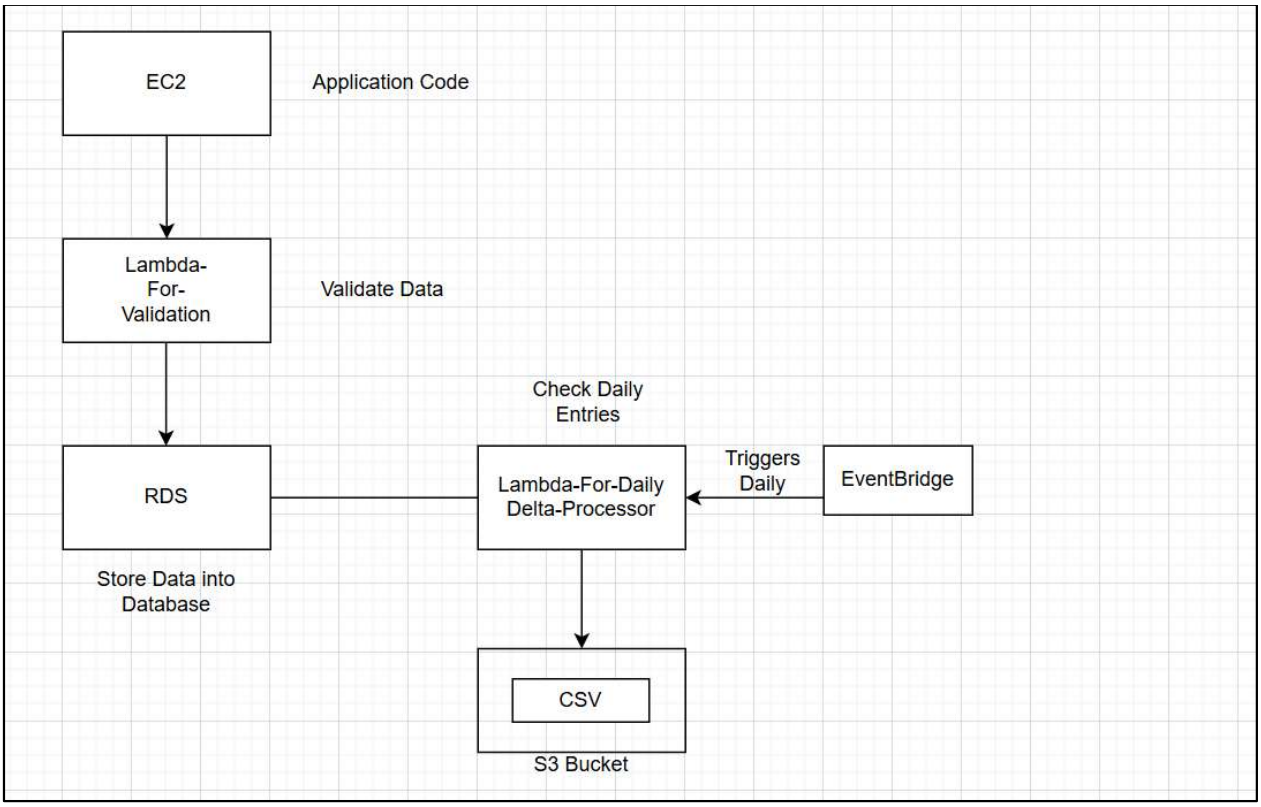
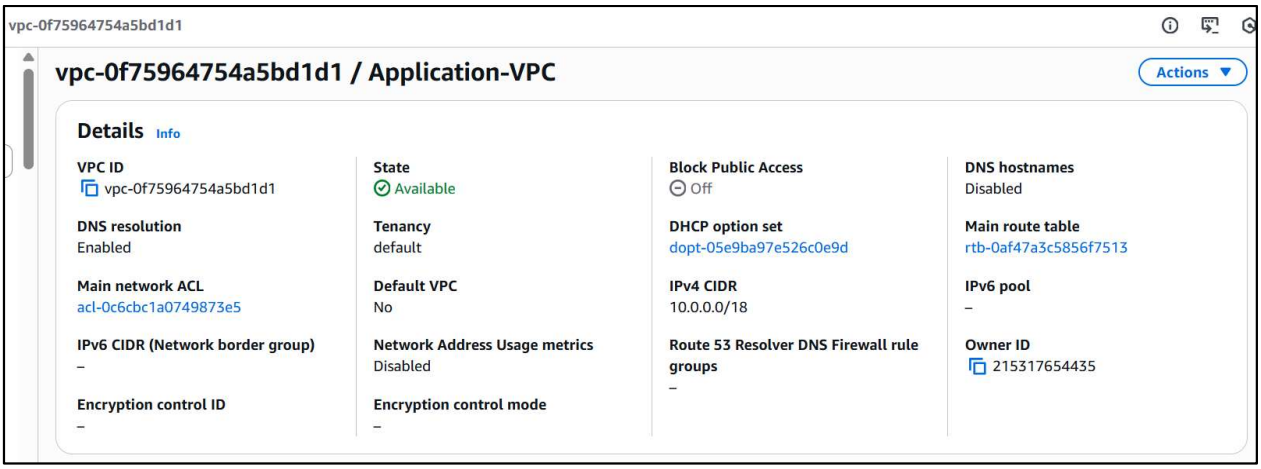
Task 21

Automated Order Processing System



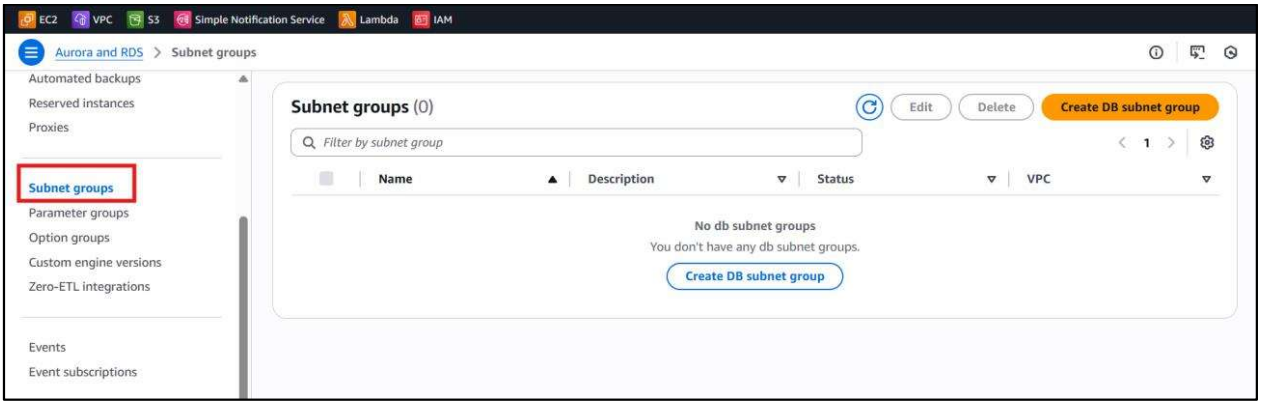
Step 1 : Create VPC

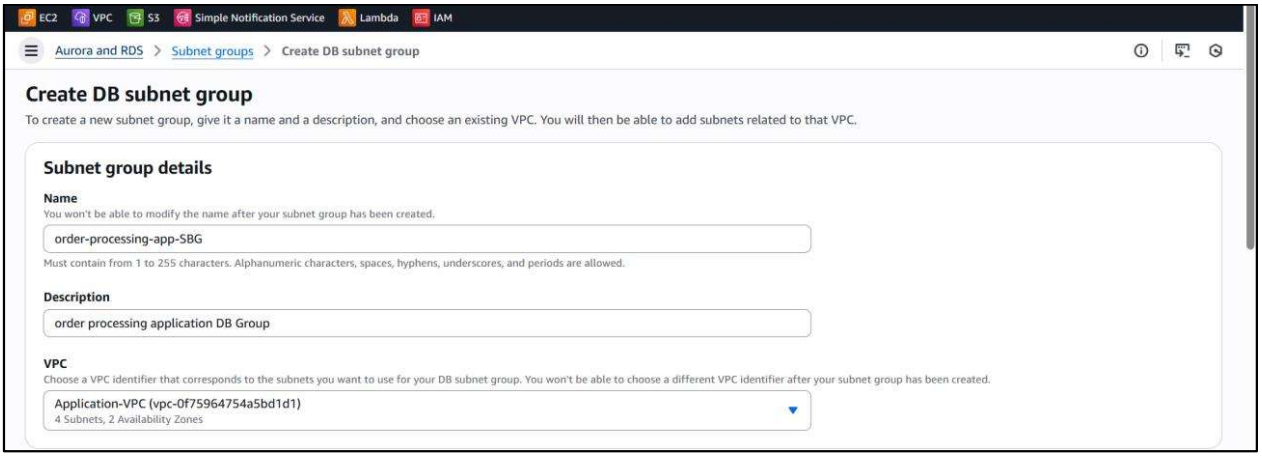
- 2 Public subnet → EC2
- 2 Private subnet → RDS

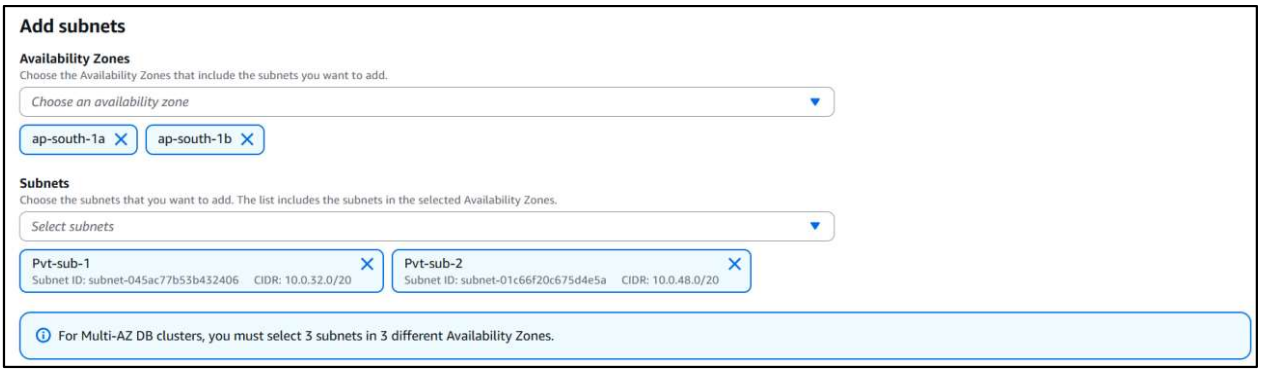


Step 2 : Create Subnet Group

- Go to RDS click on Subnet Group
- Click create subnet .
- Name : order-processing-app-SG
- VPC : same as created above.
- Availability Zone : select in which Private instance are present
- Subnets : Private-Subnet
- Create.



