the redshift table.
  - Reportjob: Use the final.sql for unloading the data to the S3 bucket.



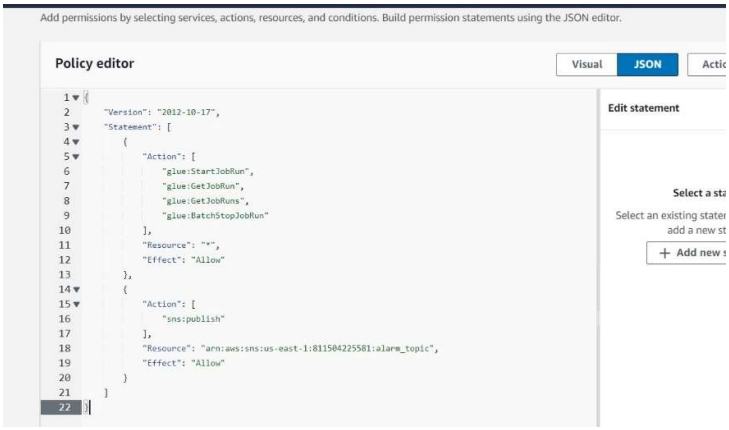
- Update the SNS ARN to your SNS topic ARN.

```
    ]
  },
  "NotifyFailure": {
    "Type": "Task",
    "Resource": "arn:aws:states:::sns:publish",
    "Parameters": {
      "TopicArn": "arn:aws:sns:us-east-1:811504225581:alarm_topic",
      "M                    ": "$.cause"
    },
    "End": true
  }
 }
}
```

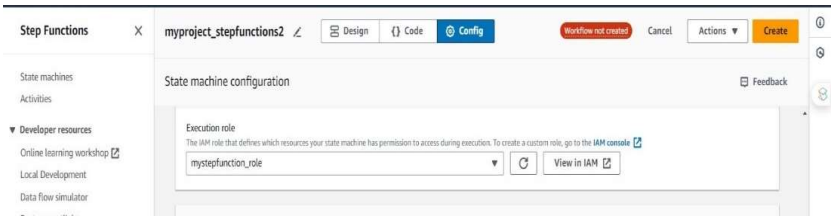
iii. Next, go to the Config option and update the following.



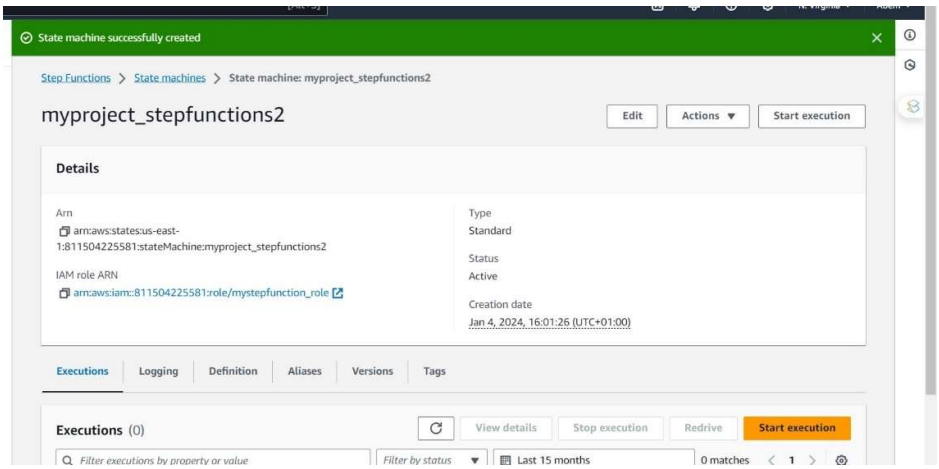
- Create an IAM role for step function similar to the ones done before. Create a policy using the json file.



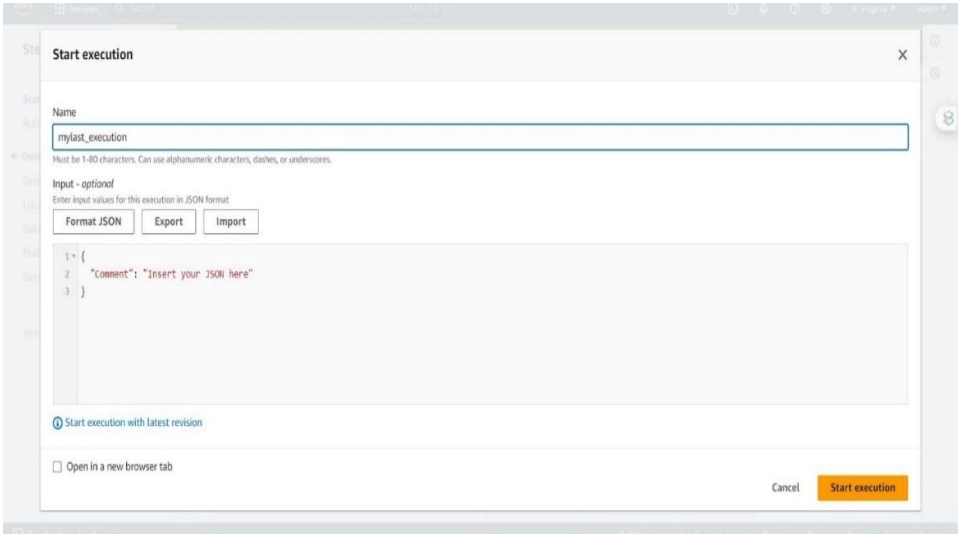
- Attach the “myproject_stepfunctionsrole” role to state machine.



iv. Click “Start Execution” button and check out the job status.

