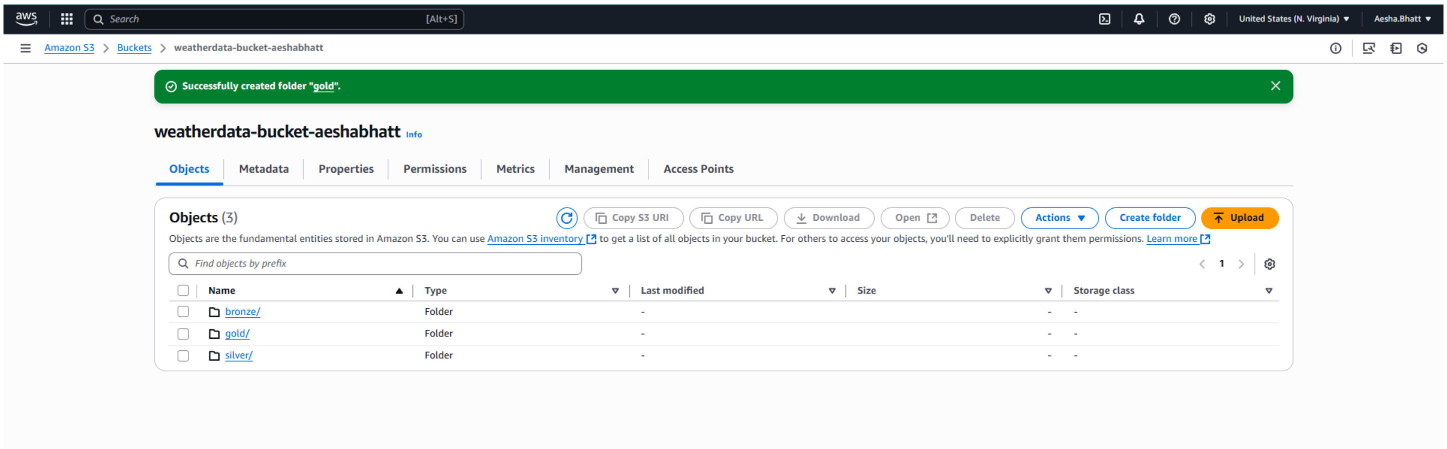
Building a Real-Time Weather Data Pipeline for Weather Analytics

Create an S3 bucket using the AWS Management Console. Once the bucket is set up, create three separate folders within it named bronze, silver, and gold to structure and manage your data efficiently.



Successfully created folder "gold".

weatherdata-bucket-aeshabhatt

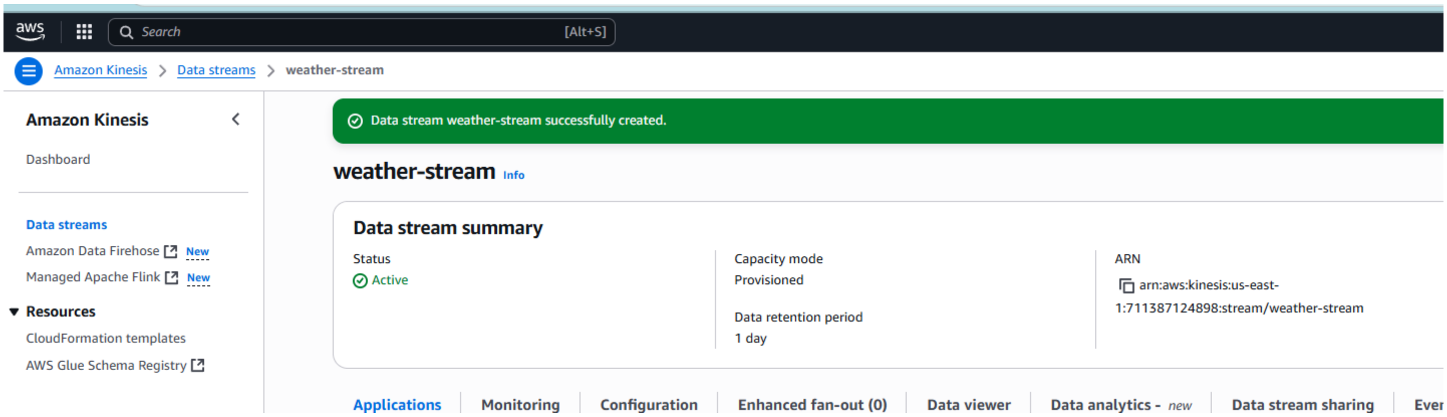
Objects (3)

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
□	bronze/	Folder	-	-	-
□	gold/	Folder	-	-	-
□	silver/	Folder	-	-	-

BRONZE LAYER:

1. Create a Kinesis Data Stream using the AWS Management Console. Specify the stream name and configure the number of shards based on the expected data throughput requirements.



Data stream weather-stream successfully created.

weather-stream

Data stream summary

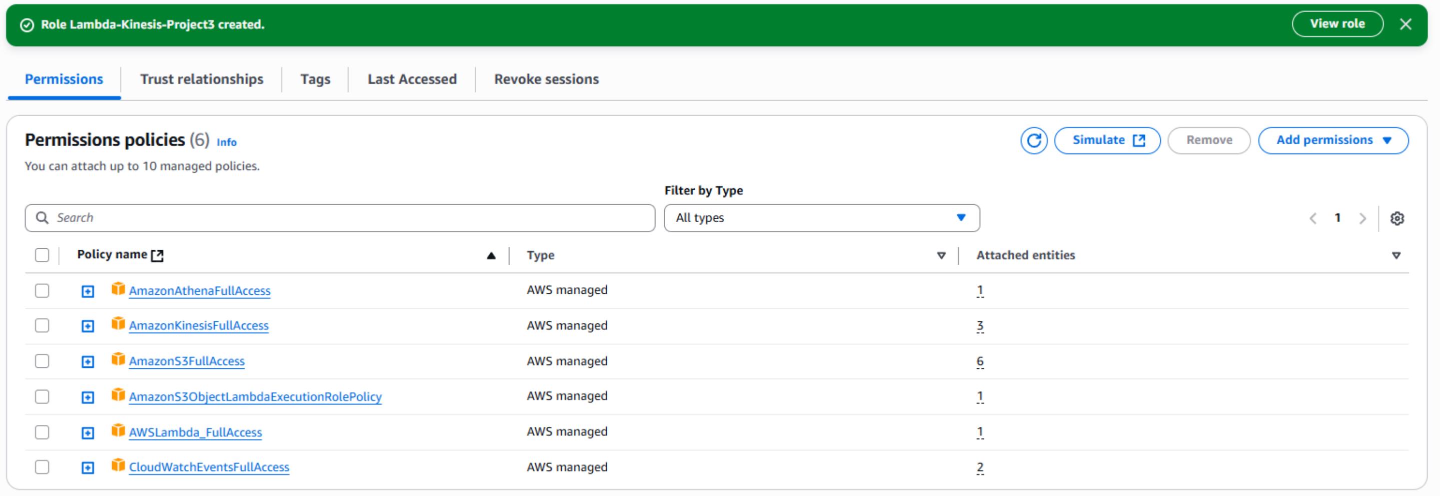
Status: Active

Capacity mode: Provisioned

Data retention period: 1 day

ARN: arn:aws:kinesis:us-east-1:711387124898:stream/weather-stream

2. Prepare and run a script to stream weather data. Begin by creating an IAM role with Lambda as the use case. Attach the necessary permissions, such as access to Kinesis, CloudWatch, and any other services the Lambda function will interact with. This role will be assigned to the Lambda function responsible for ingesting and streaming the weather data into the Kinesis Data Stream.