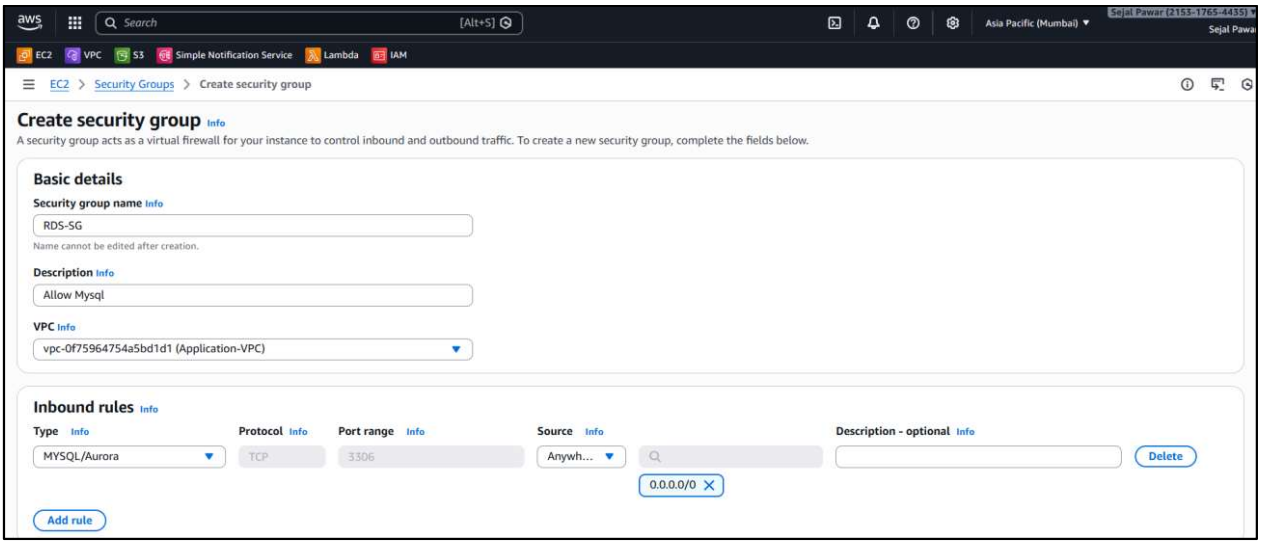



Add subnets
Availability Zones
Choose the Availability Zones that include the subnets you want to add.
Choose an availability zone
ap-south-1a
ap-south-1b
Subnets
Choose the subnets that you want to add. The list includes the subnets in the selected Availability Zones.
Select subnets
Pvt-sub-1
Subnet ID: subnet-045ac77b53b432406 CIDR: 10.0.32.0/20
Pvt-sub-2
Subnet ID: subnet-01c66f20c675d4e5a CIDR: 10.0.48.0/20
For Multi-AZ DB clusters, you must select 3 subnets in 3 different Availability Zones.

Step 3 : Create Security Group

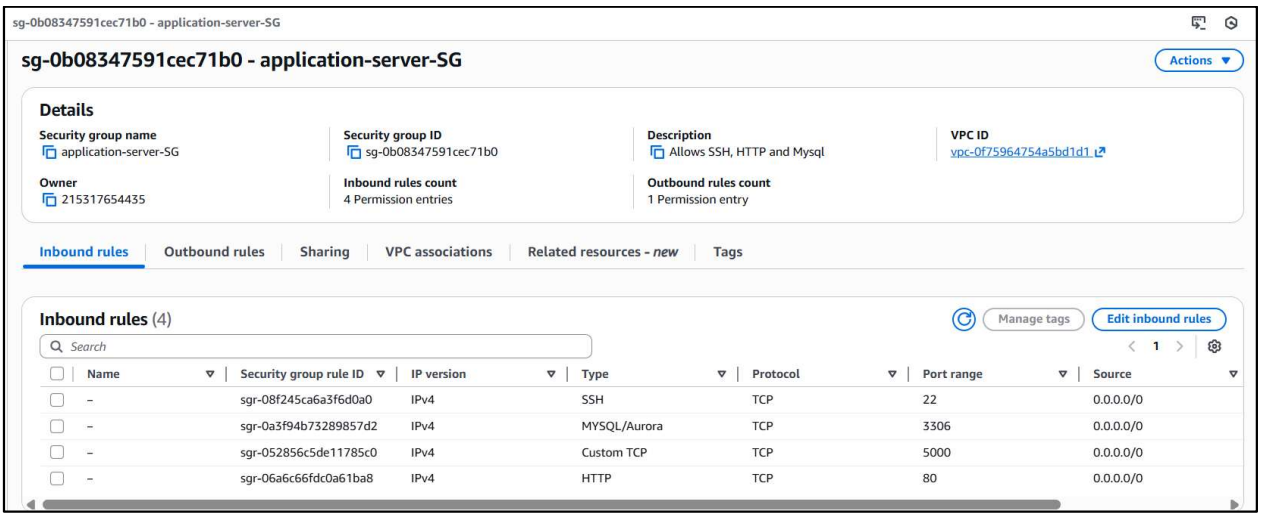
Step 3.1 : Security group for RDS

- Inbound Rule :
    - MySql : 3306 (Allow)



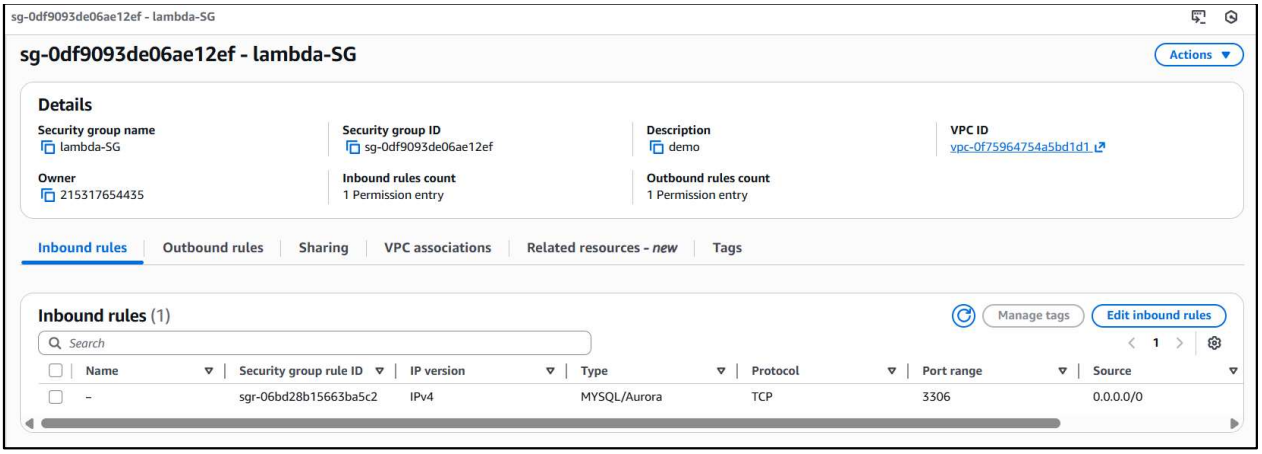
Step 3.2 : Security group for EC2

- Inbound Rule :
    - MySql : 3306 (Allow)
    - SSH : 22
    - HTTP : 80
    - Custom TCP : 50



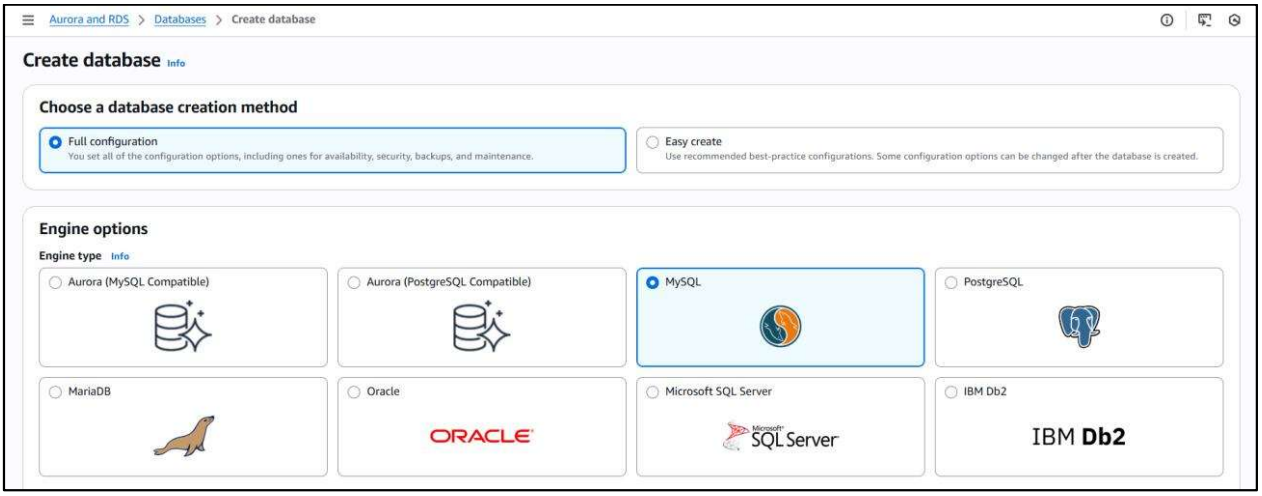
Step 3.2 : Security group for Lambda

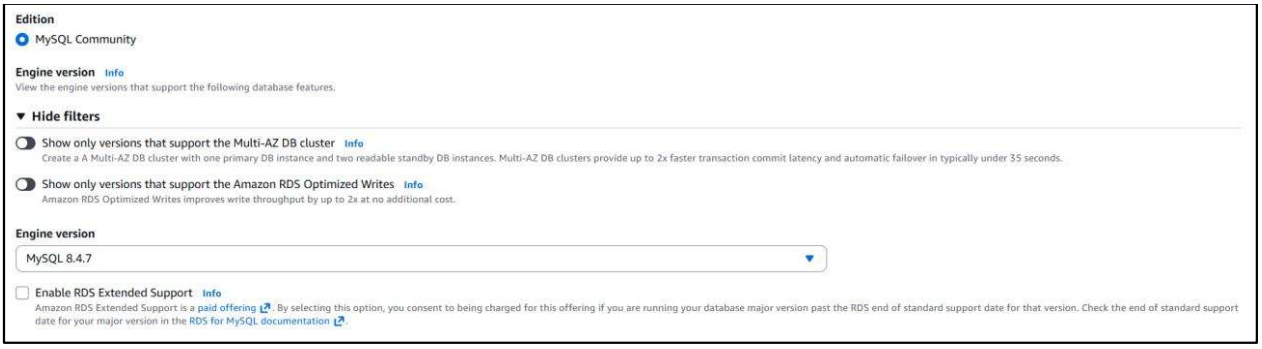
- Inbound Rule :
    - MySql : 3306 (Allow)



Step 4 : Create Database

- AWS Console → RDS → Create DB
- Engine: MySQL
- Template: Free tier
- DB name: ordersdb
- Username: admin
- Password: Strong Password
- Public access: NO
- VPC: same as EC2
- Subnet group: Created above





Templates

Choose a sample template to meet your use case.

