AWS S3 to RedShift Data Transfer using AWS Lambda Function Triggered by S3 Events: -

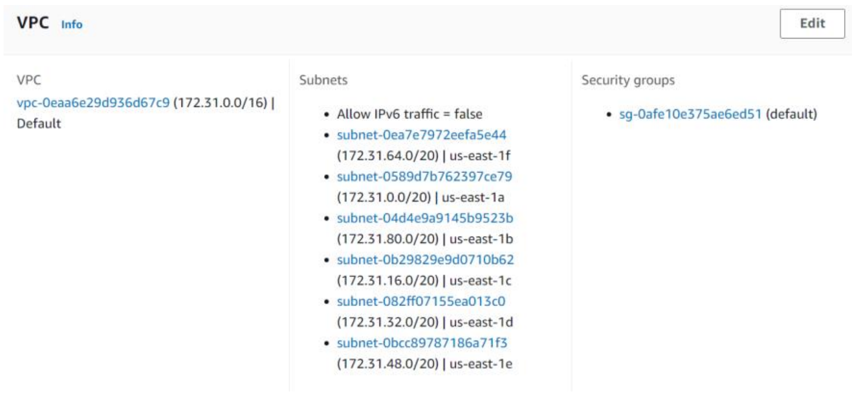
1. Creating a Redshift Cluster

Steps:

1. **Sign in to the AWS Management Console**.

2. **Navigate to Amazon Redshift**.

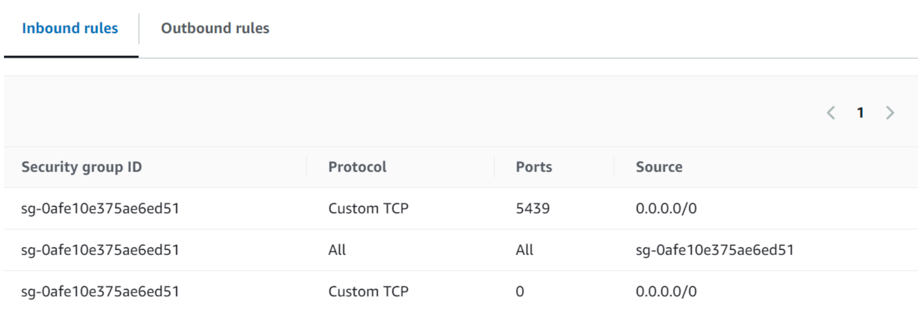
3. **Create Cluster**:
    a. Click on "Create cluster".
    b. Configure cluster settings (e.g., cluster identifier, node type, number of nodes).
    c. Choose database settings (e.g., database name, master username, and password).
    d. Set up VPC and security groups in AWS Lambda as well as RedShift Cluster (for connecting to tableau)
        i. VPC Connection - (Create new inline policies **AllowCreateNetwork**, **DescribeNetwork**, **DeleteNetwork**) in lambda connection role in IAM (for e.g. - **lambda-demo-boto3**)
        ii. Security Group – Edit Inbound Rule to add RedShift (TCP) for Tableau.



VPC
vpc-0eaa6e29d936d67c9 (172.31.0.0/16) | Default

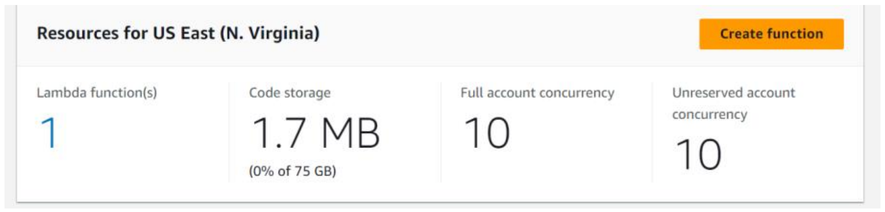
Subnets
- Allow IPv6 traffic = false
- subnet-0ea7e7972eefa5e44 (172.31.64.0/20) | us-east-1f
- subnet-0589d7b762397ce79 (172.31.0.0/20) | us-east-1a
- subnet-04d4e9a9145b9523b (172.31.80.0/20) | us-east-1b
- subnet-0b29829e9d0710b62 (172.31.16.0/20) | us-east-1c
- subnet-082ff07155ea013c0 (172.31.32.0/20) | us-east-1d
- subnet-0bcc89787186a71f3 (172.31.48.0/20) | us-east-1e