Permissions policies (6) Info

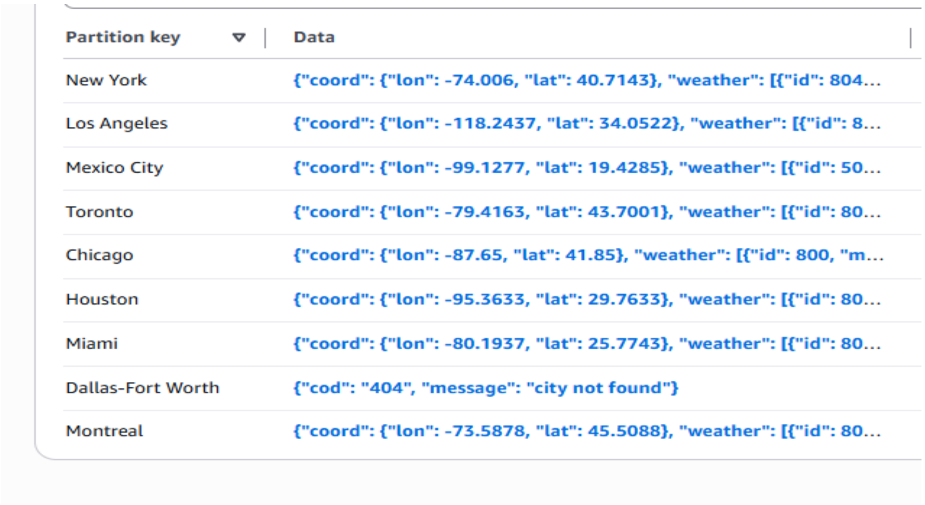
You can attach up to 10 managed policies.

Filter by Type: All types

□	Policy name	Type	Attached entities
□	AmazonAthenaFullAccess	AWS managed	1
□	AmazonKinesisFullAccess	AWS managed	3
□	AmazonS3FullAccess	AWS managed	6
□	AmazonS3ObjectLambdaExecutionRolePolicy	AWS managed	1
□	AWSLambda_FullAccess	AWS managed	1
□	CloudWatchEventsFullAccess	AWS managed	2

To run the script in AWS CloudShell:

1. Click the terminal icon on the bottom navigation bar of the AWS Console to open CloudShell.
2. Open a new file using the nano editor:
   a. Run the command: nano weather_stream-project-3.py
3. Paste the streaming script into the file.
4. To save and exit nano:
   a. Press Ctrl + O to write (save) the file
   b. Press Enter to confirm
   c. Press Ctrl + X to exit the editor
5. Make the script executable by running:
   a. chmod +x weather_stream-project-3.py
6. Run the script using the command:
   a. python3 weather_stream-project-3.py

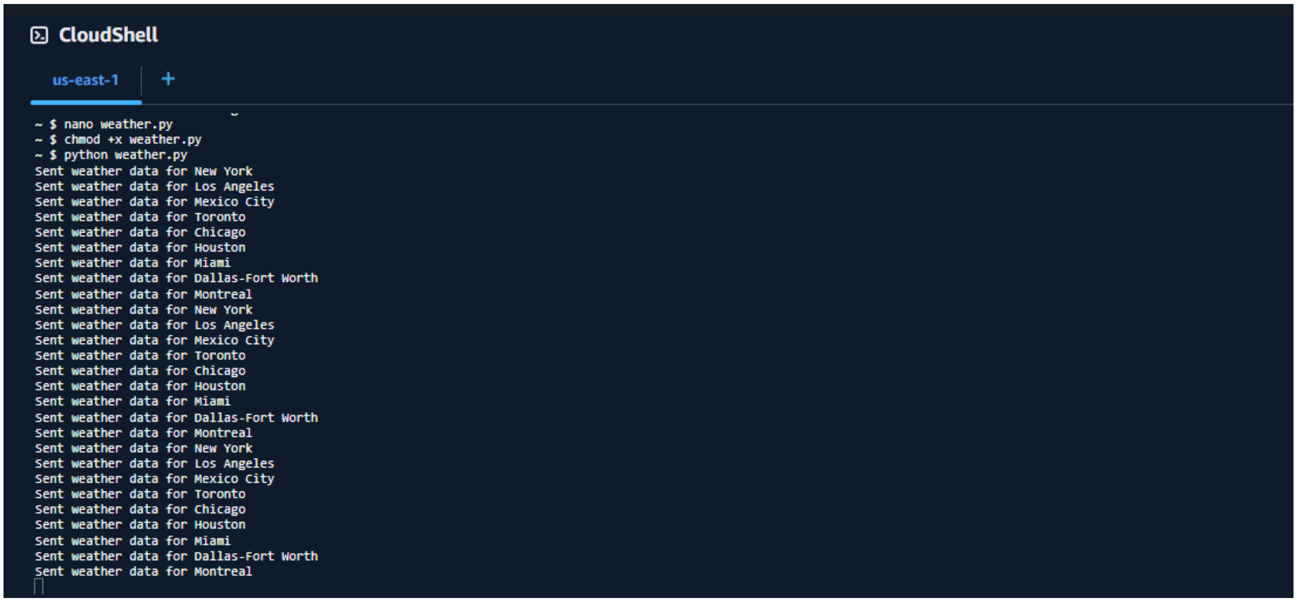


Partition key	Data
New York	{"coord": {"lon": -74.006, "lat": 40.7143}, "weather": [{"id": 804...
Los Angeles	{"coord": {"lon": -118.2437, "lat": 34.0522}, "weather": [{"id": 8...
Mexico City	{"coord": {"lon": -99.1277, "lat": 19.4285}, "weather": [{"id": 50...
Toronto	{"coord": {"lon": -79.4163, "lat": 43.7001}, "weather": [{"id": 80...
Chicago	{"coord": {"lon": -87.65, "lat": 41.85}, "weather": [{"id": 800, "m...
Houston	{"coord": {"lon": -95.3633, "lat": 29.7633}, "weather": [{"id": 80...
Miami	{"coord": {"lon": -80.1937, "lat": 25.7743}, "weather": [{"id": 80...
Dallas-Fort Worth	{"cod": "404", "message": "city not found"}
Montreal	{"coord": {"lon": -73.5878, "lat": 45.5088}, "weather": [{"id": 80...

```
import boto3
import json
import time
import requests

# Create a Kinesis client
kinesis = boto3.client("kinesis", region_name="us-east-1")

# API key and list of cities
API_KEY = "f2ffdd03d1f74ac5b09fa933e24e8876"
CITIES = [
    "New York",
    "Los Angeles",
    "Mexico City",
    "Toronto",
    "Chicago",
    "Houston",
    "Miami",
    "Dallas-Fort Worth",
    "Montreal"
]

# Infinite loop to send weather data every 60 seconds
while True:
    for city in CITIES:
        url = f"http://api.openweathermap.org/data/2.5/weather?q={city}&appid={API_KEY}"
        response = requests.get(url).json()

        # Send data to Kinesis stream
        kinesis.put_record(
            StreamName="weather-stream",
            Data=json.dumps(response),
            PartitionKey=city
        )

        print(f"Sent weather data for {city}")

    # Wait 60 seconds before the next batch
    time.sleep(60)
```

Notes:

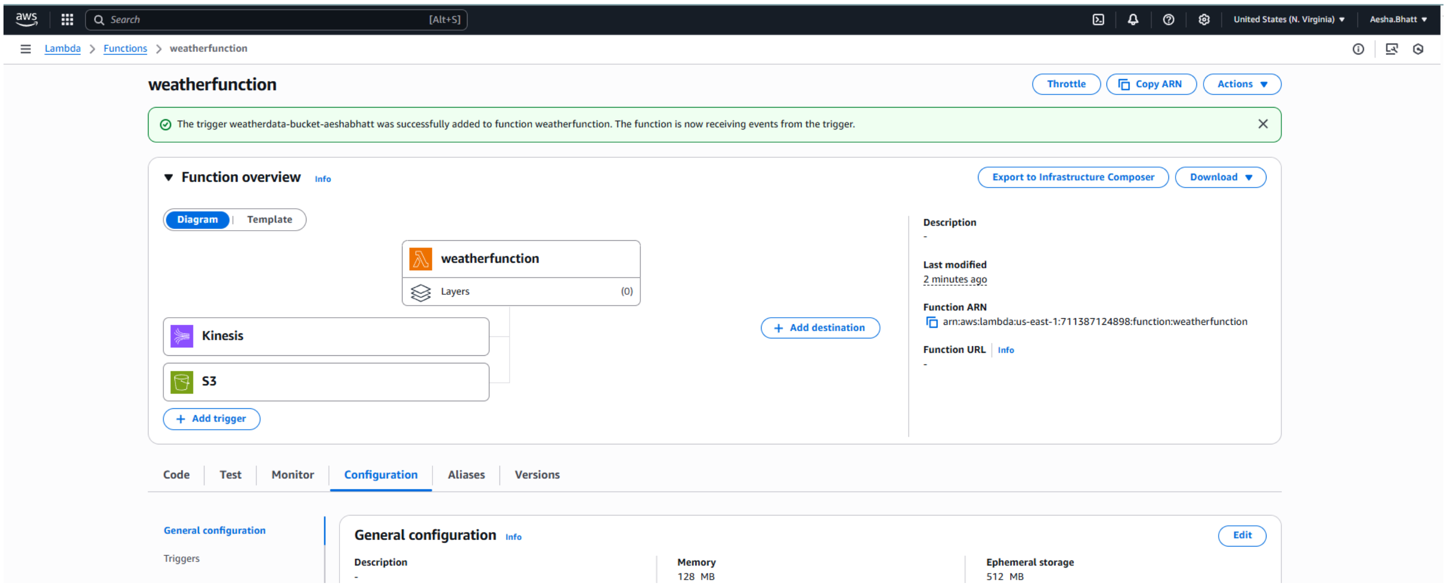
- Make sure the stream name "weather-stream" exists in **us-east-1**.
- The IAM role associated with your environment must have permissions for:
    - kinesis:PutRecord
    - logs:* (for monitoring via CloudWatch)



```
~ $ nano weather.py
~ $ chmod +x weather.py
~ $ python weather.py
Sent weather data for New York
Sent weather data for Los Angeles
Sent weather data for Mexico City
Sent weather data for Toronto
Sent weather data for Chicago
Sent weather data for Houston
Sent weather data for Miami
Sent weather data for Dallas-Fort Worth
Sent weather data for Montreal
Sent weather data for New York
Sent weather data for Los Angeles
Sent weather data for Mexico City
Sent weather data for Toronto
Sent weather data for Chicago
Sent weather data for Houston
Sent weather data for Miami
Sent weather data for Dallas-Fort Worth
Sent weather data for Montreal
Sent weather data for New York
Sent weather data for Los Angeles
Sent weather data for Mexico City
Sent weather data for Toronto
Sent weather data for Chicago
Sent weather data for Houston
Sent weather data for Miami
Sent weather data for Dallas-Fort Worth
Sent weather data for Montreal
```

SILVER LAYER

1. Create a Lambda function to handle the weather data. Use Python 3.x as the runtime and assign the previously created IAM role with appropriate permissions (access to Kinesis, S3, and CloudWatch).
2. Add the following triggers to the Lambda function:
   - **Kinesis**: Configure the Lambda function to be triggered by the Kinesis Data Stream. This allows it to consume real-time weather data as it arrives.
   - **S3**: Integrate S3 by using the boto3 client within the Lambda function to store raw weather data in the Bronze layer and cleaned/processed data in the Silver layer of the S3 bucket.



weatherfunction

The trigger weatherdata-bucket-aeshabhatt was successfully added to function weatherfunction. The function is now receiving events from the trigger.

Function overview

weatherfunction

Layers (0)

Kinesis

S3

Description
-

Last modified
2 minutes ago

Function ARN
arn:aws:lambda:us-east-1:711387124898:function:weatherfunction

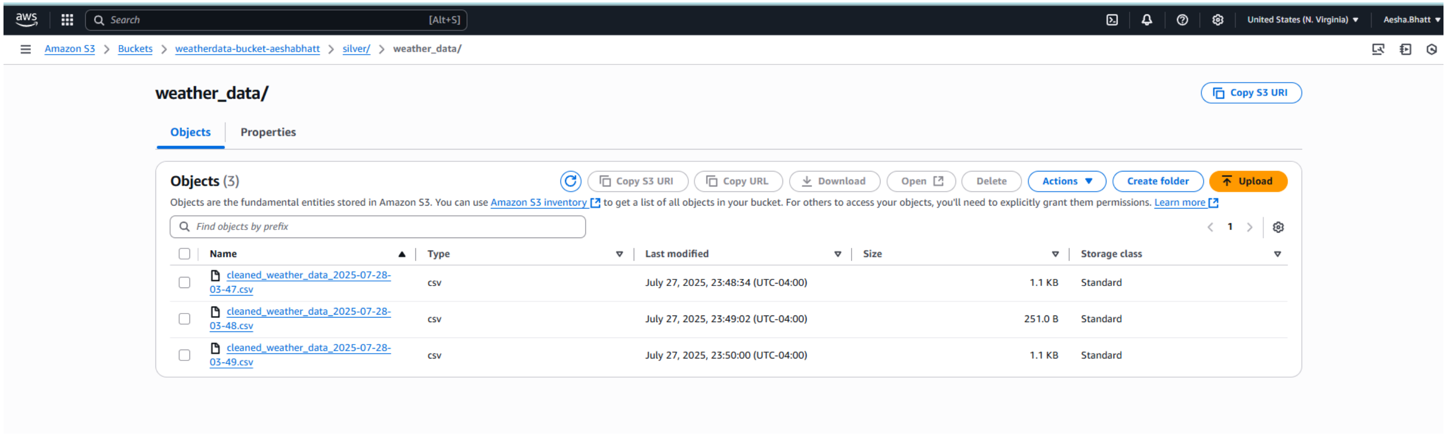
Function URL
-

General configuration

Description	Memory	Ephemeral storage
-	128 MB	512 MB

```
import json
import boto3
import base64
from datetime import datetime
import pandas as pd
import io

# AWS clients
s3 = boto3.client('s3')

# bucket & prefixes
bucket = "weatherdata-bucket-aeshabhatt"
```

```
bronze_prefix = "bronze/weather_data/"
silver_prefix = "silver/weather_data/"


def lambda_handler(event, context):
    print("Received event:", json.dumps(event))

    # Detect Kinesis event
    if 'Records' in event and 'kinesis' in event['Records'][0]:
        return handle_kinesis_event(event)

    # Detect S3 event
    elif 'Records' in event and 's3' in event['Records'][0]:
        return handle_s3_event(event)

    else:
        return {
            "statusCode": 400,
            "body": "Unknown event type"
        }

def handle_kinesis_event(event):
    """Triggered by Kinesis : stores raw JSON in Bronze layer (handles multiple records)"""
    try:
        # Decode all records in the batch
        records = [
            json.loads(base64.b64decode(rec['kinesis']['data']).decode('utf-8'))
            for rec in event['Records']
        ]

        now = datetime.utcnow().strftime("%Y-%m-%d-%H-%M")
        filename = f"weather_data_{now}.json"

        # Save as a JSON array
        body = json.dumps(records, indent=2)

        # Upload to Bronze layer
        s3.put_object(
            Bucket=bucket,
            Key=f"{bronze_prefix}{filename}",
            Body=body,
            ContentType='application/json'
        )

        print(f"Uploaded {len(records)} records to s3://{bucket}/{bronze_prefix}{filename}")
        return {"statusCode": 200, "body": f"Uploaded {len(records)} records"}

    except Exception as e:
        print(f"Error in handle_kinesis_event: {e}")
        return {"statusCode": 500, "body": str(e)}

def handle_s3_event(event):
    """Triggered by S3 : reads Bronze JSON → clean
```



weather_data/

Objects (3)