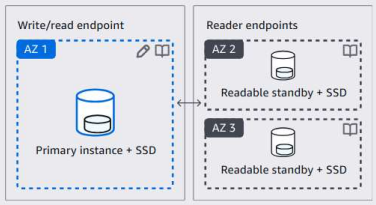
- ○ **Production**
  Use defaults for high availability and fast, consistent performance.
- ○ **Dev/Test**
  This instance is intended for development use outside of a production environment.
- ◉ **Sandbox**
  To develop new applications, test existing applications, or gain hands-on experience with Amazon RDS.

Availability and durability

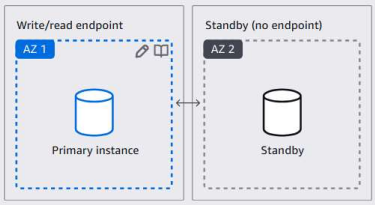
Deployment options Info

Choose the deployment option that provides the availability and durability needed for your use case. AWS is committed to a certain level of uptime depending on the deployment option you choose. Learn more in the Amazon RDS service level agreement (SLA).

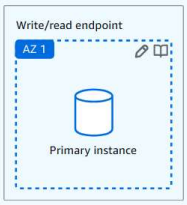
- ○ **Multi-AZ DB cluster deployment (3 instances)**
  Creates a primary DB instance with two readable standbys in separate Availability Zones. This setup provides:
  - 99.95% uptime
  - Redundancy across Availability Zones
  - Increased read capacity
  - Reduced write latency



- ○ **Multi-AZ DB instance deployment (2 instances)**
  Creates a primary DB instance with a non-readable standby instance in a separate Availability Zone. This setup provides:
  - 99.95% uptime
  - Redundancy across Availability Zones



- ◉ **Single-AZ DB instance deployment (1 instance)**
  Creates a single DB instance without standby instances. This setup provides:
  - 99.5% uptime
  - No data redundancy



Settings

DB instance identifier Info

Type a name for your DB instance. The name must be unique across all DB instances owned by your AWS account in the current AWS Region.

ordersdb

The DB instance identifier is case-insensitive, but is stored as all lowercase (as in "mydbinstance"). Constraints: 1 to 63 alphanumeric characters or hyphens. First character must be a letter. Can't contain two consecutive hyphens. Can't end with a hyphen.

Credentials Settings

Master username Info

Type a login ID for the master user of your DB instance.

admin

1 to 16 alphanumeric characters. The first character must be a letter.

Credentials management

You can use AWS Secrets Manager or manage your master user credentials.

- ○ **Managed in AWS Secrets Manager - *most secure***
  RDS generates a password for you and manages it throughout its lifecycle using AWS Secrets Manager.
- ◉ **Self managed**
  Create your own password or have RDS create a password that you manage.

☐ **Auto generate password**
Amazon RDS can generate a password for you, or you can specify your own password.

Master password | Info

••••••••

Password strength Weak

Minimum constraints: At least 8 printable ASCII characters. Can't contain any of the following symbols: / ' " @

Confirm master password | Info

••••••••

▼ **Additional credentials settings**

Database authentication options Info

- ◉ **Password authentication**
  Authenticates using database passwords.
- ○ **Password and IAM database authentication**
  Authenticates using the database password and user credentials through AWS IAM users and roles.
- ○ **Password and Kerberos authentication**
  Choose a directory in which you want to allow authorized users to authenticate with this DB instance using Kerberos Authentication.

Instance configuration

The DB instance configuration options below are limited to those supported by the engine that you selected above.

DB instance class | Info

▼ **Hide filters**

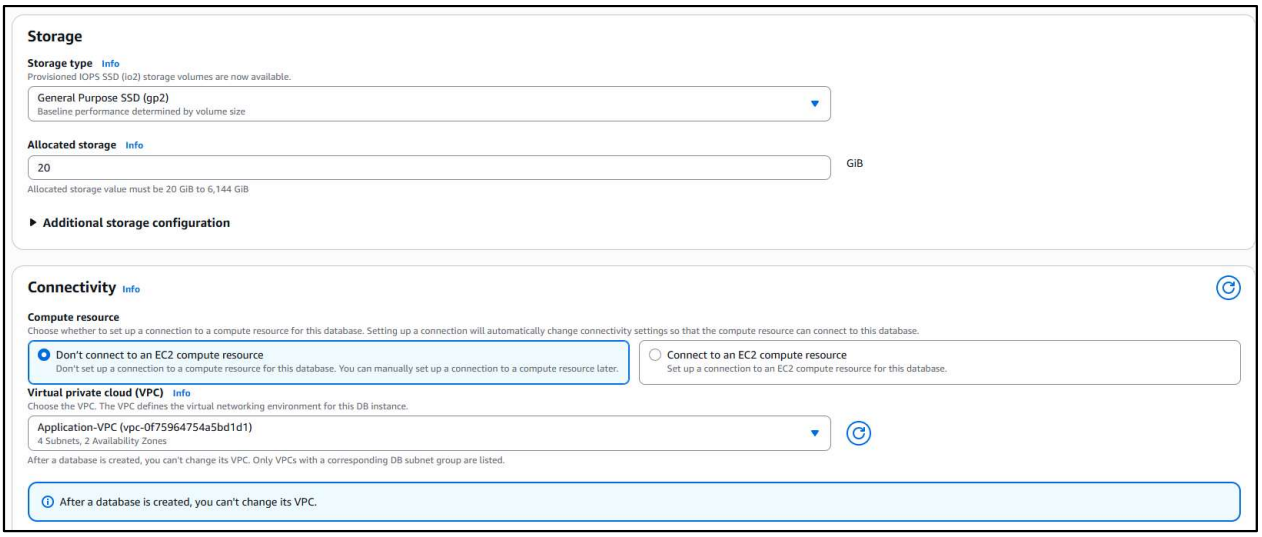
☐ **Show instance classes that support Amazon RDS Optimized Writes** Info
Amazon RDS Optimized Writes improves write throughput by up to 2x at no additional cost.

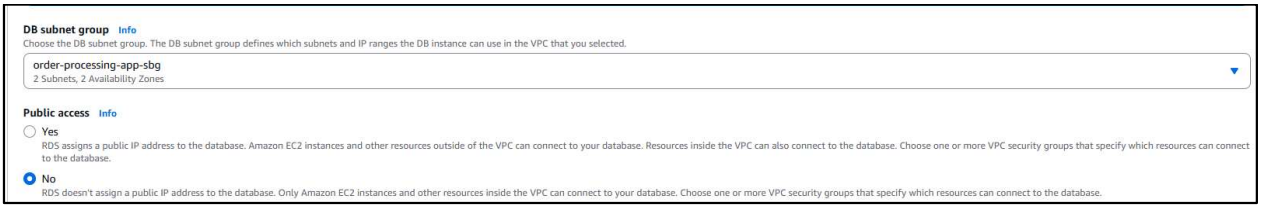
☐ **Include previous generation classes**

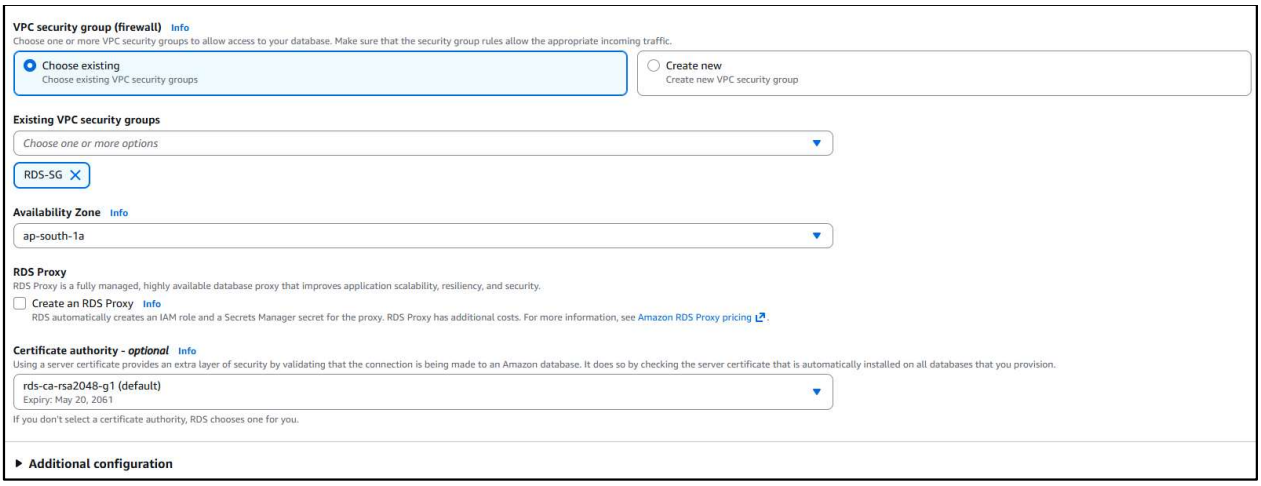
- ○ Standard classes (includes m classes)
- ○ Memory optimized classes (includes r and x classes)
- ◉ Burstable classes (includes t classes)