Security groups
- sg-0afe10e375ae6ed51 (default)



Security group ID	Protocol	Ports	Source
sg-0afe10e375ae6ed51	Custom TCP	5439	0.0.0.0/0
sg-0afe10e375ae6ed51	All	All	sg-0afe10e375ae6ed51
sg-0afe10e375ae6ed51	Custom TCP	0	0.0.0.0/0

4. **Launch the Cluster** and wait for it to be available.



Resources for US East (N. Virginia)

Lambda function(s)	Code storage	Full account concurrency	Unreserved account concurrency
1	1.7 MB (0% of 75 GB)	10	10

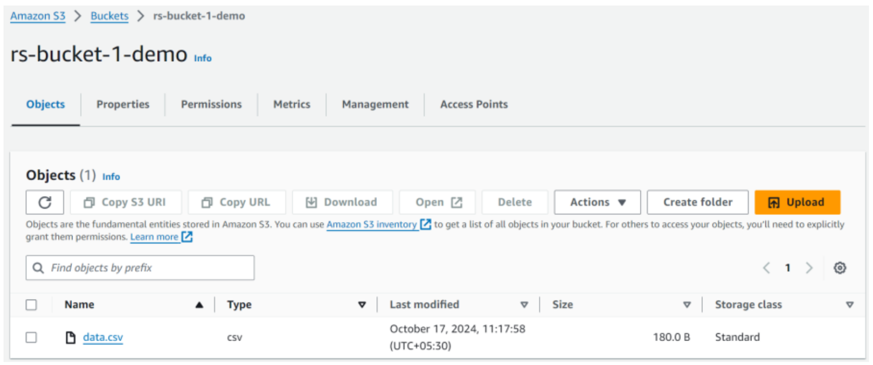
2. Creating an S3 Bucket

Steps:

1. **Navigate to Amazon S3** in the AWS Management Console.

2. **Create Bucket**:
   a. Click on "Create bucket".
   b. Provide a unique name and select the appropriate region.
   c. Configure options (e.g., versioning, encryption) as needed (optional/default).

3. **Set Permissions**:
   a. Make sure the bucket policy allows access from Redshift and Lambda.



Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
data.csv	csv	October 17, 2024, 11:17:58 (UTC+05:30)	180.0 B	Standard

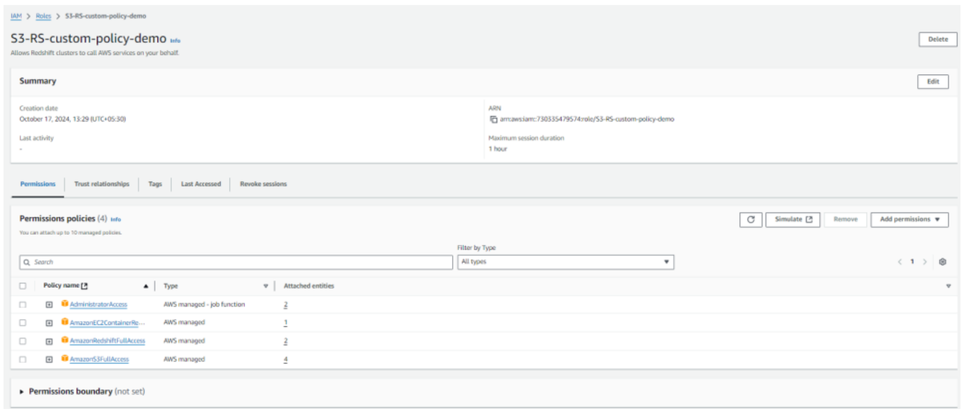
3. Setting Up IAM Roles

Roles Needed:

1. **IAM Role for Redshift**:
   a. Allows Redshift to access the S3 bucket.
   b. Attach the AmazonS3ReadOnlyAccess policy or create a custom policy.