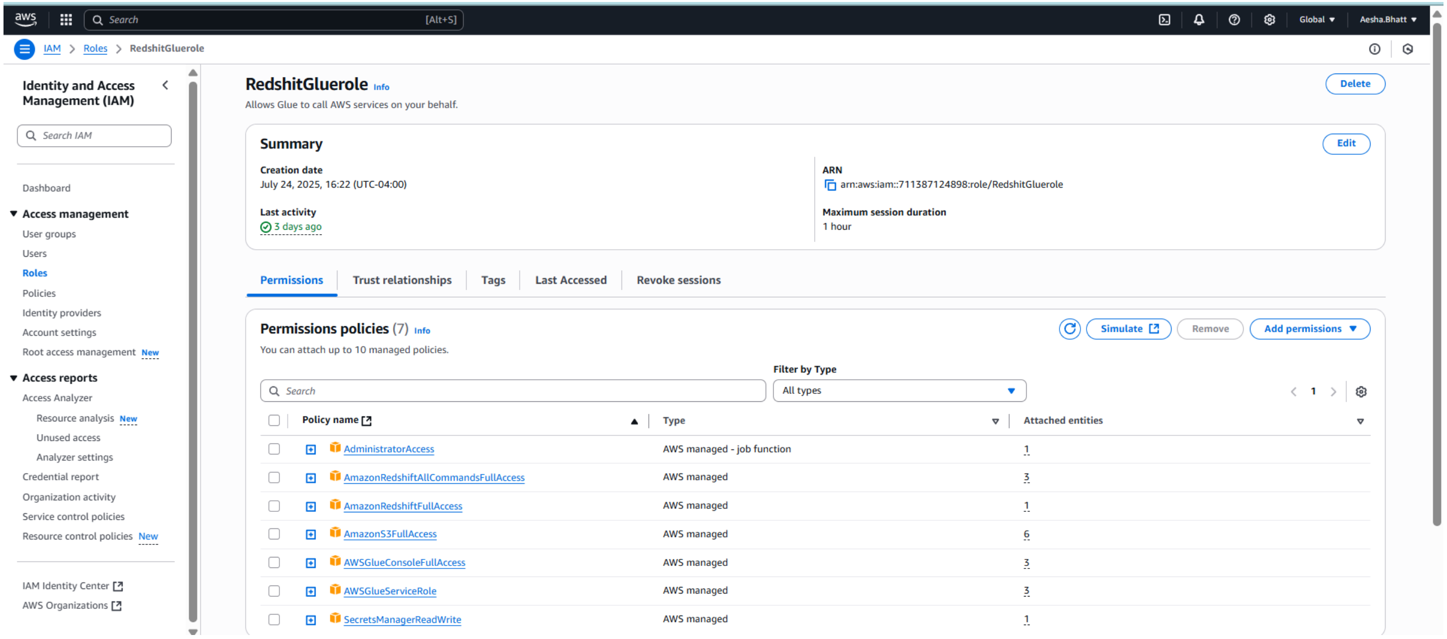
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
cleaned_weather_data_2025-07-28-03-47.csv	csv	July 27, 2025, 23:48:34 (UTC-04:00)	1.1 KB	Standard
cleaned_weather_data_2025-07-28-03-48.csv	csv	July 27, 2025, 23:49:02 (UTC-04:00)	251.0 B	Standard
cleaned_weather_data_2025-07-28-03-49.csv	csv	July 27, 2025, 23:50:00 (UTC-04:00)	1.1 KB	Standard

To connect your AWS Redshift cluster to an IAM role, follow these steps:

1. **Create or identify an IAM Role** with the necessary permissions for Redshift to access other AWS services (e.g., S3).
2. **Attach the IAM Role to your Redshift cluster:**
   - Open the AWS Management Console.
   - Navigate to **Amazon Redshift > Clusters**.
   - Select your Redshift cluster.
   - Choose **Actions > Manage IAM roles**.
   - Attach the IAM role you want Redshift to use.
3. **Use the IAM role for authentication or to access resources** such as S3 for COPY/UNLOAD commands without using AWS keys.

This enables Redshift to securely access AWS resources using the assigned IAM role.



RedshitGluerole Info

Allows Glue to call AWS services on your behalf.

Summary

Creation date
July 24, 2025, 16:22 (UTC-04:00)

Last activity
3 days ago

ARN
arn:aws:iam::711387124898:role/RedshitGluerole

Maximum session duration
1 hour

Permissions policies (7) Info

You can attach up to 10 managed policies.

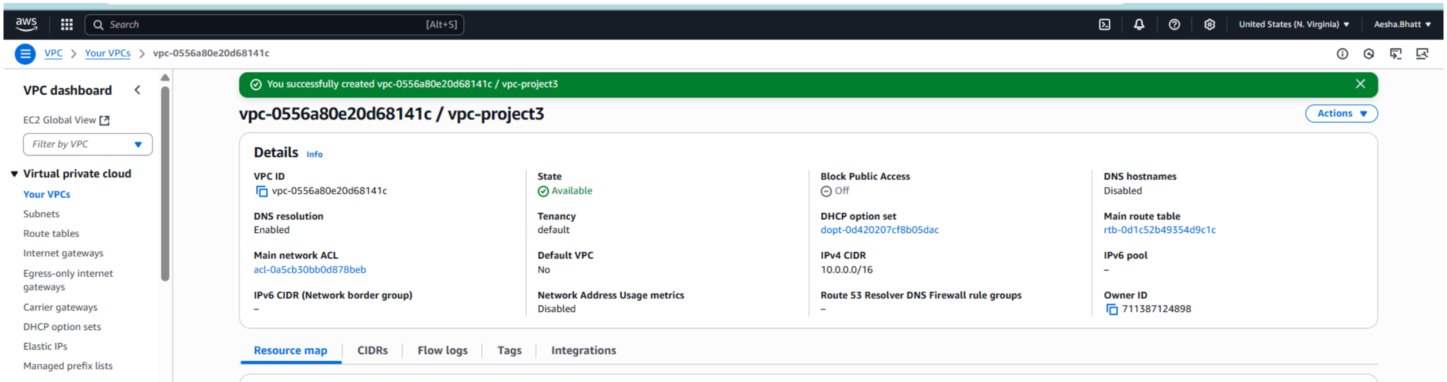
Filter by Type: All types

Policy name	Type	Attached entities
AdministratorAccess	AWS managed - job function	1
AmazonRedshiftAllCommandsFullAccess	AWS managed	3
AmazonRedshiftFullAccess	AWS managed	1
AmazonS3FullAccess	AWS managed	6
AWSGlueConsoleFullAccess	AWS managed	3
AWSGlueServiceRole	AWS managed	3
SecretsManagerReadWrite	AWS managed	1

Here's a clear step-by-step for your document:

1. Create a new VPC in the AWS Management Console.
2. Navigate to **Security Groups** under the VPC dashboard.
3. Select the security group associated with your VPC (or create a new one).
4. Edit the **Inbound Rules** of the security group.
5. Add a rule with the following settings:
    - Protocol: **TCP**
    - Port Range: **5439**
    - Source: specify the IP range or security group allowed to connect (e.g., your IP or 0.0.0.0/0 for all)
6. Save the inbound rule to allow Redshift traffic on port 5439.

This opens the default Redshift port so clients can connect.