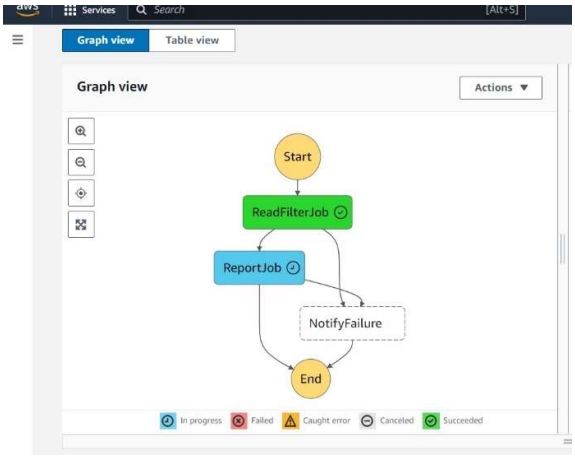


- Name the execution.

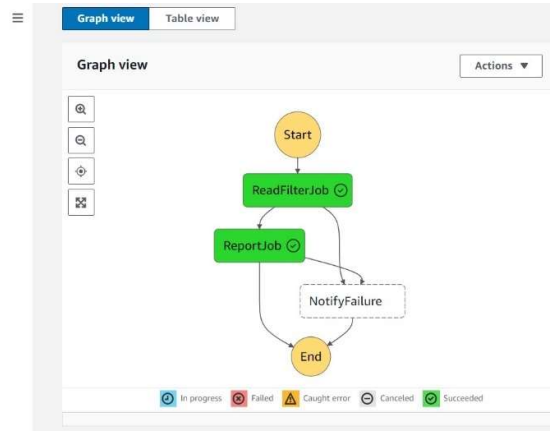


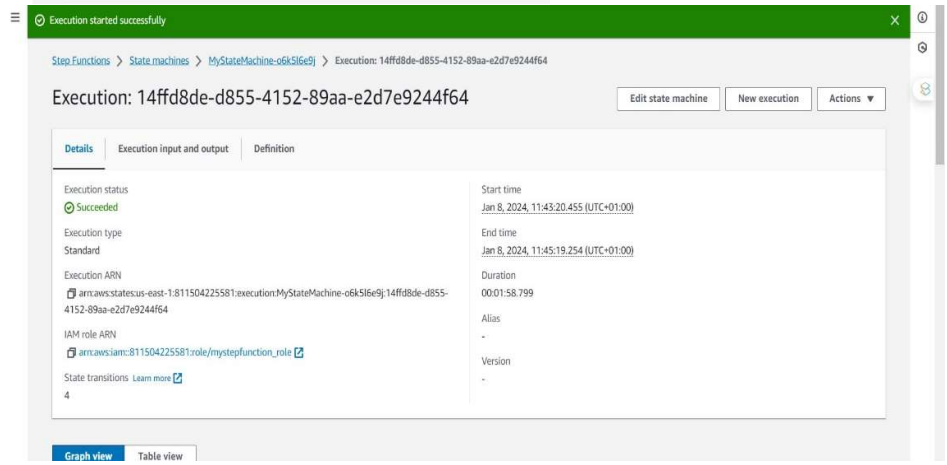
v. Ensure Glue manual run succeeded before coming to this step. Then ensure policy and parameters in the state machine definition is correct as per your environment (Choose “Edit Definition” to alter the JSON file any time you want).

vi. Now from the workflow it can see where it has failed or succeeded. Green and a tick sign shows it has succeeded. If there is warning sign in the job status, then you will get the SNS messages with the error.



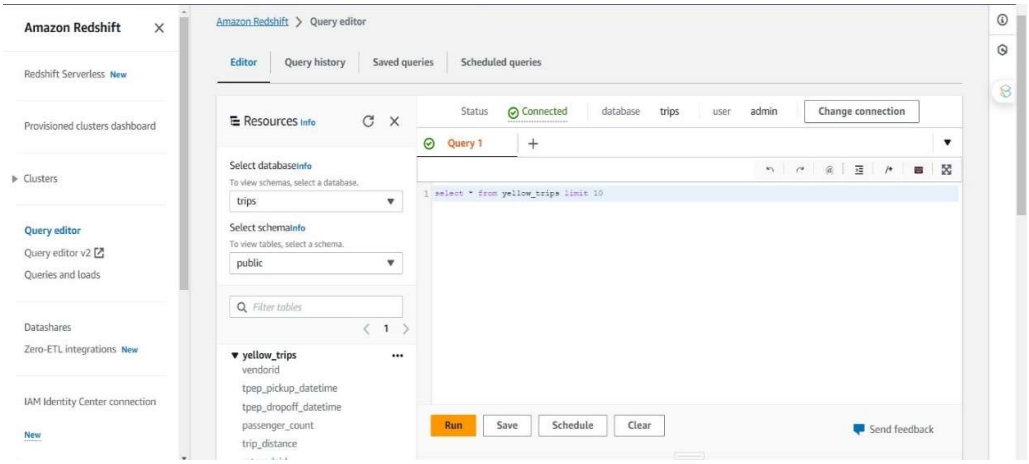
- It can be seen the state machine succeeded.