2. **IAM Role for Lambda**:
   a. Allows Lambda to execute and access necessary resources (S3, Redshift).
   b. Attach the following policies:
      i. AWSLambdaBasicExecutionRole
      ii. AmazonS3ReadOnlyAccess
      iii. Custom policy for Redshift access.
         1. S3-RS-custom-policy-demo(create Redshift policy from IAM and attach other policies like)
            a. AmazonS3FullAccess
            b. AmazonRedshiftFullAccess
            c. AmazonEC2ContainerRegistry
            d. AdministratorAccess

Note: This policy (Redshift) with attached policies must be associated with the policy used in the **RedShift Cluster** and use the same **ARN** of this policy in the lambda function IAM used



IAM > Roles > S3-RS-custom-policy-demo

S3-RS-custom-policy-demo Info

Allows Redshift clusters to call AWS services on your behalf.

Summary

Creation date
October 17, 2024, 13:29 (UTC+05:30)

Last activity
-

ARN
arn:aws:iam::730335479574:role/S3-RS-custom-policy-demo

Maximum session duration
1 hour

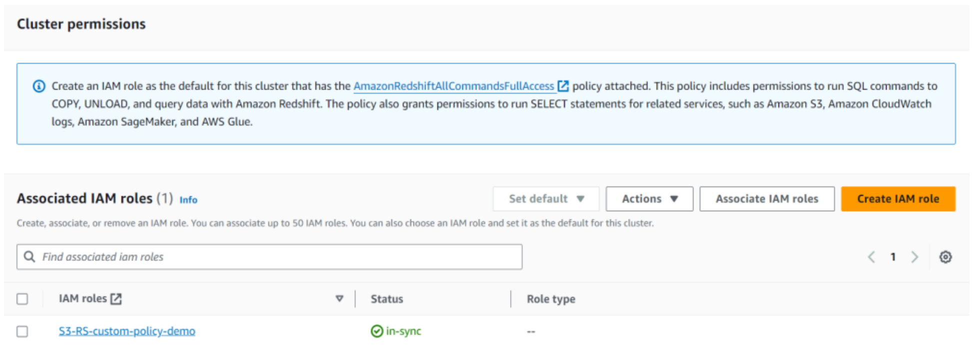
Permissions policies (4) Info

You can attach up to 10 managed policies.

Filter by Type: All types

Policy name	Type	Attached entities
AdministratorAccess	AWS managed - job function	2
AmazonEC2ContainerRe...	AWS managed	1
AmazonRedshiftFullAccess	AWS managed	2
AmazonS3FullAccess	AWS managed	4

Permissions boundary (not set)


Cluster permissions

Create an IAM role as the default for this cluster that has the AmazonRedshiftAllCommandsFullAccess policy attached. This policy includes permissions to run SQL commands to COPY, UNLOAD, and query data with Amazon Redshift. The policy also grants permissions to run SELECT statements for related services, such as Amazon S3, Amazon CloudWatch logs, Amazon SageMaker, and AWS Glue.

Associated IAM roles (1) Info

Create, associate, or remove an IAM role. You can associate up to 50 IAM roles. You can also choose an IAM role and set it as the default for this cluster.