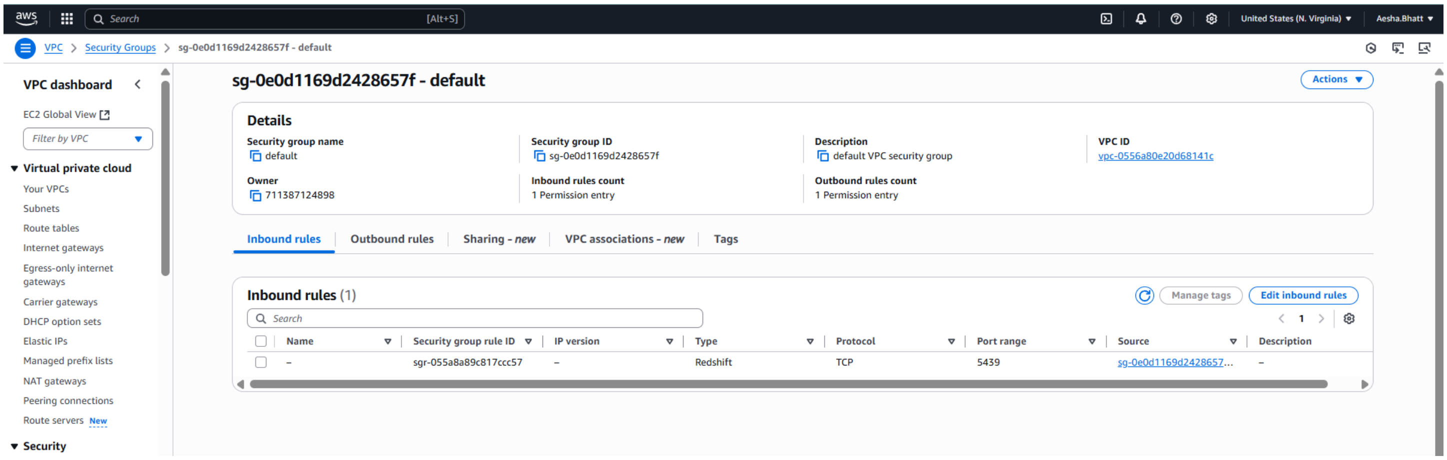


You successfully created vpc-0556a80e20d68141c / vpc-project3

vpc-0556a80e20d68141c / vpc-project3

Details Info

VPC ID vpc-0556a80e20d68141c	State Available	Block Public Access Off	DNS hostnames Disabled
DNS resolution Enabled	Tenancy default	DHCP option set dopt-0d420207cf8b05dac	Main route table rtb-0d1c52b49354d9c1c
Main network ACL acl-0a5cb30bb0d878beb	Default VPC No	IPv4 CIDR 10.0.0.0/16	IPv6 pool –
IPv6 CIDR (Network border group) –	Network Address Usage metrics Disabled	Route 53 Resolver DNS Firewall rule groups –	Owner ID 711387124898



sg-0e0d1169d2428657f - default

Details

Security group name default	Security group ID sg-0e0d1169d2428657f	Description default VPC security group	VPC ID vpc-0556a80e20d68141c
Owner 711387124898	Inbound rules count 1 Permission entry	Outbound rules count 1 Permission entry	

Inbound rules (1)

	Name	Security group rule ID	IP version	Type	Protocol	Port range	Source	Description
□	–	sgr-055a8a89c817ccc57	–	Redshift	TCP	5439	sg-0e0d1169d2428657...	–

To create a Redshift workgroup configuration, follow these steps:

1. Open the AWS Management Console and navigate to **Amazon Redshift**.

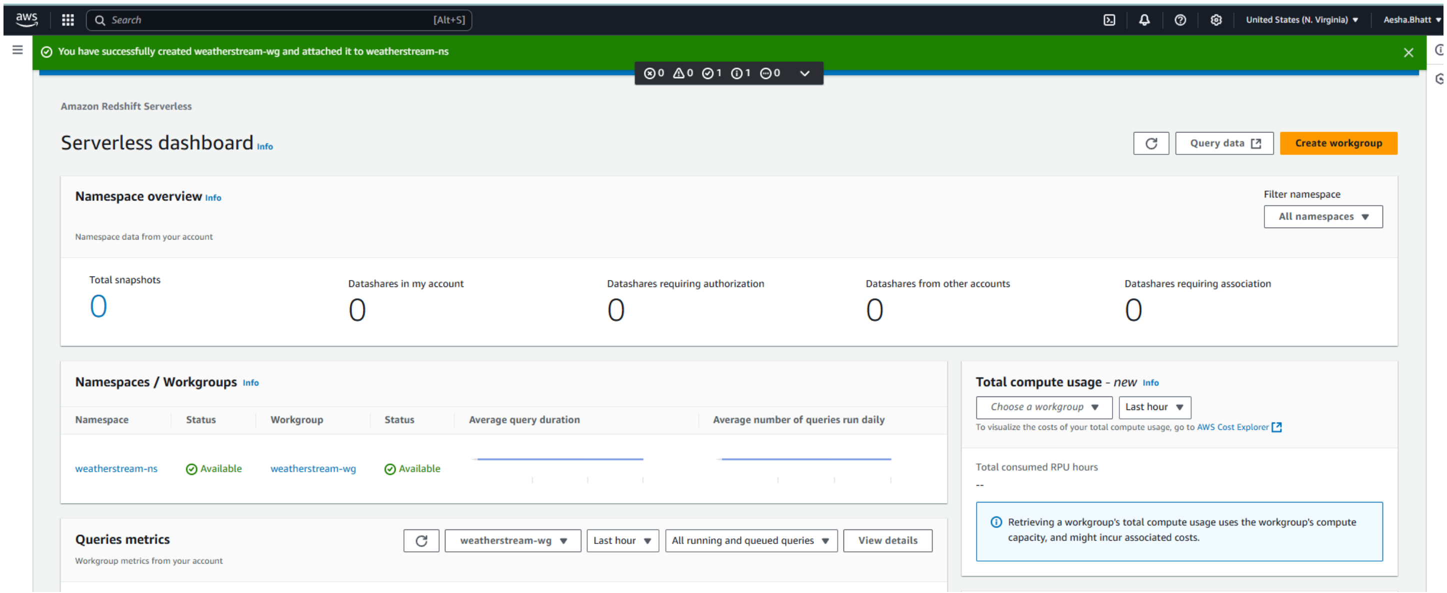
2. In the left navigation pane, select **Workgroups**.

3. Click on **Create workgroup**.

4. Enter a **workgroup name**.

5. Select the **VPC** where your Redshift cluster and resources reside.
6. Choose the **subnets** (preferably private subnets in multiple availability zones) that the workgroup will use.
7. Assign **security groups** that control access to the workgroup.
8. Configure additional settings such as:
   - Encryption options.
   - Enhanced VPC routing.
   - Logging options.
9. Review the configuration and click **Create workgroup** to finalize.

This workgroup configuration helps manage the networking, security, and access settings for your Redshift Serverless environment.