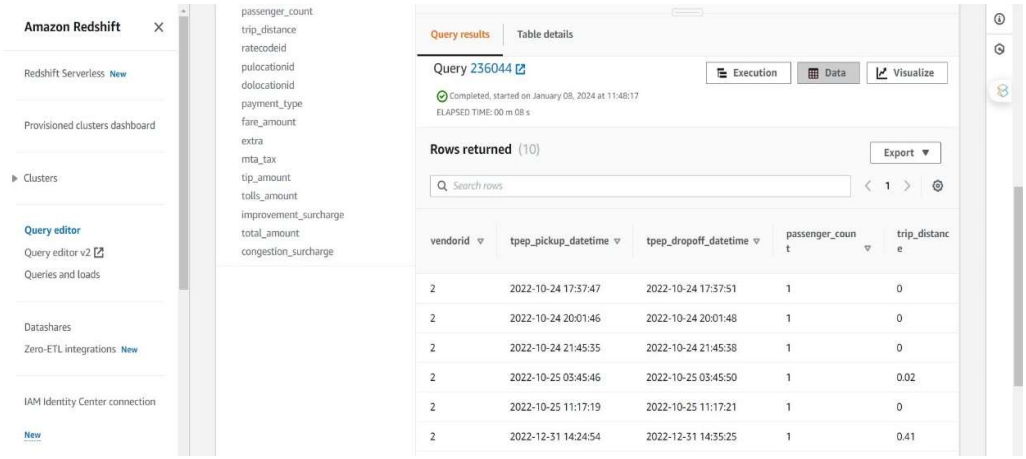


vii. Sometimes, if it succeeds the data is not inserted or loaded to the redshift and the new data bucket. So, it is necessary to check it.

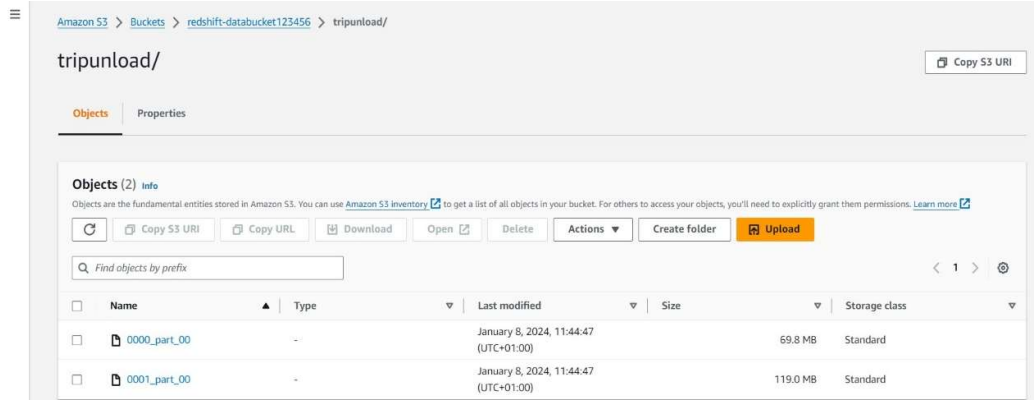
viii. Now navigate to Redshift > Query Editor> Connect to database. Use the secret manager credentials to connect to trips database in the redshift.



ix. And run a query. If the query runs are successful and you can see the data is returned then its successful.

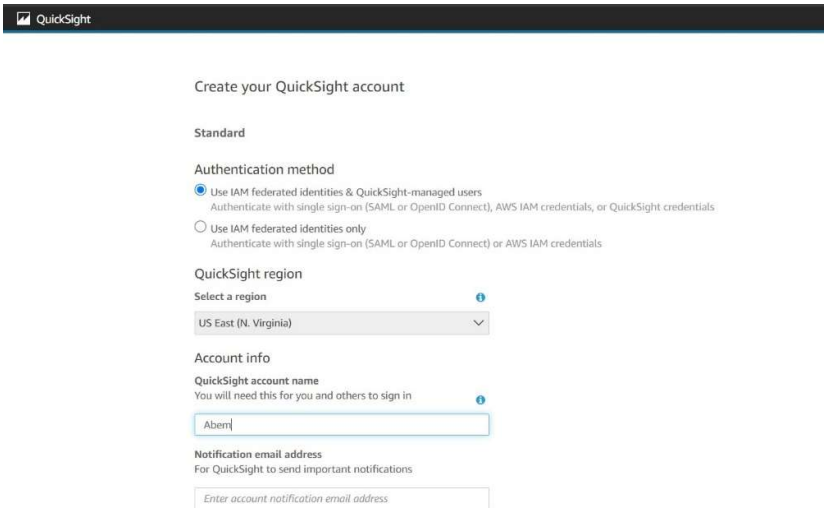


x. Now, check the script bucket where the data is loaded. If loaded successfully, we can see from the size of the data.