IAM roles	Status	Role type
S3-RS-custom-policy-demo	in-sync	--

Steps:

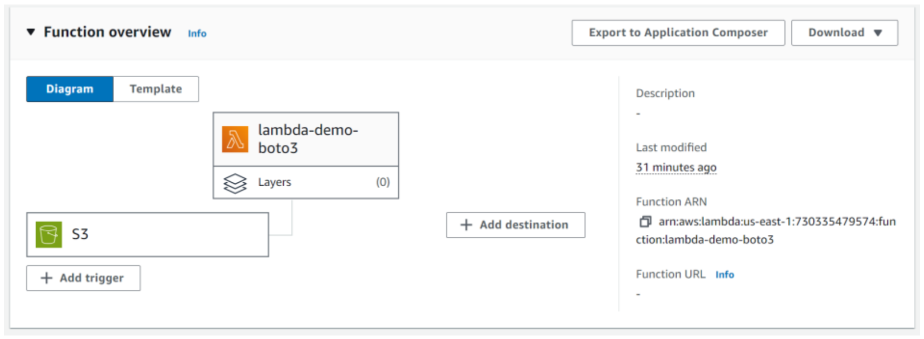
1. **Navigate to IAM** in the AWS Management Console.

2. **Create a Role** for each service:
    a. Select the service (Redshift).
    b. Attach the necessary policies (S3, redshift, admin, ec2).
    c. Note the Role **ARN** for later use in lambda function.

4. Creating the Lambda Function

Steps:

1. **Navigate to AWS Lambda** in the AWS Management Console.
2. **Create Function**:
    a. Click on "Create function".
    b. Choose "Author from scratch".
    c. Configure basic settings (function name, runtime, and role).