You have successfully created weatherstream-wg and attached it to weatherstream-ns

Amazon Redshift Serverless

Serverless dashboard Info

Namespace overview Info

Namespace data from your account

Filter namespace: All namespaces

Total snapshots	Datashares in my account	Datashares requiring authorization	Datashares from other accounts	Datashares requiring association
0	0	0	0	0

Namespaces / Workgroups Info

Namespace	Status	Workgroup	Status	Average query duration	Average number of queries run daily
weatherstream-ns	✓ Available	weatherstream-wg	✓ Available		

Total compute usage - *new* Info

Choose a workgroup ▾ Last hour ▾

To visualize the costs of your total compute usage, go to AWS Cost Explorer

Total consumed RPU hours

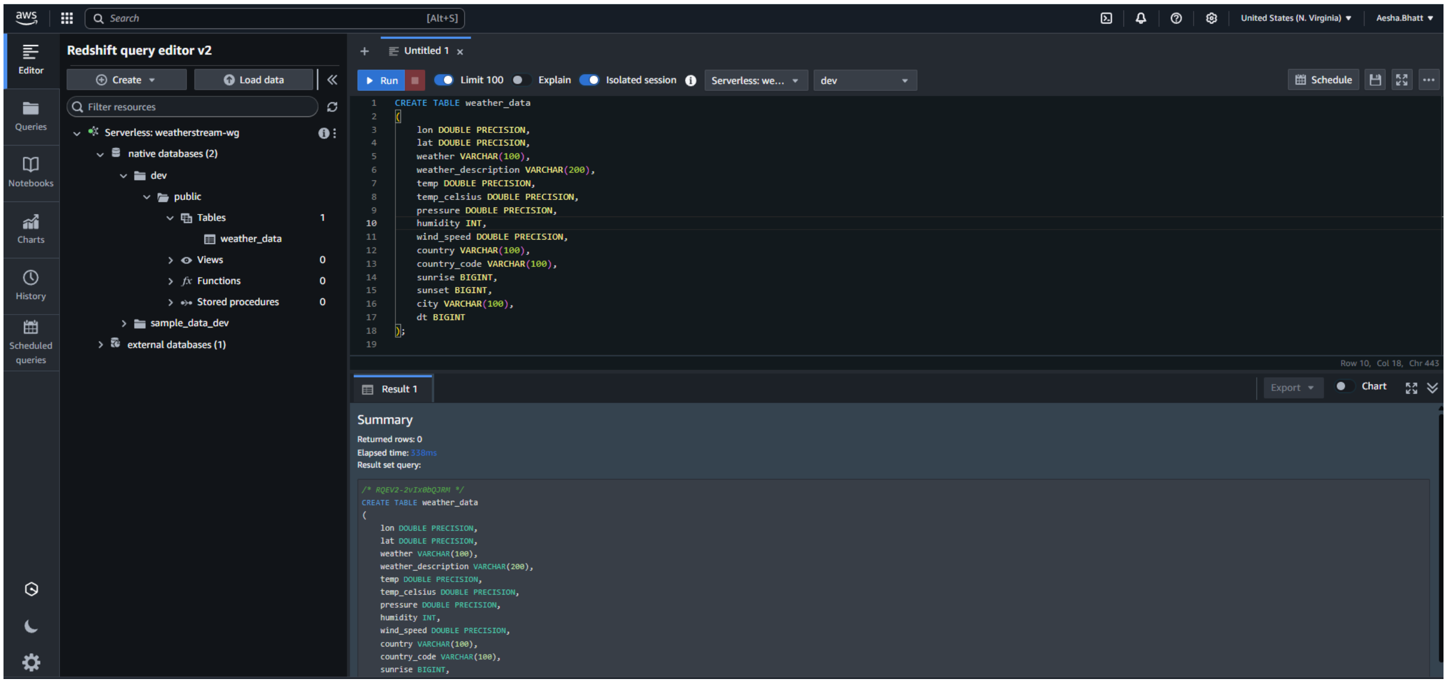
--

ⓘ Retrieving a workgroup's total compute usage uses the workgroup's compute capacity, and might incur associated costs.

Queries metrics

Workgroup metrics from your account

weatherstream-wg ▾ Last hour ▾ All running and queued queries ▾ View details



```
CREATE TABLE weather_data
(
    lon DOUBLE PRECISION,
    lat DOUBLE PRECISION,
    weather VARCHAR(100),
    weather_description VARCHAR(200),
    temp DOUBLE PRECISION,
    temp_celsius DOUBLE PRECISION,
    pressure DOUBLE PRECISION,
    humidity INT,
    wind_speed DOUBLE PRECISION,
    country VARCHAR(100),
    country_code VARCHAR(100),
    sunrise BIGINT,
    sunset BIGINT,
    city VARCHAR(100),
    dt BIGINT
);
```

Summary

Returned rows: 0

Elapsed time: 338ms

Result set query:

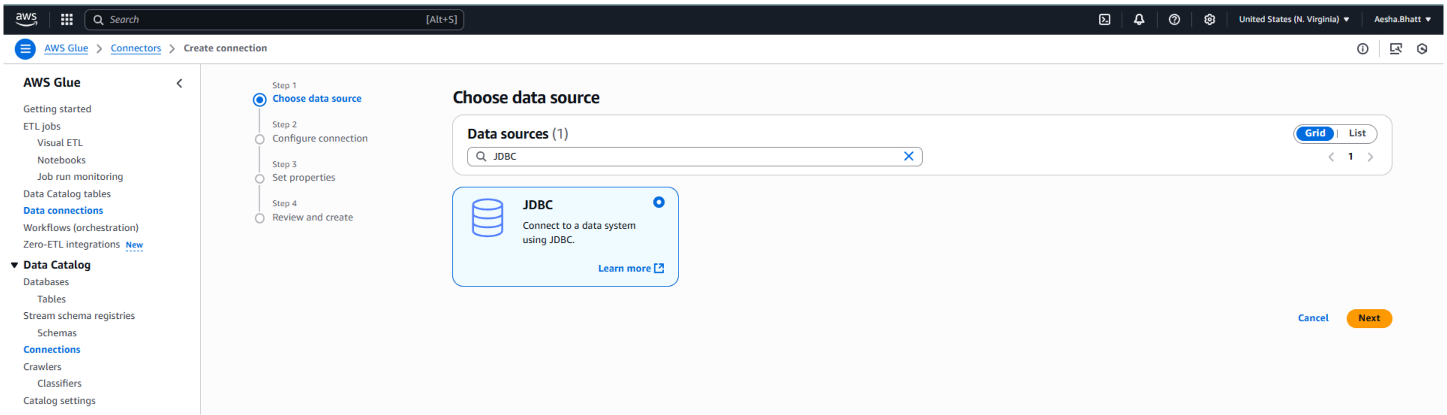
```
/* RQEV2-2vIx0bQJRM */
CREATE TABLE weather_data
(
    lon DOUBLE PRECISION,
    lat DOUBLE PRECISION,
    weather VARCHAR(100),
    weather_description VARCHAR(200),
    temp DOUBLE PRECISION,
    temp_celsius DOUBLE PRECISION,
    pressure DOUBLE PRECISION,
    humidity INT,
    wind_speed DOUBLE PRECISION,
    country VARCHAR(100),
    country_code VARCHAR(100),
    sunrise BIGINT,
```

GOLD LAYER

To create an AWS Glue Connection, follow these steps:

1. Open the AWS Management Console and navigate to **AWS Glue**.
2. In the left navigation pane, select **Connections**.
3. Click **Add connection**.
4. Enter a **name** for the connection.
5. Choose the **connection type** (e.g., JDBC for databases).
6. Configure the connection properties such as:
   - JDBC URL or endpoint
   - Username and password (if required)
   - VPC, subnet, and security groups (to enable Glue to access resources inside your VPC)
7. Review the details and click **Create connection** to save it.

This connection allows AWS Glue jobs and crawlers to securely access data stores inside your VPC or external databases.