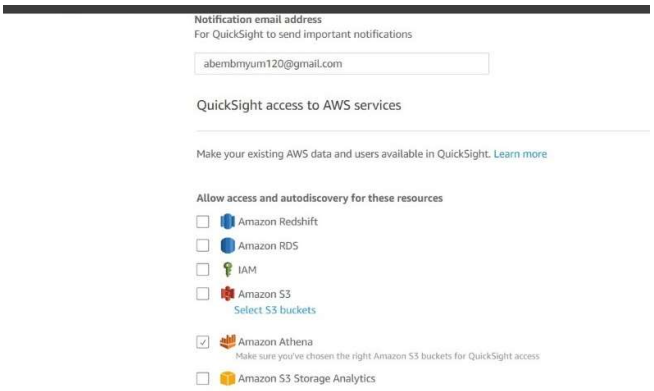


8.7 Create Quicksight

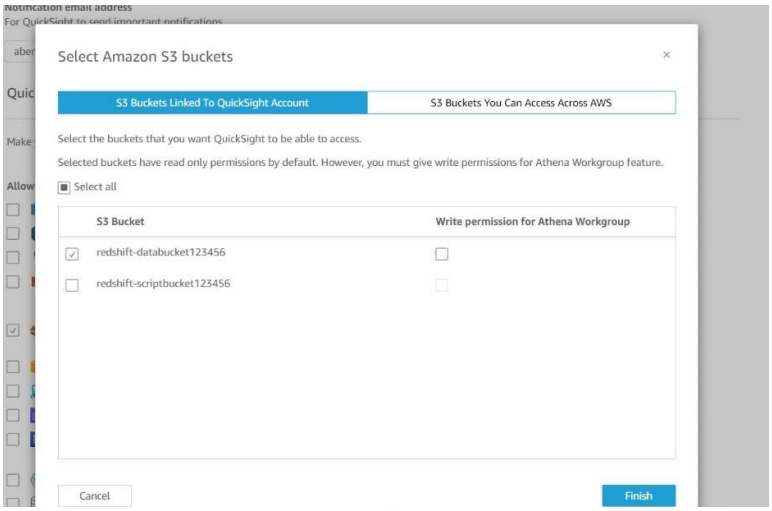
i. Navigate to QuickSight from services and activate Quicksight subscription under "Standard" edition



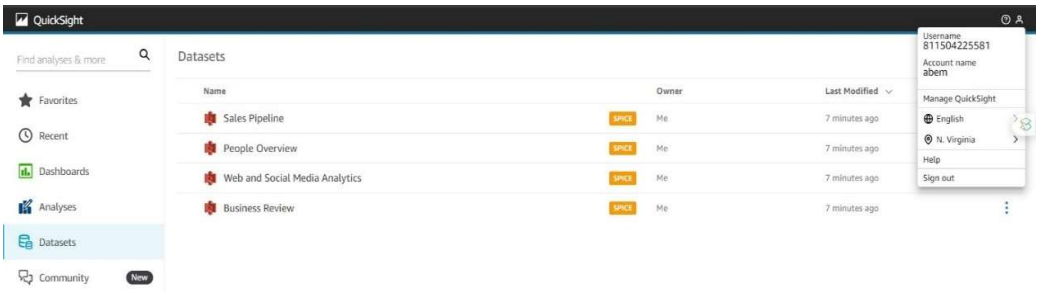
ii. Choose the services that you want to access. Here, S3.



And select the buckets that is required.

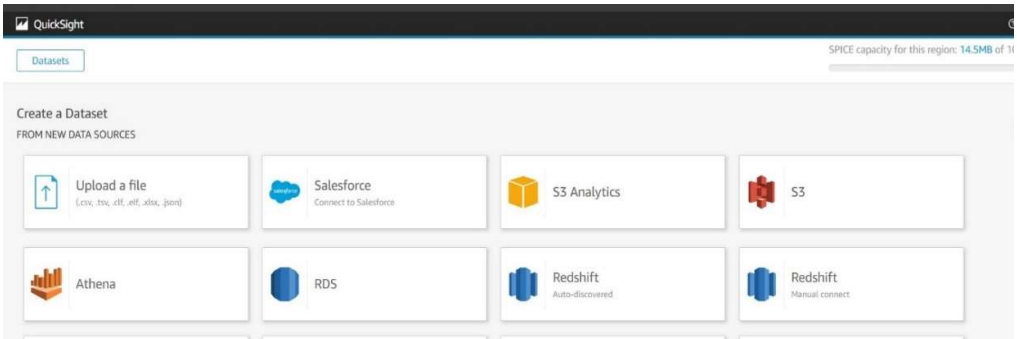


iii. Choose the same region as the S3 account by clicking on right top corner Account name dropdown



iv. Choose Manage Quicksight (right top corner Account name dropdown)-> Choose S3 bucket that you want to connect to if not done before.

v. Click on “Create Dataset” and add “S3” and upload the manifest.json.

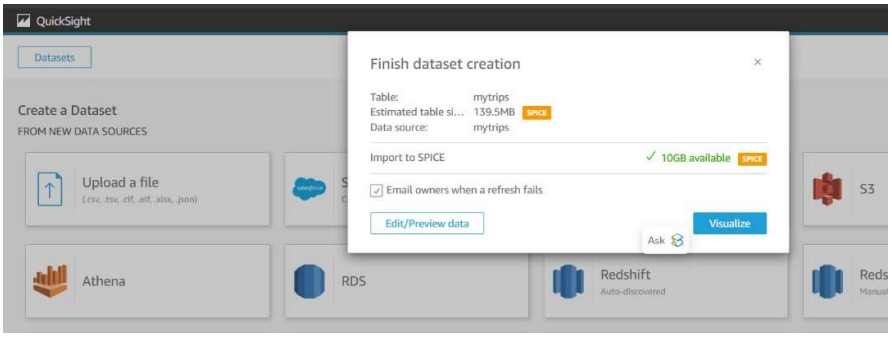


A manifest file is a JSON file that provides instructions on how to access and interpret data stored in S3. It contains metadata about the data files QuickSight should read.