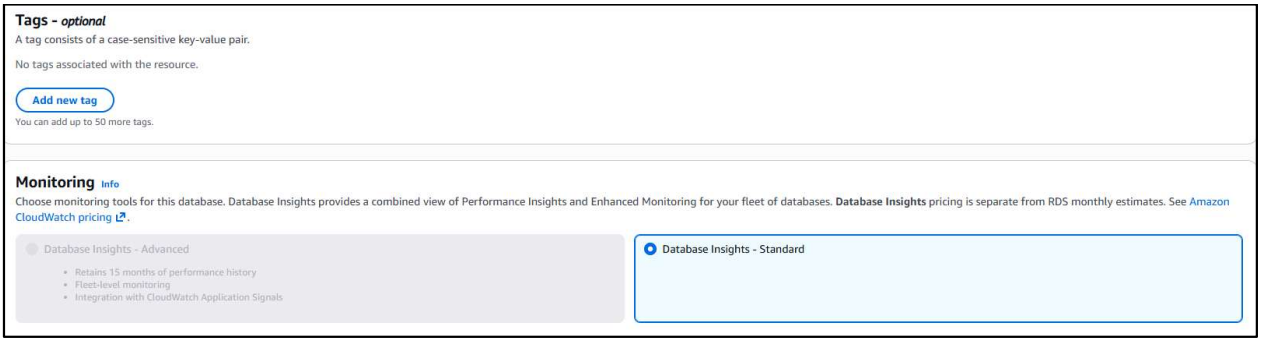
Instance type

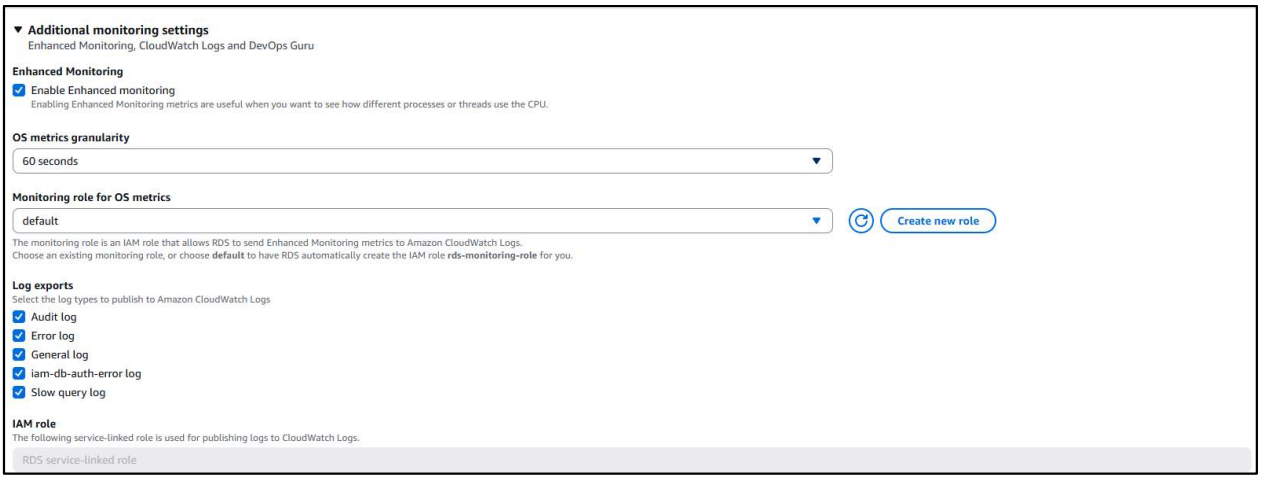
db.t4g.micro
2 vCPUs 1 GiB RAM EBS Bandwidth: Up to 2,085 Mbps Network: Up to 5 Gbps

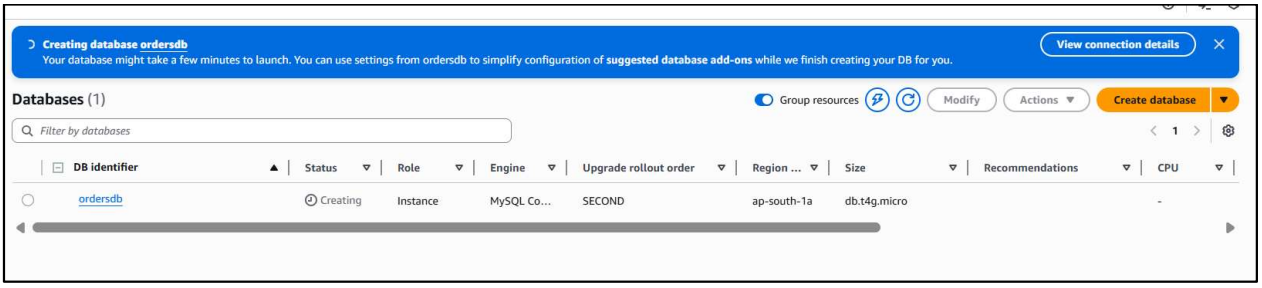

Storage
Storage type Info
Provisioned IOPS SSD (io2) storage volumes are now available.
General Purpose SSD (gp2)
Baseline performance determined by volume size
Allocated storage Info
20
GiB
Allocated storage value must be 20 GiB to 6,144 GiB
Additional storage configuration
Connectivity Info
Compute resource
Choose whether to set up a connection to a compute resource for this database. Setting up a connection will automatically change connectivity settings so that the compute resource can connect to this database.
Don't connect to an EC2 compute resource
Don't set up a connection to a compute resource for this database. You can manually set up a connection to a compute resource later.
Connect to an EC2 compute resource
Set up a connection to an EC2 compute resource for this database.
Virtual private cloud (VPC) Info
Choose the VPC. The VPC defines the virtual networking environment for this DB instance.
Application-VPC (vpc-0f75964754a5bd1d1)
4 Subnets, 2 Availability Zones
After a database is created, you can't change its VPC. Only VPCs with a corresponding DB subnet group are listed.
After a database is created, you can't change its VPC.


DB subnet group Info
Choose the DB subnet group. The DB subnet group defines which subnets and IP ranges the DB instance can use in the VPC that you selected.
order-processing-app-sbg
2 Subnets, 2 Availability Zones
Public access Info
Yes
RDS assigns a public IP address to the database. Amazon EC2 instances and other resources outside of the VPC can connect to your database. Resources inside the VPC can also connect to the database. Choose one or more VPC security groups that specify which resources can connect to the database.
No
RDS doesn't assign a public IP address to the database. Only Amazon EC2 instances and other resources inside the VPC can connect to your database. Choose one or more VPC security groups that specify which resources can connect to the database.


VPC security group (firewall) Info
Choose one or more VPC security groups to allow access to your database. Make sure that the security group rules allow the appropriate incoming traffic.
Choose existing
Choose existing VPC security groups
Create new
Create new VPC security group
Existing VPC security groups
Choose one or more options
RDS-SG
Availability Zone Info
ap-south-1a
RDS Proxy
RDS Proxy is a fully managed, highly available database proxy that improves application scalability, resiliency, and security.
Create an RDS Proxy Info
RDS automatically creates an IAM role and a Secrets Manager secret for the proxy. RDS Proxy has additional costs. For more information, see Amazon RDS Proxy pricing.
Certificate authority - optional Info
Using a server certificate provides an extra layer of security by validating that the connection is being made to an Amazon database. It does so by checking the server certificate that is automatically installed on all databases that you provision.
rds-ca-rsa2048-g1 (default)
Expiry: May 20, 2061
If you don't select a certificate authority, RDS chooses one for you.
Additional configuration


Tags - optional
A tag consists of a case-sensitive key-value pair.
No tags associated with the resource.
Add new tag
You can add up to 50 more tags.
Monitoring Info
Choose monitoring tools for this database. Database Insights provides a combined view of Performance Insights and Enhanced Monitoring for your fleet of databases. Database Insights pricing is separate from RDS monthly estimates. See Amazon CloudWatch pricing.
Database Insights - Advanced
Retains 15 months of performance history
Fleet-level monitoring
Integration with CloudWatch Application Signals
Database Insights - Standard

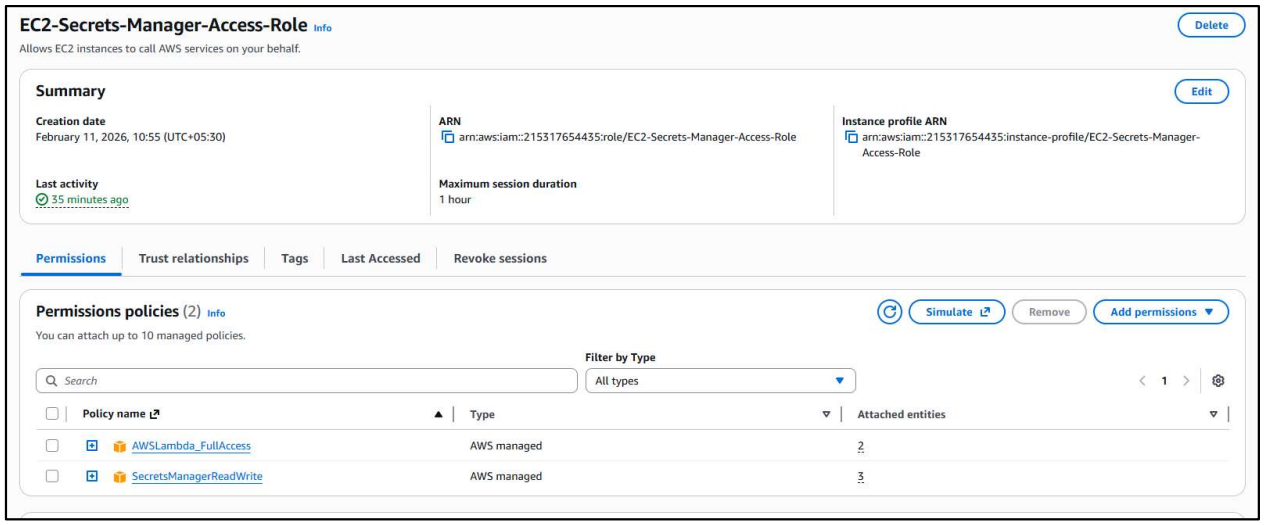

Additional monitoring settings
Enhanced Monitoring, CloudWatch Logs and DevOps Guru
Enhanced Monitoring
Enable Enhanced monitoring
Enabling Enhanced Monitoring metrics are useful when you want to see how different processes or threads use the CPU.
OS metrics granularity
60 seconds
Monitoring role for OS metrics
default
Create new role
The monitoring role is an IAM role that allows RDS to send Enhanced Monitoring metrics to Amazon CloudWatch Logs.
Choose an existing monitoring role, or choose default to have RDS automatically create the IAM role rds-monitoring-role for you.
Log exports
Select the log types to publish to Amazon CloudWatch Logs
Audit log
Error log
General log
iam-db-auth-error log
Slow query log
IAM role
The following service-linked role is used for publishing logs to CloudWatch Logs.
RDS service-linked role


Creating database ordersdb
Your database might take a few minutes to launch. You can use settings from ordersdb to simplify configuration of suggested database add-ons while we finish creating your DB for you.
View connection details
Databases (1)
Group resources
Modify
Actions
Create database
Filter by databases
1
DB identifier
Status
Role
Engine
Upgrade rollout order
Region ...
Size
Recommendations
CPU
ordersdb
Creating
Instance
MySQL Co...
SECOND
ap-south-1a
db.t4g.micro
-

Step 5 : Create EC2

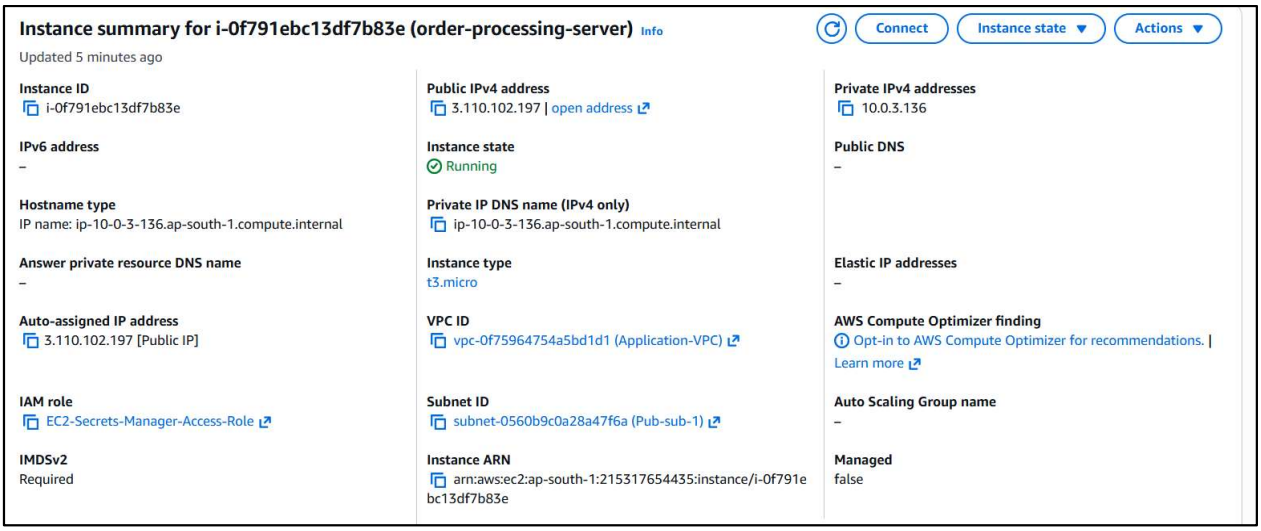
Step 5.1 : Create IAM Role

- Name : EC2-Secrets-Manager-Access-Role
- Permissions :
    - SecretManagerReadWrite
    - AWSLambda_FullAccess
    - AmazonS3FullAccess
- Attach to EC2.