Function overview Info

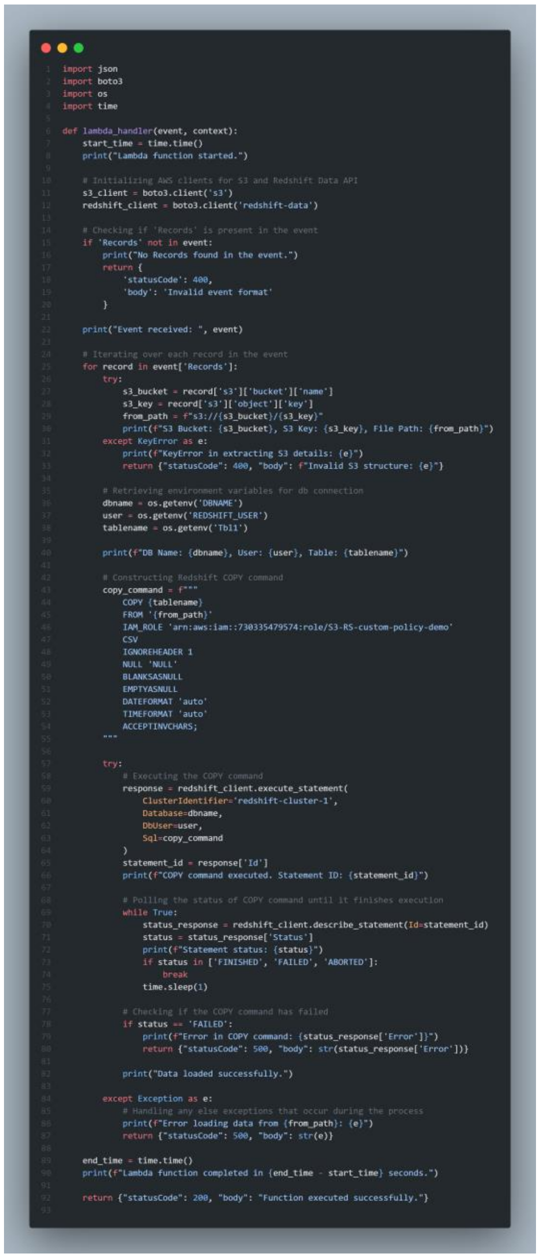
Description
-

Last modified
31 minutes ago

Function ARN
arn:aws:lambda:us-east-1:730335479574:function:lambda-demo-boto3

Function URL Info
-

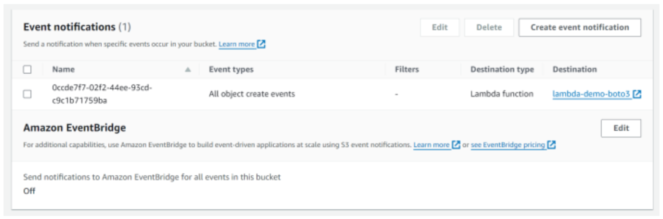
3. **Write the Function**:
    a. Use the following template for the Lambda function (using Boto3):



```python
import json
import boto3
import os
import time

def lambda_handler(event, context):
    start_time = time.time()
    print("Lambda function started.")

    # Initializing AWS clients for S3 and Redshift Data API
    s3_client = boto3.client('s3')
    redshift_client = boto3.client('redshift-data')

    # Checking if 'Records' is present in the event
    if 'Records' not in event:
        print("No Records found in the event.")
        return {
            'statusCode': 400,
            'body': 'Invalid event format'
        }

    print("Event received: ", event)

    # Iterating over each record in the event
    for record in event['Records']:
        try:
            s3_bucket = record['s3']['bucket']['name']
            s3_key = record['s3']['object']['key']
            from_path = f"s3://{s3_bucket}/{s3_key}"
            print(f"S3 Bucket: {s3_bucket}, S3 Key: {s3_key}, File Path: {from_path}")
        except KeyError as e:
            print(f"KeyError in extracting S3 details: {e}")
            return {"statusCode": 400, "body": f"Invalid S3 structure: {e}"}

        # Retrieving environment variables for db connection
        dbname = os.getenv('DBNAME')
        user = os.getenv('REDSHIFT_USER')
        tablename = os.getenv('Tbl1')

        print(f"DB Name: {dbname}, User: {user}, Table: {tablename}")

        # Constructing Redshift COPY command
        copy_command = f"""
            COPY {tablename}
            FROM '{from_path}'
            IAM_ROLE 'arn:aws:iam::730335479574:role/S3-RS-custom-policy-demo'
            CSV
            IGNOREHEADER 1
            NULL 'NULL'
            BLANKSASNULL
            EMPTYASNULL
            DATEFORMAT 'auto'
            TIMEFORMAT 'auto'
            ACCEPTINVCHARS;
        """

        try:
            # Executing the COPY command
            response = redshift_client.execute_statement(
                ClusterIdentifier='redshift-cluster-1',
                Database=dbname,
                DbUser=user,
                Sql=copy_command
            )
            statement_id = response['Id']
            print(f"COPY command executed. Statement ID: {statement_id}")

            # Polling the status of COPY command until it finishes execution
            while True:
                status_response = redshift_client.describe_statement(Id=statement_id)
                status = status_response['Status']
                print(f"Statement status: {status}")
                if status in ['FINISHED', 'FAILED', 'ABORTED']:
                    break
                time.sleep(1)

            # Checking if the COPY command has failed
            if status == 'FAILED':
                print(f"Error in COPY command: {status_response['Error']}")
                return {"statusCode": 500, "body": str(status_response['Error'])}

            print("Data loaded successfully.")

        except Exception as e:
            # Handling any else exceptions that occur during the process
            print(f"Error loading data from {from_path}: {e}")
            return {"statusCode": 500, "body": str(e)}

    end_time = time.time()
    print(f"Lambda function completed in {end_time - start_time} seconds.")

    return {"statusCode": 200, "body": "Function executed successfully."}
```

4. **Set Environment Variables**:
    a. Add S3_BUCKET with the name of the S3 bucket.

5. **Configure Trigger**:
    a. Add an S3 trigger for the Lambda function.
    b. Specify the event type (e.g., "All object create events/PUT/POST").