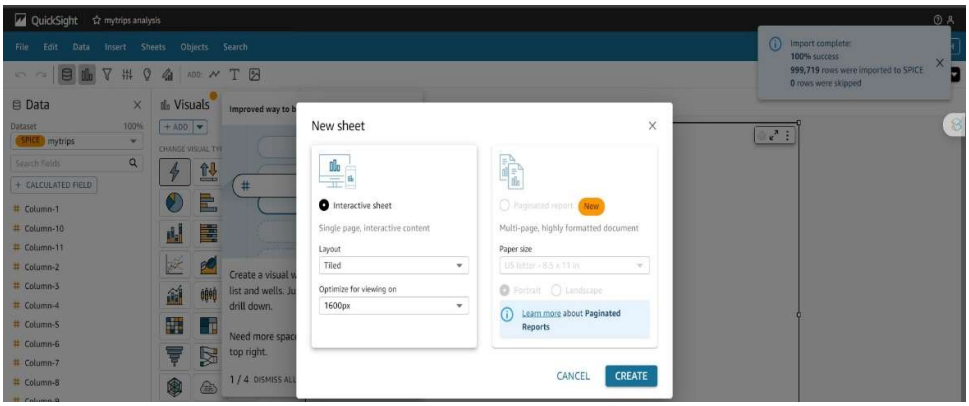
The structure of a manifest file typically includes:

- File Locations: Paths to the specific S3 files or directories containing the data.
- Global Upload Settings: Instructions on how to read the data, such as file formats (CSV, JSON, Parquet), delimiters for CSV files, and data compression information.
- Schema Definition: Although QuickSight can often infer schema from the data files, the manifest file can include schema definitions to resolve ambiguities or errors in schema detection

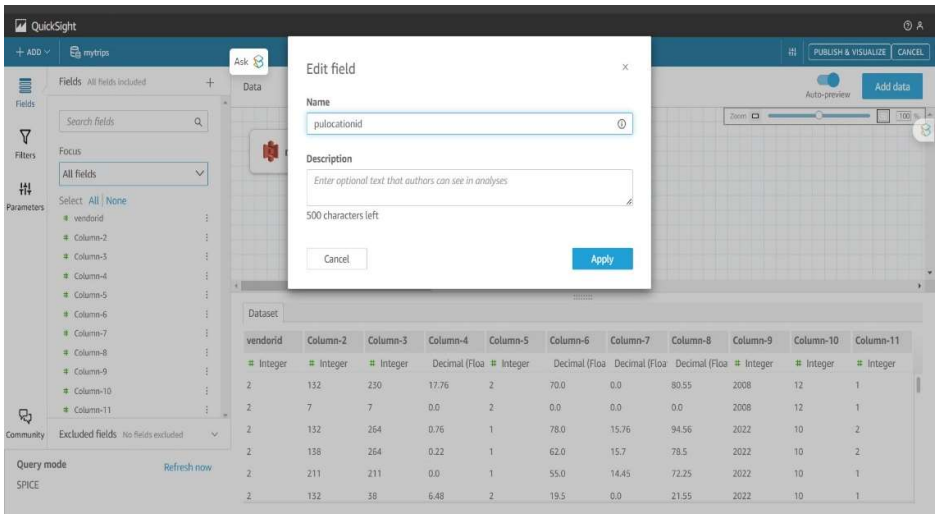
vi. After uploading everything required, data creation is finished and now ready to visualize.



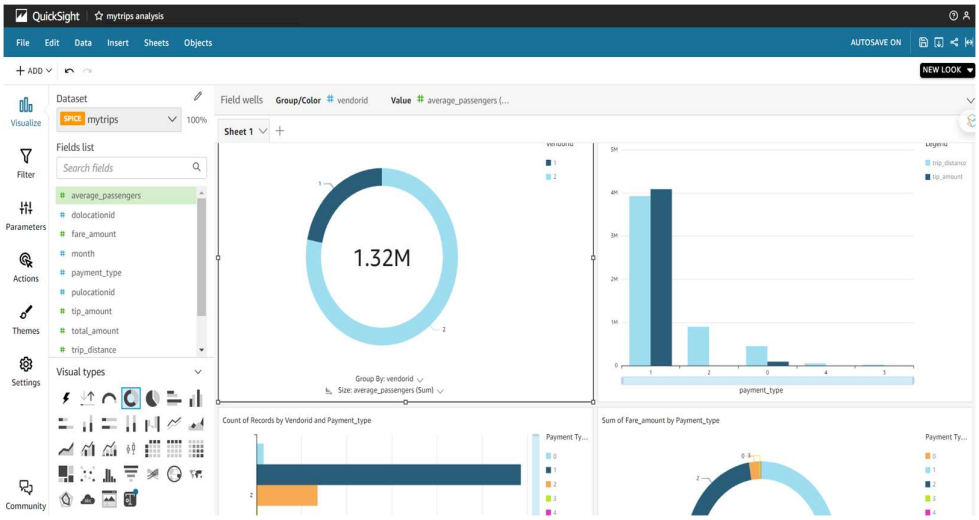
vii. Click on visulaize, a template will be shown.