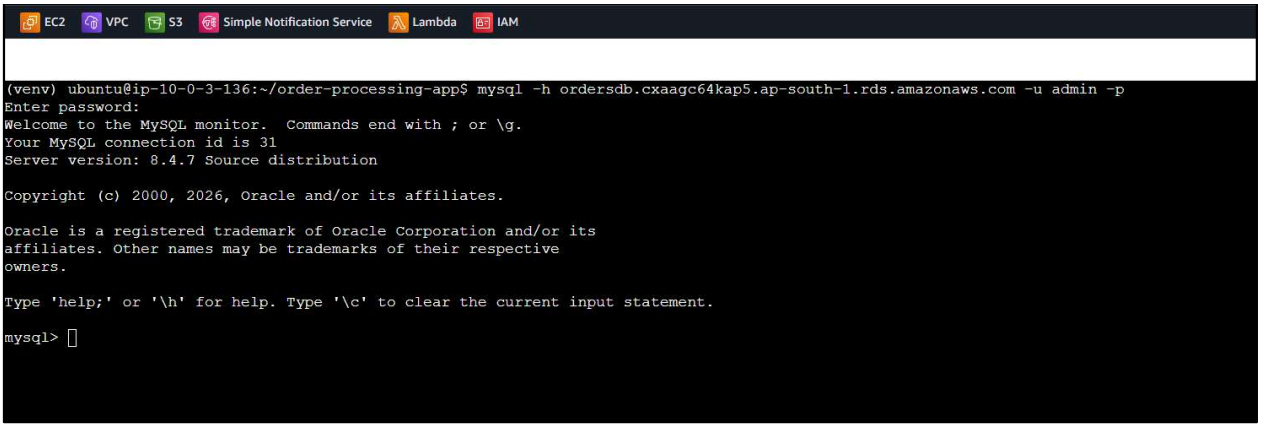
Step 5.2 : Create EC2

- Name : order-app-server
- Choose OS (AMI) : Ubuntu Server 22.04 LTS (easy for beginners)
- Instance type: t3.micro
- Security group: Security group created above for EC2
- Launch



Step 5.3 : Connect Database

- Connect RDS database through EC2.
  - Mysql –h <RDS-endpoint> -u <username> –p <password>



- Verify databases
- Show databases;

- Create new database for application.
- CREATE DATABASE <database_name>;

```
mysql> CREATE DATABASE orders;
Query OK, 1 row affected (0.09 sec)
```

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| orders             |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.06 sec)

mysql>
```

- Create table
  - Use <database_name>
- Write table creation query;

```
mysql> USE orders;
Database changed
mysql>
mysql> CREATE TABLE orders (
    ->   order_id INT AUTO_INCREMENT PRIMARY KEY,
    ->   customer_name VARCHAR(100),
    ->   amount DECIMAL(10,2),
    ->   status VARCHAR(20),
    ->   created_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP
    -> );
Query OK, 0 rows affected (0.04 sec)
```

```
mysql> show tables;
+------------------+
| Tables_in_orders |
+------------------+
| orders           |
+------------------+
1 row in set (0.03 sec)
```

```
mysql> desc orders;
+--------------+--------------+------+-----+-------------------+-------------------+
| Field        | Type         | Null | Key | Default           | Extra             |
+--------------+--------------+------+-----+-------------------+-------------------+
| order_id     | int          | NO   | PRI | NULL              | auto_increment    |
| customer_name | varchar(100) | YES  |     | NULL              |                   |
| amount       | decimal(10,2) | YES  |     | NULL              |                   |
| status       | varchar(20)  | YES  |     | NULL              |                   |
| created_time | timestamp    | YES  |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
+--------------+--------------+------+-----+-------------------+-------------------+
5 rows in set (0.07 sec)
```

Step 5.4 : Install dependencies on Server.

- Install python and check version
  - sudo apt install python3 –y
  - Python --version

```
(venv) ubuntu@ip-10-0-3-136:~/order-processing-app$ python --version
Python 3.12.3
(venv) ubuntu@ip-10-0-3-136:~/order-processing-app$
```

- Create virtual environment.
  - Python3 –m venv venv
- Activate virtual environment
  - Source venv/bin/activate

```
ubuntu@ip-10-0-3-136:~/order-processing-app$ sudo chown -R ubuntu:ubuntu ~/order-processing-app
ubuntu@ip-10-0-3-136:~/order-processing-app$ python3 -m venv venv
ubuntu@ip-10-0-3-136:~/order-processing-app$ source venv/bin/activate
(venv) ubuntu@ip-10-0-3-136:~/order-processing-app$ ls
venv
```

- Install flask
  - Pip install flask
  - pip3 install flask boto3
  - Pip install flask pymysql boto3

```
ubuntu@ip-10-0-3-136:~/order-processing-app$ source venv/bin/activate
(venv) ubuntu@ip-10-0-3-136:~/order-processing-app$ ls
venv
(venv) ubuntu@ip-10-0-3-136:~/order-processing-app$ pip install flask
Collecting flask
  Using cached flask-3.1.2-py3-none-any.whl.metadata (3.2 kB)
Collecting blinker>=1.9.0 (from flask)
  Using cached blinker-1.9.0-py3-none-any.whl.metadata (1.6 kB)
Collecting click>=8.1.3 (from flask)
  Using cached click-8.3.1-py3-none-any.whl.metadata (2.6 kB)
Collecting itsdangerous>=2.2.0 (from flask)
  Using cached itsdangerous-2.2.0-py3-none-any.whl.metadata (1.9 kB)
Collecting jinja2>=3.1.2 (from flask)
  Using cached jinja2-3.1.6-py3-none-any.whl.metadata (2.9 kB)
Collecting markupsafe>=2.1.1 (from flask)
  Using cached markupsafe-3.0.3-cp312-cp312-manylinux2014_x86_64.manylinux_2_17_x86_64.manylinux_2_28_x86_64.whl.metadata (2.7 kB)
Collecting werkzeug>=3.1.0 (from flask)
  Using cached werkzeug-3.1.5-py3-none-any.whl.metadata (4.0 kB)
Using cached flask-3.1.2-py3-none-any.whl (103 kB)
```

```
(venv) ubuntu@ip-10-0-3-136:~/order-processing-app$ pip install flask pymysql boto3
Requirement already satisfied: flask in ./venv/lib/python3.12/site-packages (3.1.2)
Collecting pymysql
  Using cached pymysql-1.1.2-py3-none-any.whl.metadata (4.3 kB)
Collecting boto3
  Downloading boto3-1.42.47-py3-none-any.whl.metadata (6.8 kB)
Requirement already satisfied: blinker>=1.9.0 in ./venv/lib/python3.12/site-packages (from flask) (1.9.0)
Requirement already satisfied: click>=8.1.3 in ./venv/lib/python3.12/site-packages (from flask) (8.3.1)
Requirement already satisfied: itsdangerous>=2.2.0 in ./venv/lib/python3.12/site-packages (from flask) (2.2.0)
Requirement already satisfied: jinja2>=3.1.2 in ./venv/lib/python3.12/site-packages (from flask) (3.1.6)
Requirement already satisfied: markupsafe>=2.1.1 in ./venv/lib/python3.12/site-packages (from flask) (3.0.3)
Requirement already satisfied: werkzeug>=3.1.0 in ./venv/lib/python3.12/site-packages (from flask) (3.1.5)
Collecting botocore<1.43.0,>=1.42.47 (from boto3)
  Downloading botocore-1.42.47-py3-none-any.whl.metadata (5.9 kB)
Collecting jmespath<2.0.0,>=0.7.1 (from boto3)
  Using cached jmespath-1.1.0-py3-none-any.whl.metadata (7.6 kB)
Collecting s3transfer<0.17.0,>=0.16.0 (from boto3)
  Using cached s3transfer-0.16.0-py3-none-any.whl.metadata (1.7 kB)
```

Step 5.5 : Create Application and run

- Create application file
- Sudo nano app.py
- Write the code

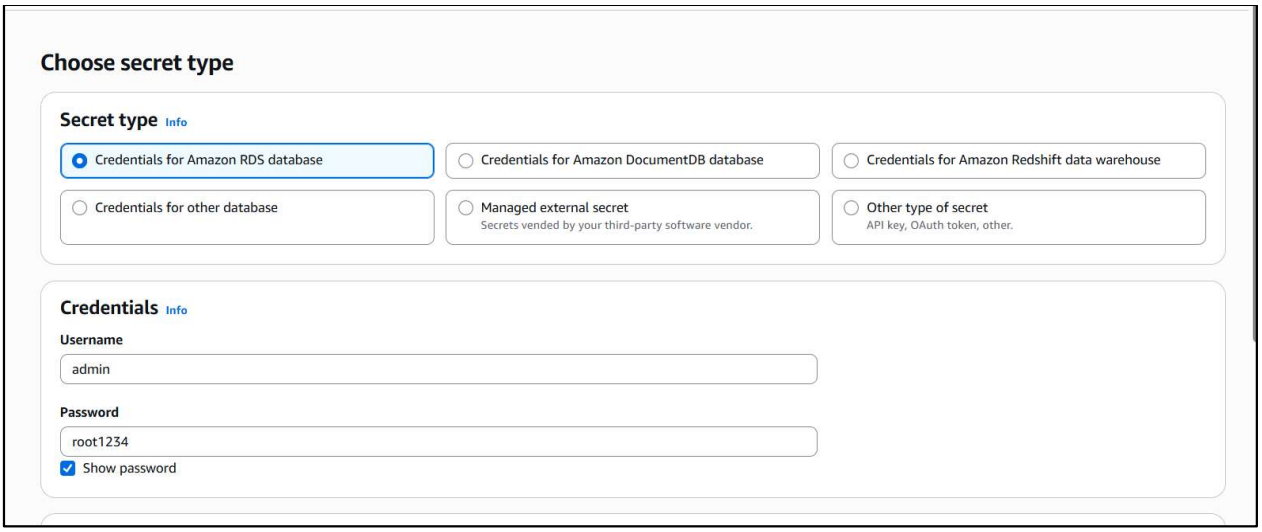
```
(venv) ubuntu@ip-10-0-3-136:~/order-processing-app$ nano app.py
(venv) ubuntu@ip-10-0-3-136:~/order-processing-app$
```

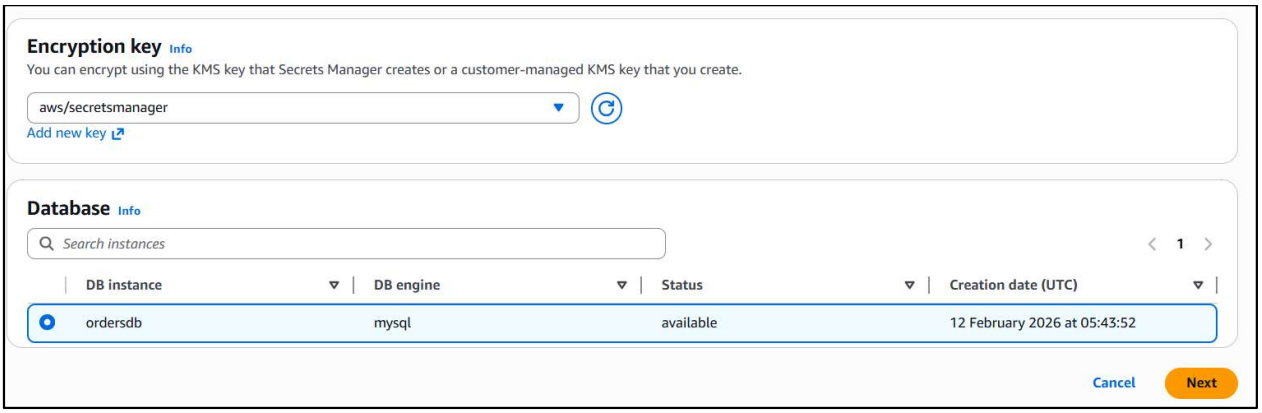
EC2 VPC S3 Simple Notification Service Lambda IAM