Event notifications (1)

Send a notification when specific events occur in your bucket. Learn more

Name	Event types	Filters	Destination type	Destination
0ccde7f7-02f2-44ee-93cd-c9c1b71759ba	All object create events	-	Lambda function	lambda-demo-boto3

Amazon EventBridge

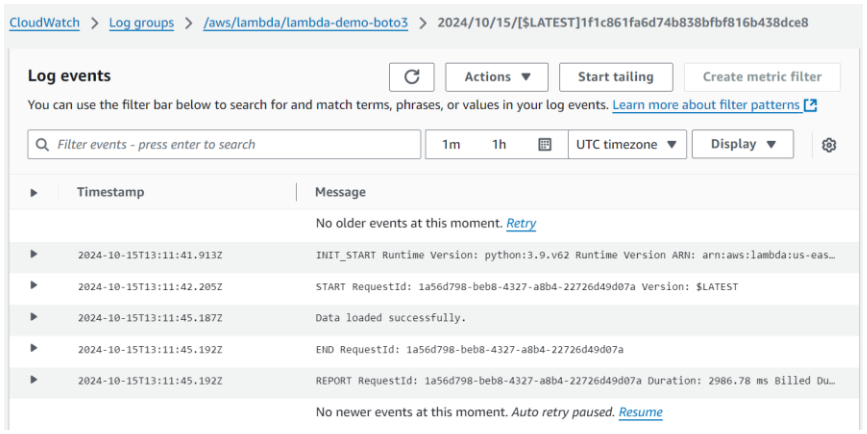
For additional capabilities, use Amazon EventBridge to build event-driven applications at scale using S3 event notifications. Learn more or see EventBridge pricing

Send notifications to Amazon EventBridge for all events in this bucket
Off

5. Testing the Setup

1. **Upload a File to S3**:
    a. Place a CSV file in the S3 bucket to trigger the Lambda function.

2. **Check Lambda Execution**:
    a. Go to the AWS Lambda console and check the logs in CloudWatch for any errors or success messages.



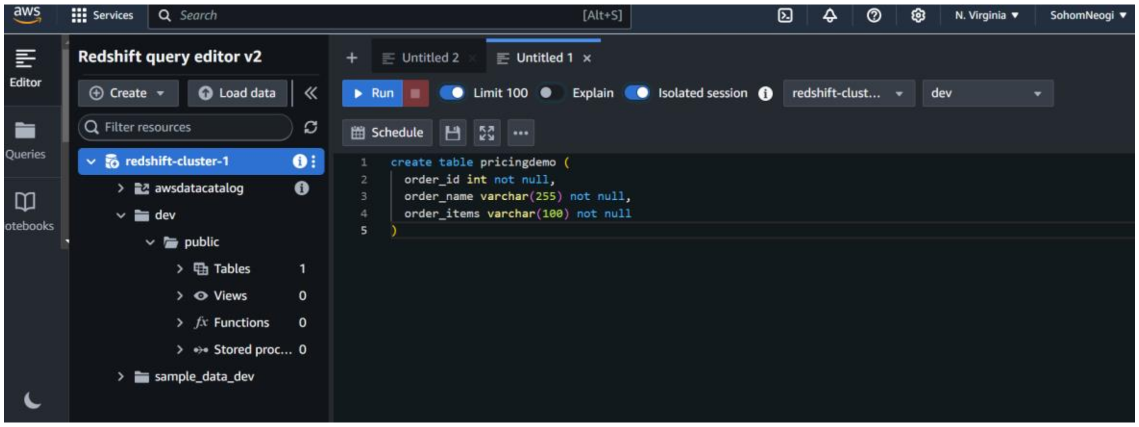
CloudWatch > Log groups > /aws/lambda/lambda-demo-boto3 > 2024/10/15/[$LATEST]1f1c861fa6d74b838bfbf816b438dce8