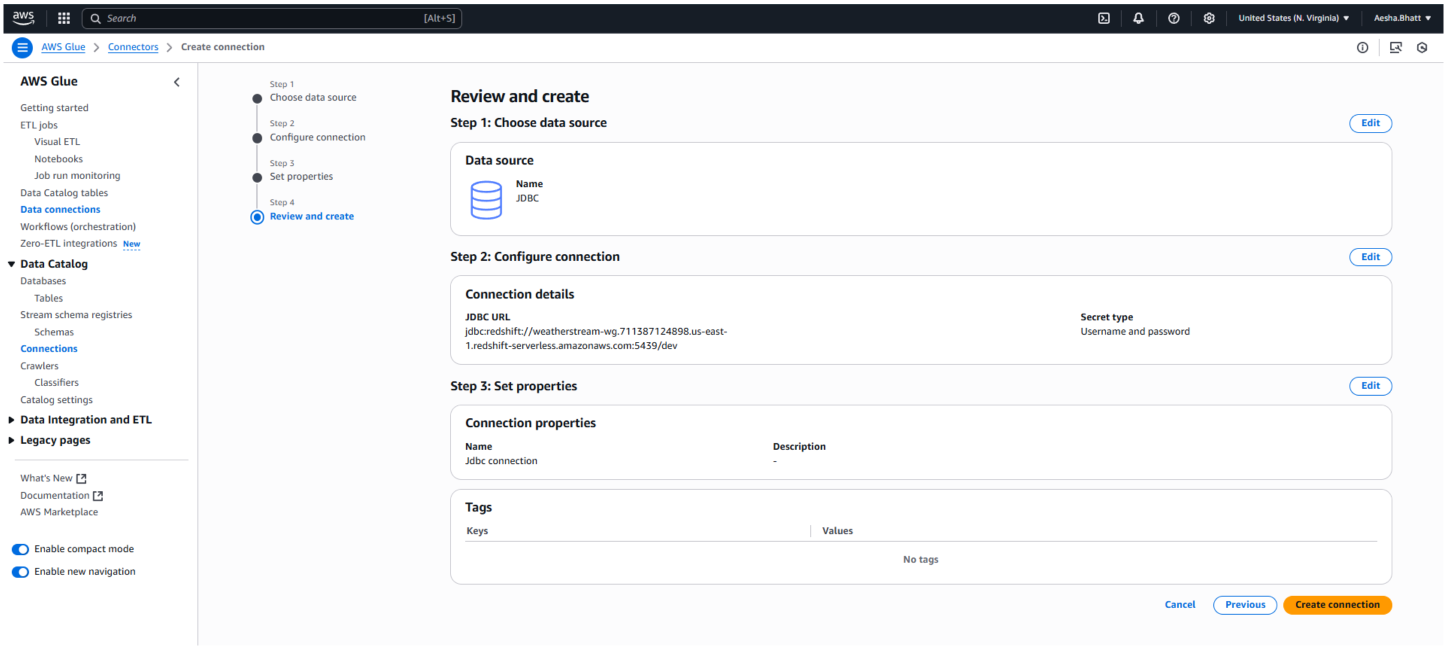
Choose data source

Data sources (1)

JDBC

JDBC

Connect to a data system using JDBC.



Review and create

Step 1: Choose data source

Edit

Data source

Name
JDBC

Step 2: Configure connection

Edit

Connection details

JDBC URL
jdbc:redshift://weatherstream-wg.711387124898.us-east-1.redshift-serverless.amazonaws.com:5439/dev

Secret type
Username and password

Step 3: Set properties

Edit

Connection properties

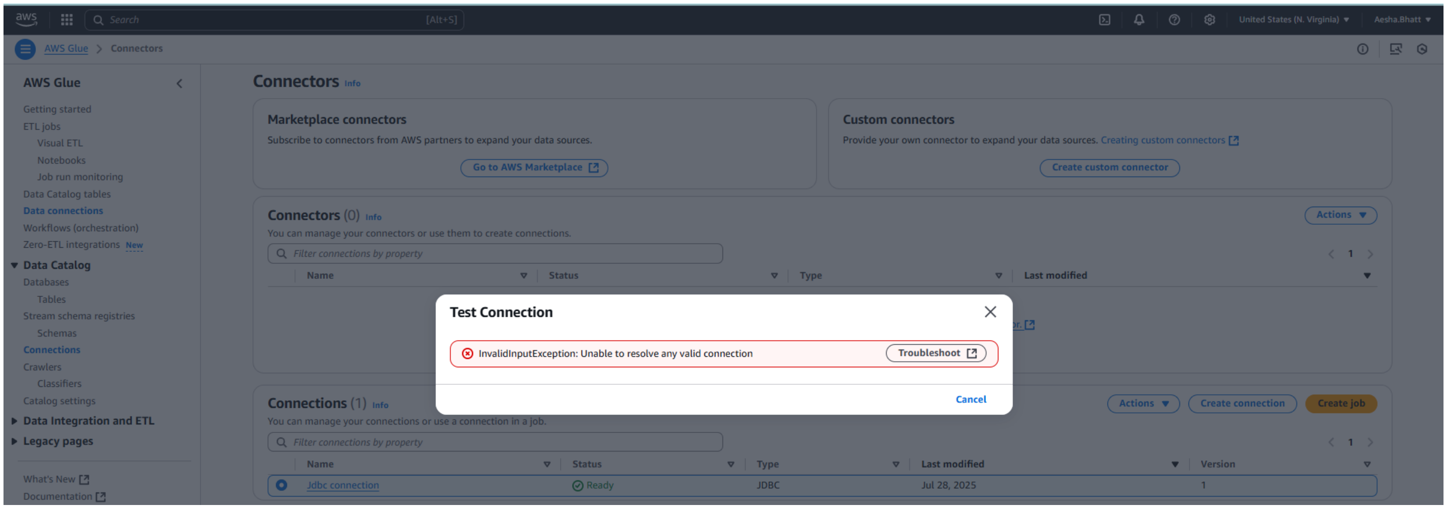
Name
Jdbc connection

Description
-

Tags

Keys	Values
No tags	

Cancel | Previous | Create connection



Connectors Info

Marketplace connectors

Subscribe to connectors from AWS partners to expand your data sources.

Go to AWS Marketplace

Custom connectors

Provide your own connector to expand your data sources. Creating custom connectors

Create custom connector

Connectors (0) Info

You can manage your connectors or use them to create connections.

Actions ▼

Filter connections by property

Name	Status	Type	Last modified

Test Connection

InvalidInputException: Unable to resolve any valid connection

Troubleshoot

Cancel

Connections (1) Info

You can manage your connections or use a connection in a job.

Actions ▼ | Create connection | Create job

Filter connections by property

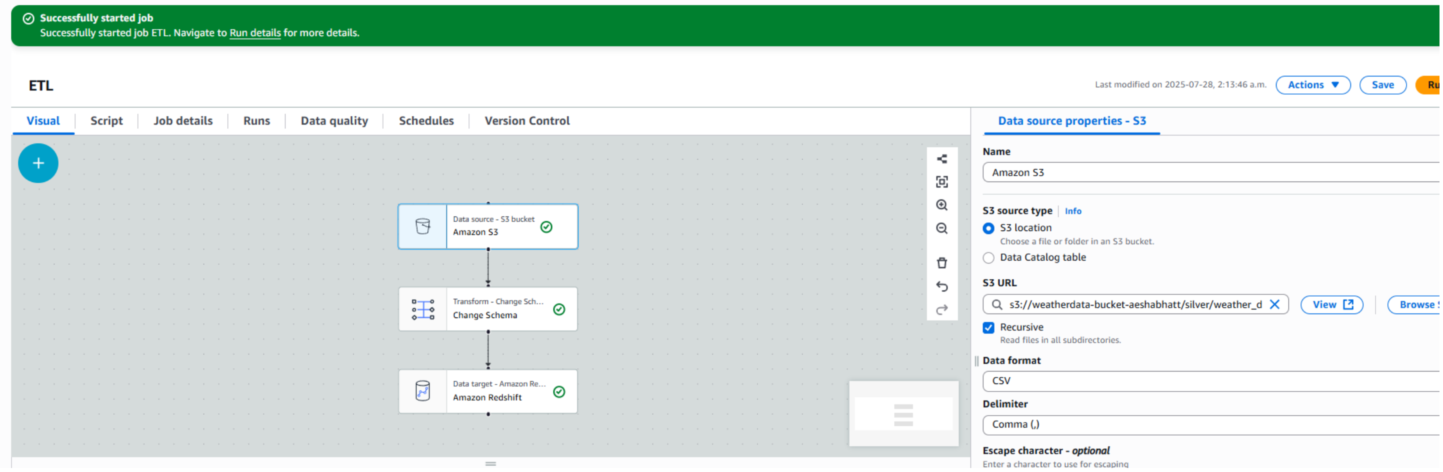
	Name	Status	Type	Last modified	Version
●	Jdbc connection	Ready	JDBC	Jul 28, 2025	1