```
  GNU nano 7.2                                                    app.py *
from flask import Flask, request, render_template_string
import boto3
import json
import pymysql

app = Flask(__name__)

lambda_client = boto3.client('lambda')

HTML = """
<!DOCTYPE html>
<html>
<head>
<title>Order Processing System</title>
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body class="bg-light">

<div class="container mt-5">
<div class="card shadow-lg p-4">

<h2 class="mb-4 text-center">Automated Order Processing</h2>

<form method="post">
<div class="mb-3">
```

Step 6 : Configure Secrets in secret manager.

- AWS Console → Secrets Manager
- Create secret:
    - Type: Credentials for RDS
    - username: admin
    - password: <Password>
    - host: RDS endpoint
    - db: ordersdb
- Secret name:
    - order-db-secret





Configure secret

Secret name and description Info

Secret name

A descriptive name that helps you find your secret later.

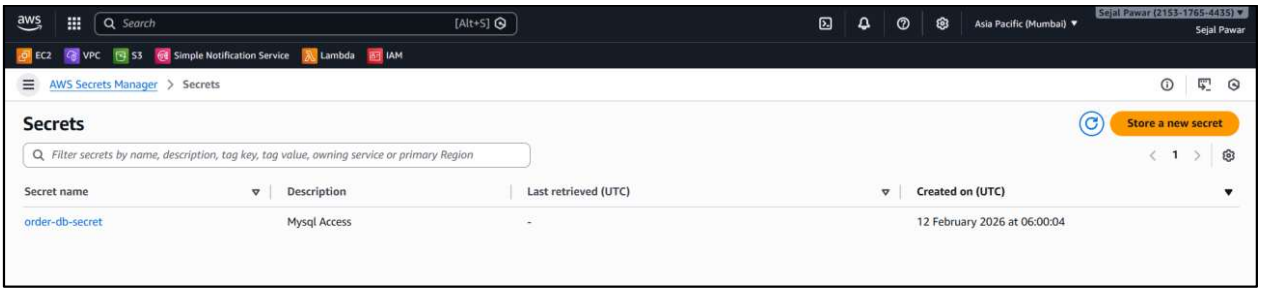
orders-db-secret

Secret name must only contain alphanumeric characters and the characters /_+=.@-

Description - *optional*

MYSQL credentials

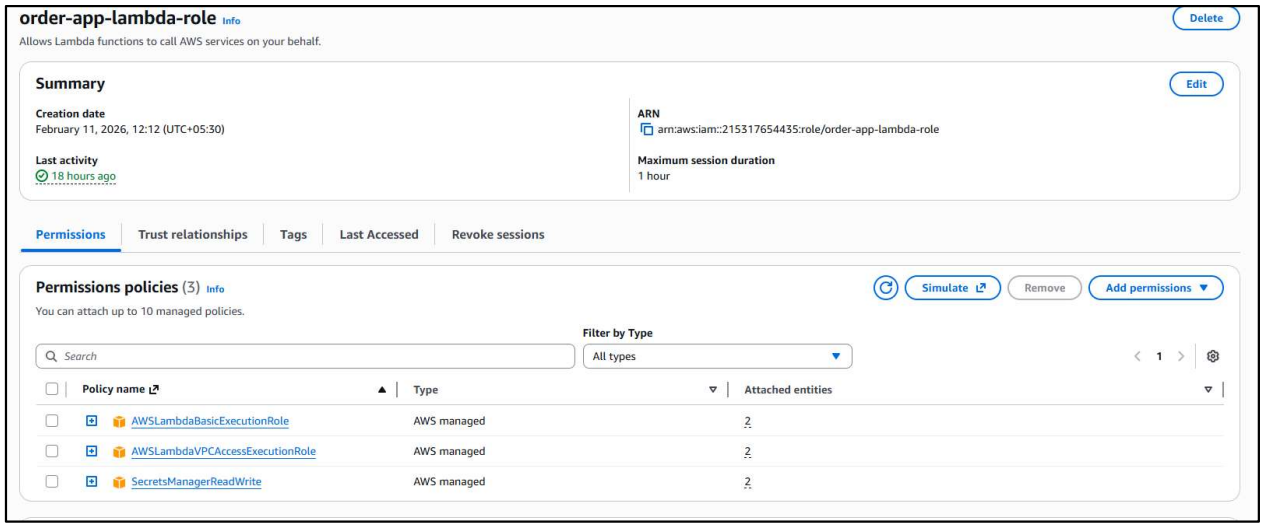
Maximum 250 characters.



Step 7 : Configure Lambda

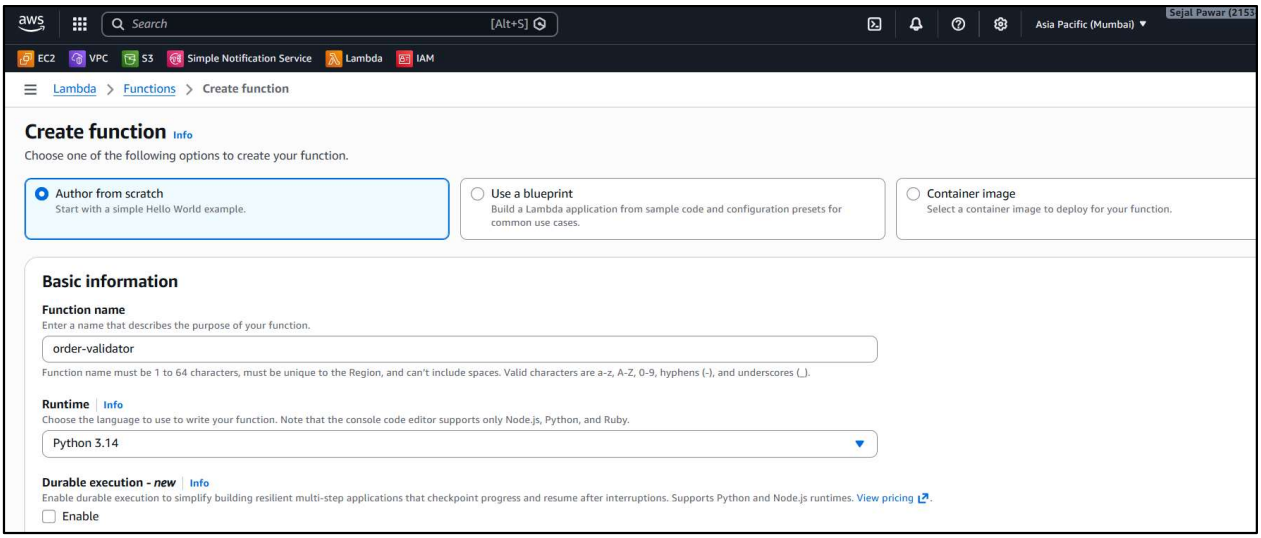
Step 7.1 : Create IAM Role for Lambda

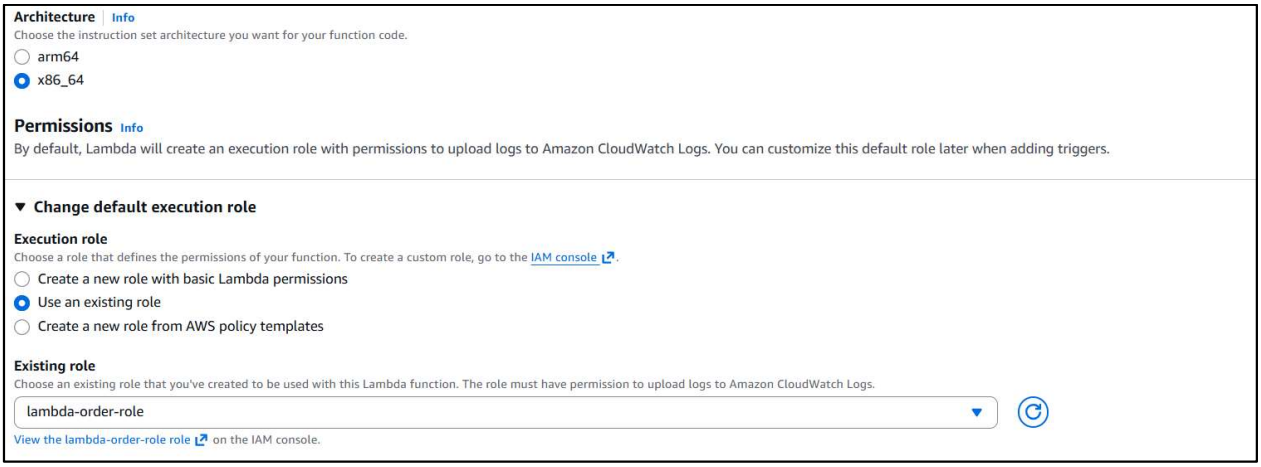
- Name : order-app-lamda-Role
- Permissions :
    - SecretManagerReadWrite
    - AWSLambdaBasicExecutionRole
    - AWSLamdaVPCAccessExecutionRole
- Attach to EC2.



Step 7.2 : Create lambda Function.

- Create Function
- Author from scratch
- Function name : order-validator
- Runtime : Python 3.14
- Attach role created above.





Step 7.3 : Create .zip file of dependencies on local system.

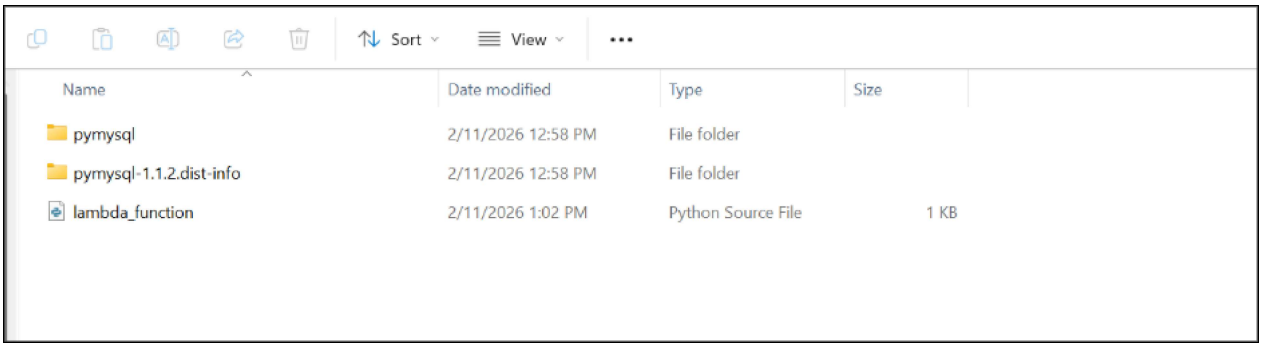
- Install python and check version.
  - Python –version