Log events

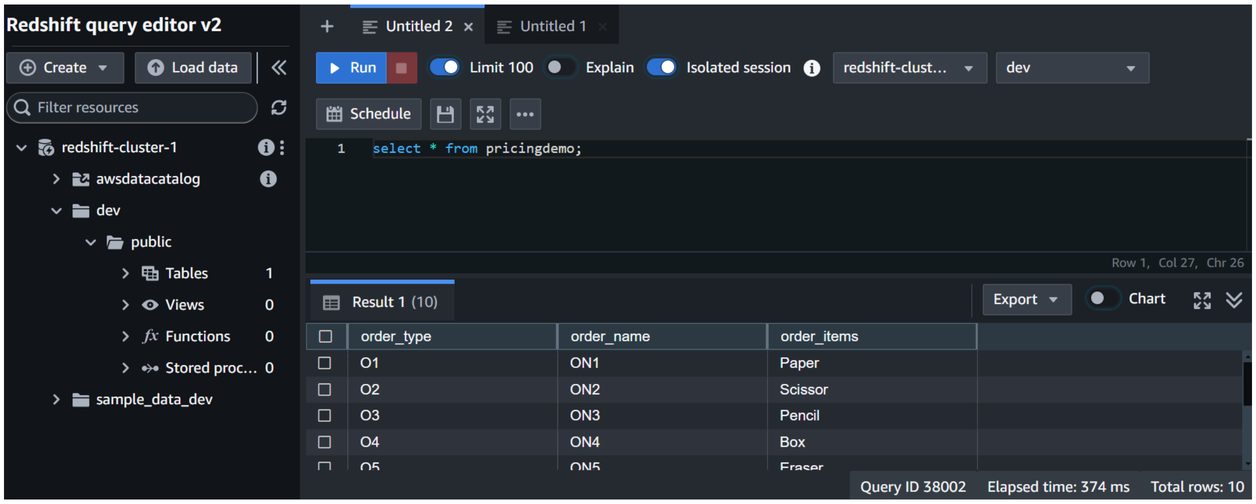
You can use the filter bar below to search for and match terms, phrases, or values in your log events. Learn more about filter patterns

```
Timestamp                   Message
                            No older events at this moment. Retry
2024-10-15T13:11:41.913Z    INIT_START Runtime Version: python:3.9.v62 Runtime Version ARN: arn:aws:lambda:us-eas…
2024-10-15T13:11:42.205Z    START RequestId: 1a56d798-beb8-4327-a8b4-22726d49d07a Version: $LATEST
2024-10-15T13:11:45.187Z    Data loaded successfully.
2024-10-15T13:11:45.192Z    END RequestId: 1a56d798-beb8-4327-a8b4-22726d49d07a
2024-10-15T13:11:45.192Z    REPORT RequestId: 1a56d798-beb8-4327-a8b4-22726d49d07a Duration: 2986.78 ms Billed Du…
                            No newer events at this moment. Auto retry paused. Resume
```

3. **Verify Data in Redshift**:
    a. Use a SQL client or the Redshift Query Editor to verify that the data has been copied to the table.



```
create table pricingdemo (
  order_id int not null,
  order_name varchar(255) not null,
  order_items varchar(100) not null
)
```



```
select * from pricingdemo;
```

order_type	order_name	order_items
O1	ON1	Paper
O2	ON2	Scissor
O3	ON3	Pencil
O4	ON4	Box
O5	ON5	Eraser

Query ID 38002 Elapsed time: 374 ms Total rows: 10

6. Monitoring and Maintenance

- **CloudWatch Logs**: Monitor logs for the Lambda function for any errors or issues.
- **Redshift Monitoring**: Use Redshift's monitoring tools to check for performance and query issues.

Conclusion

This process automates the transfer of SQL data from S3 to Amazon Redshift using AWS Lambda. By following these steps, one can set up a robust data pipeline (ETL Pipeline) that can handle incoming data efficiently. One can adjust the COPY command and IAM roles as needed based on specific requirements and security policies.

Sources

S3-RS