To create an AWS Glue Visual ETL job, follow these steps:

1. Open the AWS Management Console and go to **AWS Glue**.
2. In the left navigation pane, select **Jobs**.
3. Click **Add job**.
4. Enter a **name** for your job.
5. Choose **Visual with a source and target** as the job type.
6. Select the **IAM role** that has the necessary permissions for Glue to access your data sources and targets.
7. Under **This job runs**, choose the type of data processing (e.g., Spark).
8. Choose the **data source** for your ETL job (e.g., a table in the Glue Data Catalog or a connection).
9. Choose the **data target** where you want to write the processed data (e.g., S3 bucket).
10. Use the **Visual editor** to drag and drop transforms, apply mappings, filters, and other transformations as needed.
11. Review the job settings and click **Save**.

12. To run the job, select it and click **Run job**.

This creates and executes a Glue ETL job using the visual interface, simplifying ETL development without writing code manually.



Successfully started job
Successfully started job ETL. Navigate to Run details for more details.

ETL

Last modified on 2025-07-28, 2:13:46 a.m.

Data source properties - S3

Name: Amazon S3

S3 source type | Info
● S3 location
Choose a file or folder in an S3 bucket.
○ Data Catalog table

S3 URL: s3://weatherdata-bucket-aeshabhatt/silver/weather_d

☑ Recursive
Read files in all subdirectories.

Data format: CSV

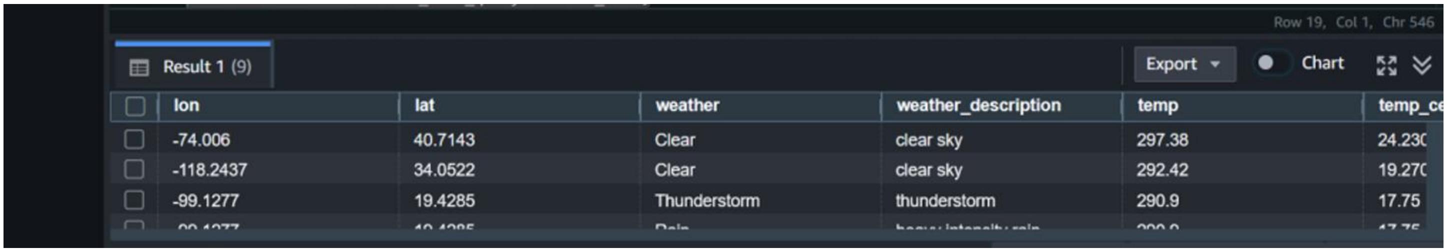
Delimiter: Comma (,)

Escape character - *optional*
Enter a character to use for escaping

To check data in a Redshift table, follow these steps:

1. Connect to your Redshift cluster using a SQL client (such as **Redshift Query Editor**, **SQL Workbench/J**, or any other SQL client).
2. Select the appropriate **database** and **schema** where your table resides.
3. Review the query results to verify the data in the table.

This lets you quickly inspect the contents of your Redshift table.

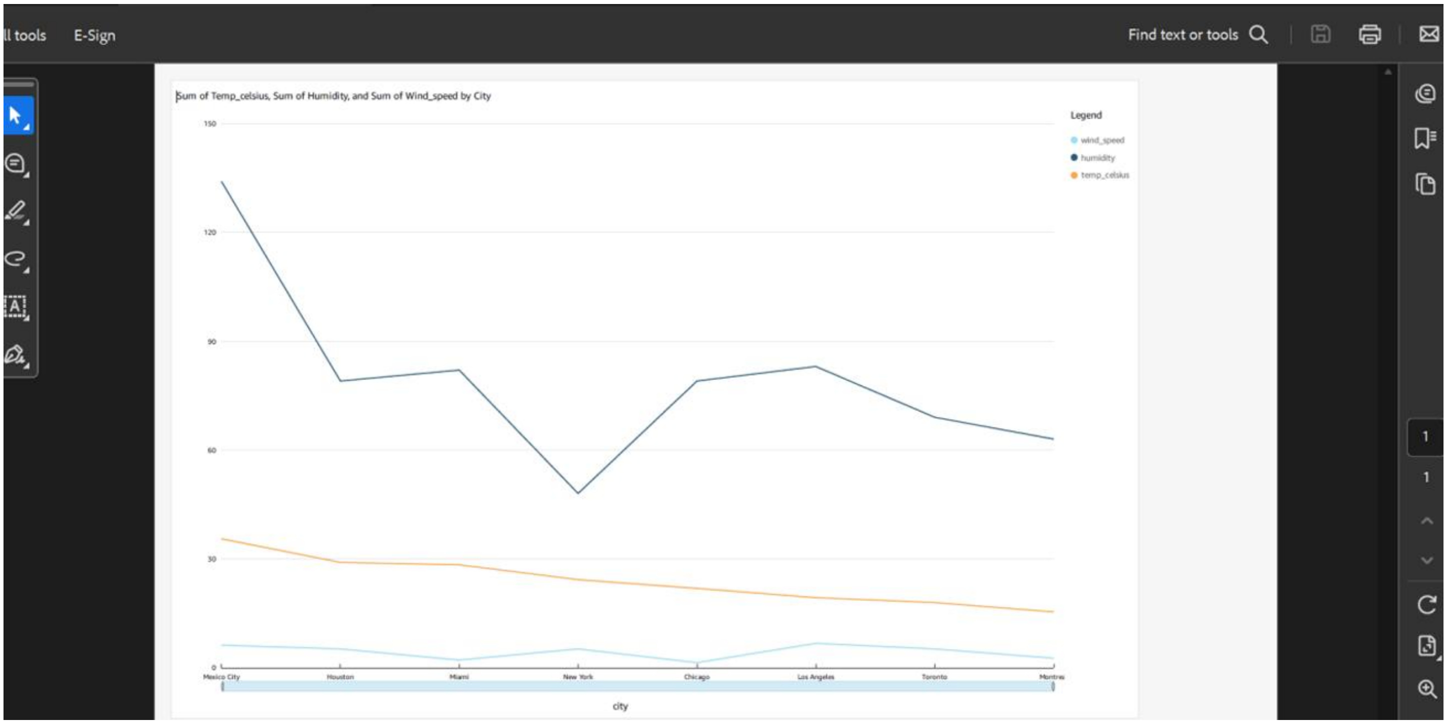


Row 19, Col 1, Chr 546

Result 1 (9)

lon	lat	weather	weather_description	temp	temp_ce
-74.006	40.7143	Clear	clear sky	297.38	24.23C
-118.2437	34.0522	Clear	clear sky	292.42	19.27C
-99.1277	19.4285	Thunderstorm	thunderstorm	290.9	17.75
-99.1277	19.4285	Rain	heavy intensity rain	290.9	17.75

Visualisation:


ll tools
E-Sign
Find text or tools
Sum of Temp_celsius, Sum of Humidity, and Sum of Wind_speed by City
Legend
● wind_speed
● humidity
● temp_celsius
Mexico City
Houston
Miami
New York
Chicago
Los Angeles
Toronto
Montrea
city