```
PS C:\Users\SejalPawar\Desktop\lambda-package> python --version
Python 3.14.3
```

- Install pymysql
  - Python –m pip install pymysql –t

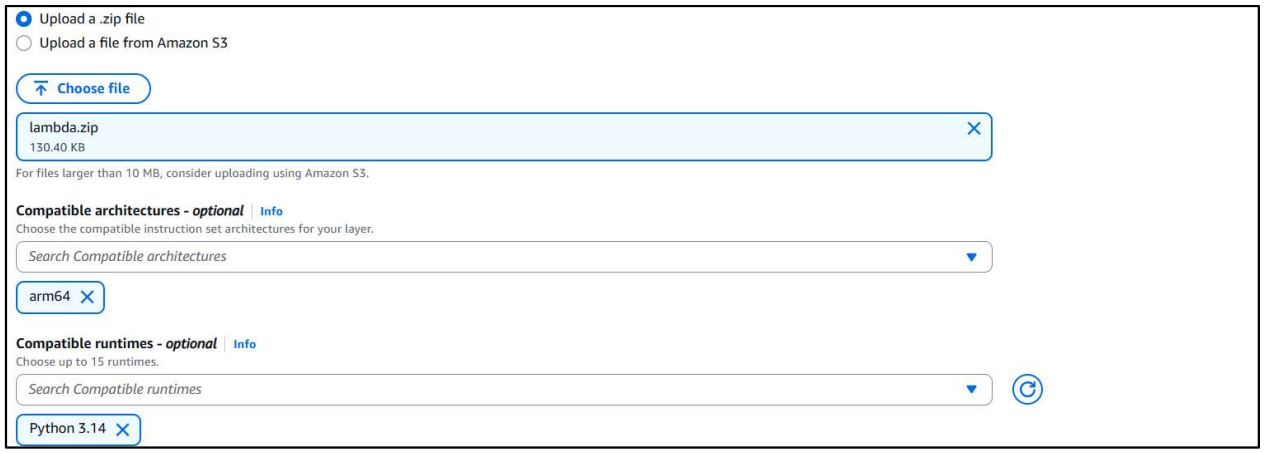
```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
powershell