viii. Rename columns to appropriate ones to avoid confusion as follows:

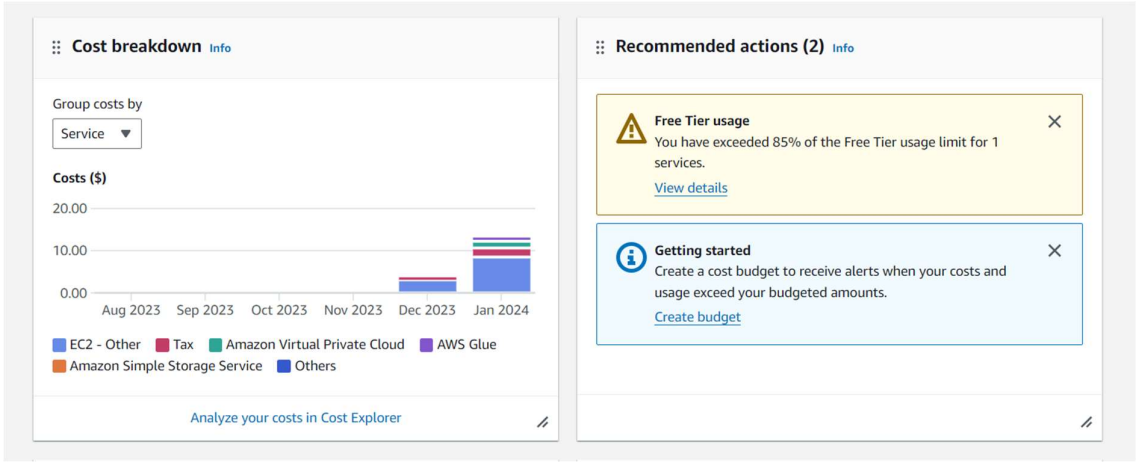


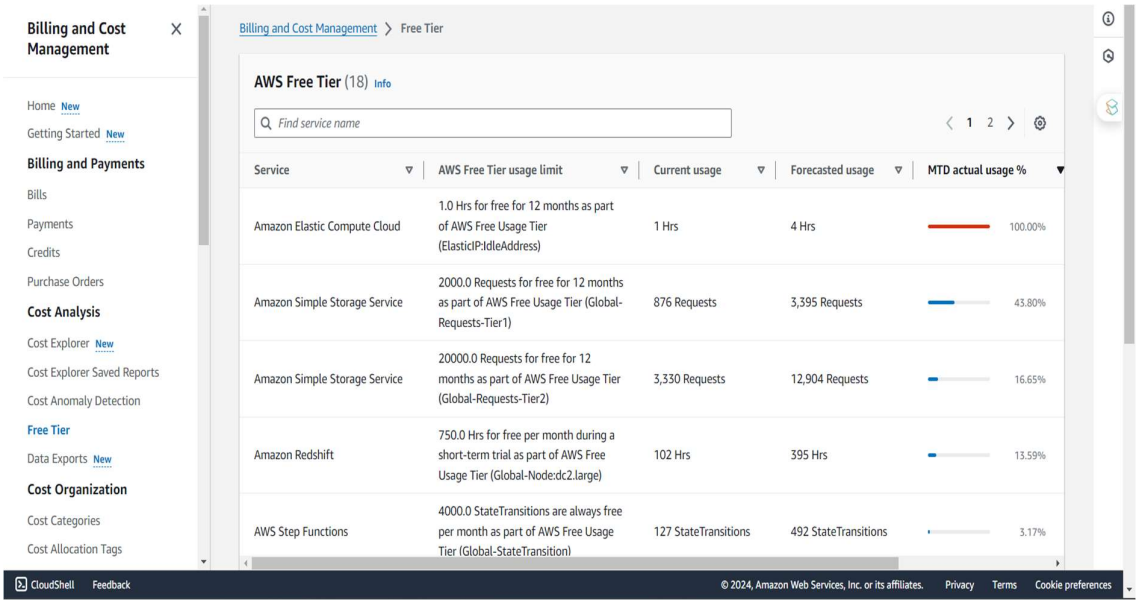
ix. Click on Analysis-> create a new one-> add “Add Visuals” on top left to add more charts and populate it by dragging and dropping corresponding column names.

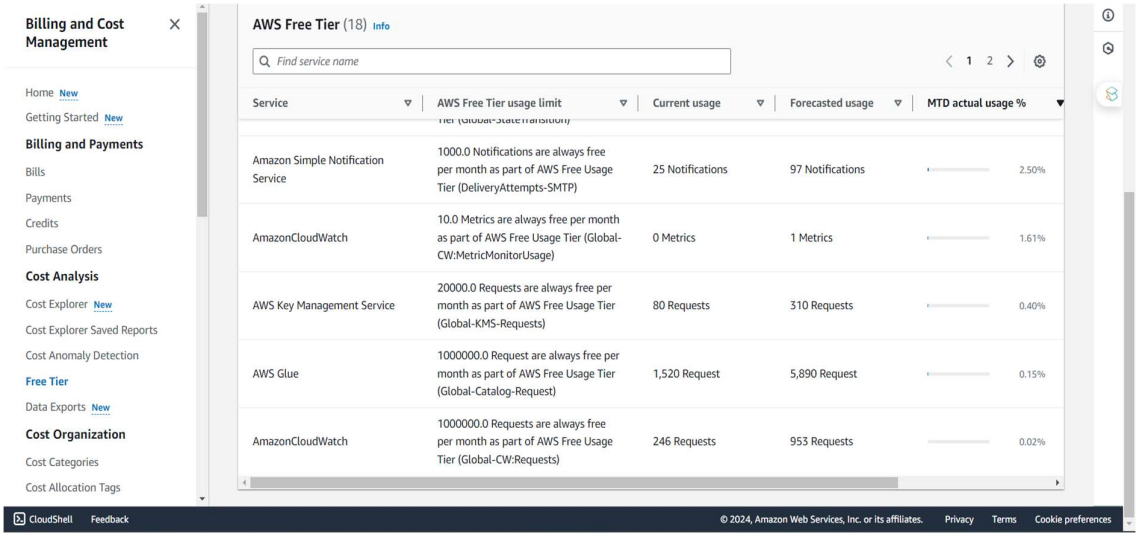


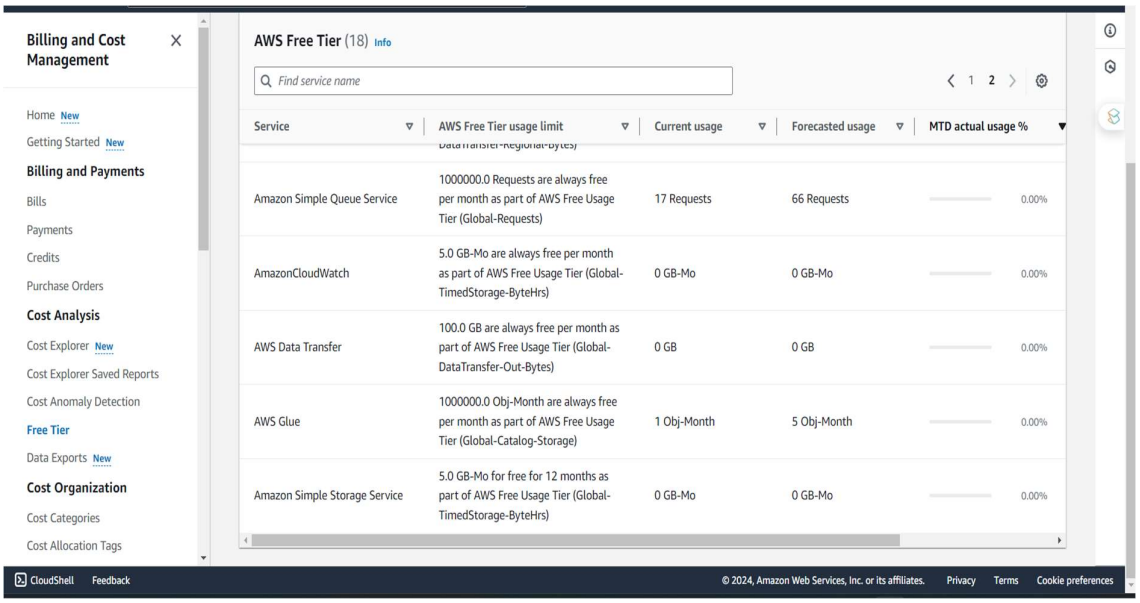
9. Cost estimation

Nearly all the utilized services fell within the free-tier billing structure, with the exception of Elastic IPs, which resulted in higher billing costs.


Cost breakdown Info
Group costs by
Service
Costs ($)
20.00
10.00
0.00
Aug 2023
Sep 2023
Oct 2023
Nov 2023
Dec 2023
Jan 2024
EC2 - Other
Tax
Amazon Virtual Private Cloud
AWS Glue
Amazon Simple Storage Service
Others
Analyze your costs in Cost Explorer
Recommended actions (2) Info
Free Tier usage
You have exceeded 85% of the Free Tier usage limit for 1 services.
View details
Getting started
Create a cost budget to receive alerts when your costs and usage exceed your budgeted amounts.
Create budget


Billing and Cost Management
Home New
Getting Started New
Billing and Payments
Bills
Payments
Credits
Purchase Orders
Cost Analysis
Cost Explorer New
Cost Explorer Saved Reports
Cost Anomaly Detection
Free Tier
Data Exports New
Cost Organization
Cost Categories
Cost Allocation Tags
Billing and Cost Management > Free Tier
AWS Free Tier (18) Info
Find service name
1 2
Service
AWS Free Tier usage limit
Current usage
Forecasted usage
MTD actual usage %
Amazon Elastic Compute Cloud
1.0 Hrs for free for 12 months as part of AWS Free Usage Tier (ElasticIP:IdleAddress)
1 Hrs
4 Hrs
100.00%
Amazon Simple Storage Service
2000.0 Requests for free for 12 months as part of AWS Free Usage Tier (Global-Requests-Tier1)
876 Requests
3,395 Requests
43.80%
Amazon Simple Storage Service
20000.0 Requests for free for 12 months as part of AWS Free Usage Tier (Global-Requests-Tier2)
3,330 Requests
12,904 Requests
16.65%
Amazon Redshift
750.0 Hrs for free per month during a short-term trial as part of AWS Free Usage Tier (Global-Node:dc2.large)
102 Hrs
395 Hrs
13.59%
AWS Step Functions
4000.0 StateTransitions are always free per month as part of AWS Free Usage Tier (Global-StateTransition)
127 StateTransitions
492 StateTransitions
3.17%
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Billing and Cost Management
Home New
Getting Started New
Billing and Payments
Bills
Payments
Credits
Purchase Orders
Cost Analysis
Cost Explorer New
Cost Explorer Saved Reports
Cost Anomaly Detection
Free Tier
Data Exports New
Cost Organization
Cost Categories
Cost Allocation Tags
AWS Free Tier (18) Info
Find service name
1 2
Service
AWS Free Tier usage limit
Current usage
Forecasted usage
MTD actual usage %
Amazon Simple Notification Service
1000.0 Notifications are always free per month as part of AWS Free Usage Tier (DeliveryAttempts-SMTP)
25 Notifications
97 Notifications
2.50%
AmazonCloudWatch
10.0 Metrics are always free per month as part of AWS Free Usage Tier (Global-CW:MetricMonitorUsage)
0 Metrics
1 Metrics
1.61%
AWS Key Management Service
20000.0 Requests are always free per month as part of AWS Free Usage Tier (Global-KMS-Requests)
80 Requests
310 Requests
0.40%
AWS Glue
1000000.0 Request are always free per month as part of AWS Free Usage Tier (Global-Catalog-Request)
1,520 Request
5,890 Request
0.15%
AmazonCloudWatch
1000000.0 Requests are always free per month as part of AWS Free Usage Tier (Global-CW:Requests)
246 Requests
953 Requests
0.02%
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences



10. Conclusion

In summary, the project effectively orchestrated ETL workflows for Amazon Redshift using AWS Step Functions and AWS Glue.

Key highlights include:

- **Efficient Data Transformation**: AWS Glue streamlined data extraction and transformation tasks, enhancing efficiency and data quality.
- **Workflow Orchestration**: AWS Step Functions provided a seamless orchestration of Glue jobs, ensuring a reliable and scalable ETL workflow.
- **Redshift Data Warehousing**: Leveraging Amazon Redshift as a scalable data warehouse for analytics, offering optimized query performance.
- **S3 for Staging**: Amazon S3 served as a versatile staging area for raw and processed data, ensuring data consistency and integrity.
- **QuickSight Visualization**: Integration with QuickSight enabled the visualization and analysis of Redshift data, creating insightful dashboards.

This project illustrates the power of AWS services in handling large-scale ETL tasks, providing a foundation for scalable and resilient data processing workflows.

11. References

i. "New York City Taxi and Limousine Commission (TLC) Trip Record Data," Registry of Open Data on AWS. [Online]. Available: https://registry.opendata.aws/nyc-tlc-trip-records-pds.

ii. "Orchestrate Amazon Redshift-based ETL Workflows with AWS Step Functions and AWS Glue," Amazon Web Services, Inc. [Online]. Available: https://aws.amazon.com/blogs/big-data/orchestrate-amazon-redshift-based-etl-workflows-with-aws-step-functions-and-aws-glue/.

iii. "AWS Glue Developer Guide," Amazon Web Services, Inc. [Online]. Available: https://docs.aws.amazon.com/glue/latest/dg/add-job-python.html.

iv. "Identity and Access Management (IAM) User Guide," Amazon Web Services, Inc. [Online]. Available: https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles.html.

v. "Getting Started with Amazon Redshift Spectrum," Amazon Web Services, Inc. [Online]. Available: https://docs.aws.amazon.com/redshift/latest/dg/c-getting-started-using-spectrum.html.

vi. "Creating Python Libraries in an Egg Format," AWS Glue Developer Guide, Amazon Web Services, [Online]. Available: https://docs.aws.amazon.com/glue/latest/dg/add-job-python.html#create-python-egg-library.

vii. "Creating External Schemas for Amazon Redshift Spectrum," Amazon Redshift Database Developer Guide, Amazon Web Services, [Online]. Available: https://docs.aws.amazon.com/redshift/latest/dg/c-spectrum-external-schemas.html.

viii. "Defining External Tables in Amazon Redshift Spectrum," Amazon Redshift Database Developer Guide, Amazon Web Services, [Online]. Available: https://docs.aws.amazon.com/redshift/latest/dg/c-spectrum-external-tables.html.

ix. "ALTER TABLE for External Tables," Amazon Redshift Database Developer Guide, Amazon Web Services, [Online]. Available: https://docs.aws.amazon.com/redshift/latest/dg/r_ALTER_TABLE_external-table.html.

x. "Creating Tables," Amazon Redshift Database Developer Guide, Amazon Web Services, [Online]. Available: https://docs.aws.amazon.com/redshift/latest/dg/t_Creating_tables.html.

xi. "Creating an Activity State Machine," AWS Step Functions Developer Guide, Amazon Web Services, [Online]. Available: https://docs.aws.amazon.com/step-functions/latest/dg/tutorial-creating-activity-state-machine.html.

xii. Amazon Web Services, Inc., "Creating a VPC," Amazon Virtual Private Cloud User Guide, [Online]. Available: https://docs.aws.amazon.com/vpc/latest/userguide/create-vpc.html.