Python 3.14.3
PS C:\Users\SejalPawar\Desktop\lambda-package> python -m pip install pymysql -t .
Collecting pymysql
  Downloading pymysql-1.1.2-py3-none-any.whl.metadata (4.3 kB)
Downloading pymysql-1.1.2-py3-none-any.whl (45 kB)
Installing collected packages: pymysql
Successfully installed pymysql-1.1.2
```

- Zip the folder



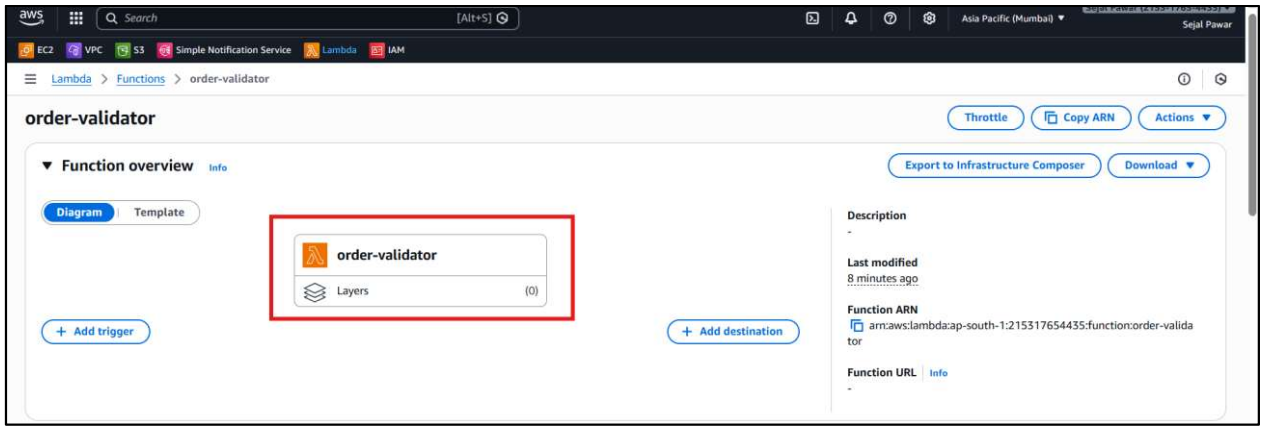
Step 7.4 : Create layer

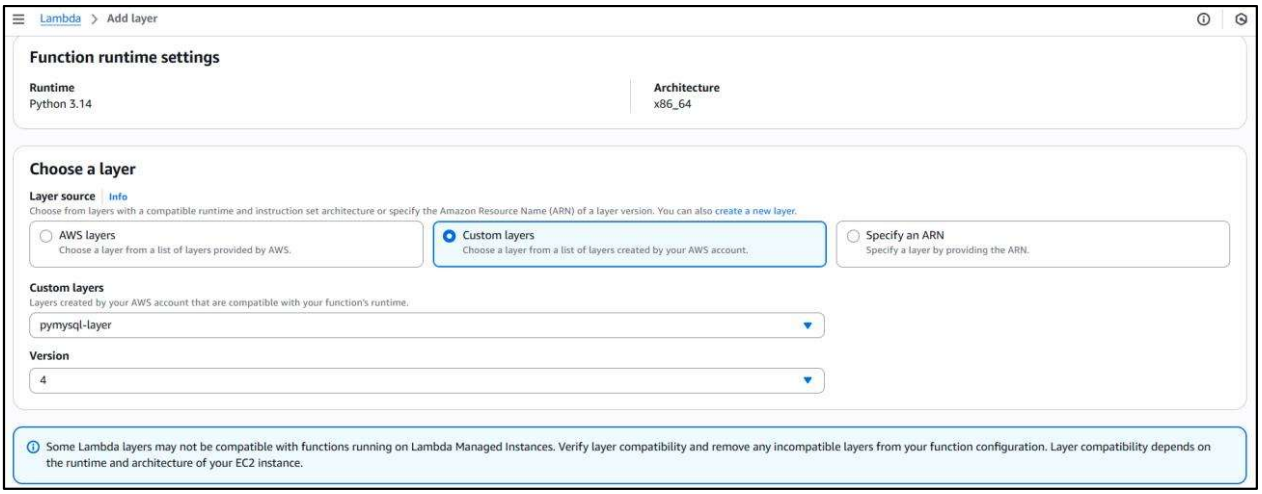
- Click on layers
- Create layer
- Name : pymysql-layer
- Upload .zip file.
- Architecture : same as Lambda function
- Runtime : same as lambda function



Step 7.5 : Attach layer to function

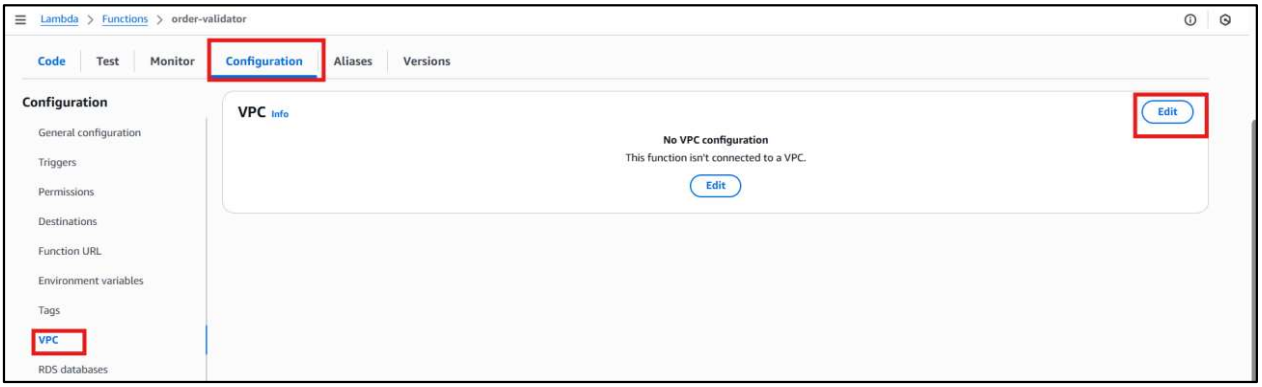
- Click on layers.
- Add layers
- Custom layers
- Select layer created above.
- Select Version.
- Create

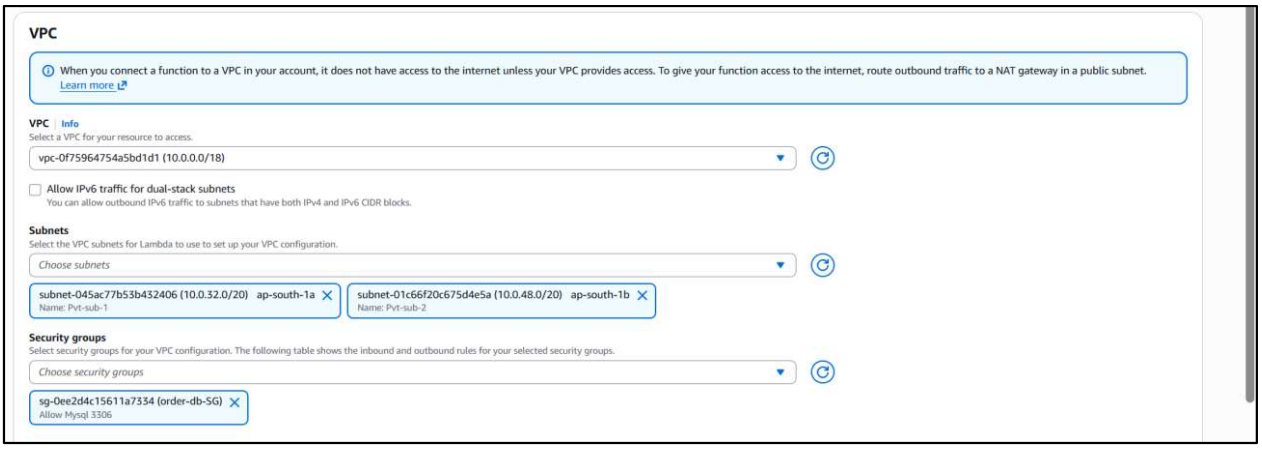




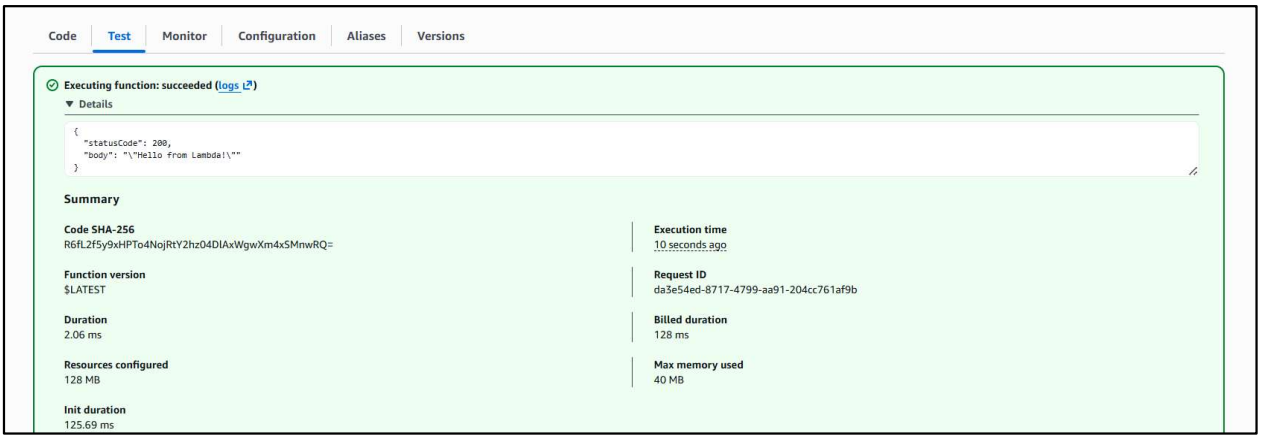
Step 7.6 : Configure networking part for Lambda.

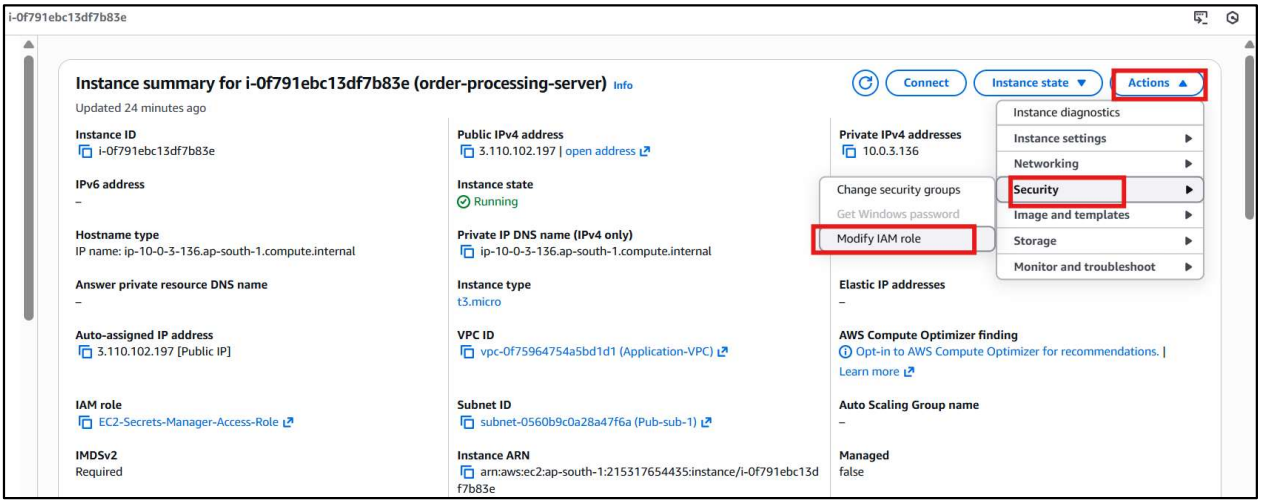
- Click on configuration
- Select VPC > Edit
- Select same VPC as RDS and EC2
- Select Private subnets
- Attach security group
    - Inbound allow : 3306 (MySql)





- Deploy and test the function.



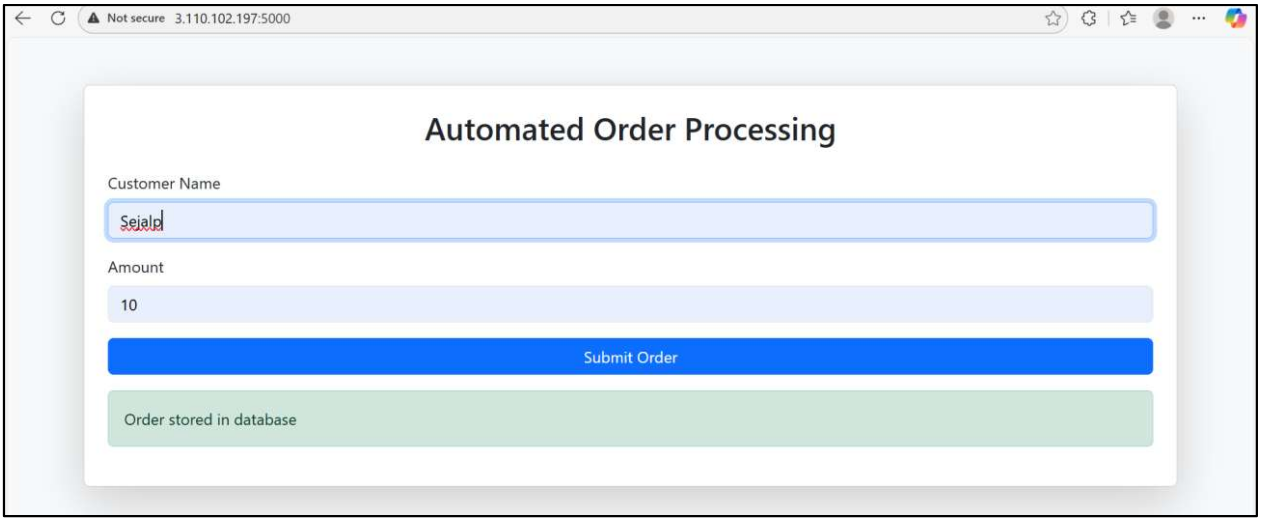



EC2 > Instances > i-0f791ebc13df7b83e > Modify IAM role
Modify IAM role Info
Attach an IAM role to your instance.
Instance ID
i-0f791ebc13df7b83e (order-processing-server)
IAM role
Select an IAM role to attach to your instance or create a new role if you haven't created any. The role you select replaces any roles that are currently attached to your instance.
EC2-Secrets-Manager-Access-Role
Create new IAM role
Cancel
Update IAM role

Step 8 : Step Run application

- Run the application on browser.
    - <public_ip:5000>

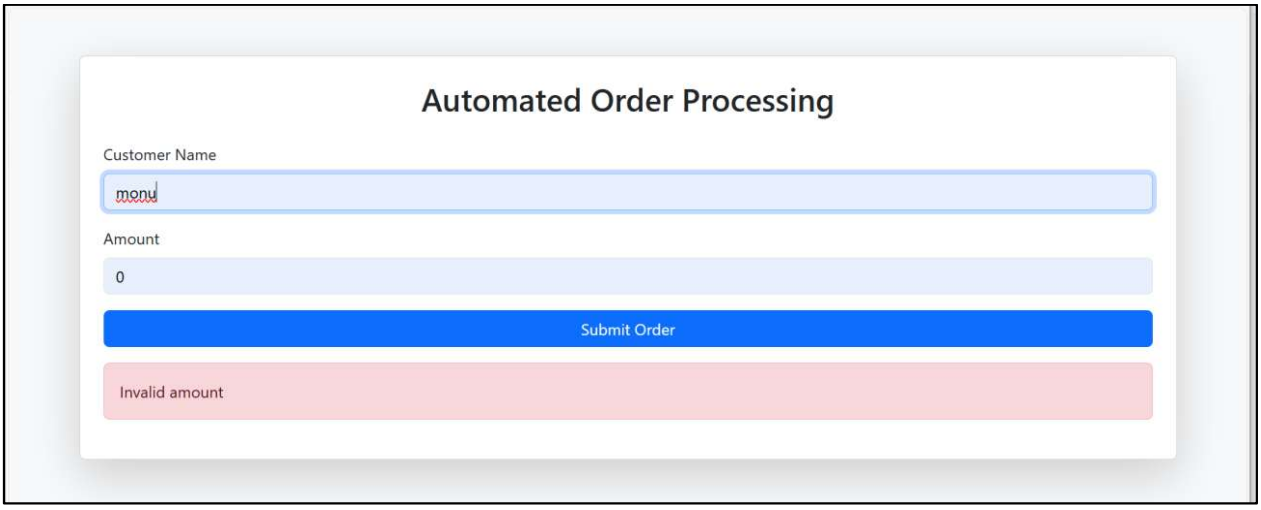
```
ubuntu@ip-10-0-3-136:~/order-processing-app$ source venv/bin/activate
(venv) ubuntu@ip-10-0-3-136:~/order-processing-app$ python3 app.py
 * Serving Flask app 'app'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://10.0.3.136:5000
Press CTRL+C to quit
110.227.211.247 - - [12/Feb/2026 11:02:37] "GET / HTTP/1.1" 200 -
Lambda Response: {'statusCode': 200, 'body': '{"status": "FAILED", "message": "Database error: \'dbname\'"}'}
110.227.211.247 - - [12/Feb/2026 11:02:43] "POST / HTTP/1.1" 200 -
Lambda Response: {'statusCode': 200, 'body': '{"status": "SUCCESS", "message": "Order stored in database"}'}
111.125.244.2 - - [12/Feb/2026 11:07:15] "POST / HTTP/1.1" 200 -
```



- Verify in Mysql server
- Order is **PROCESSED** and stored in RDS

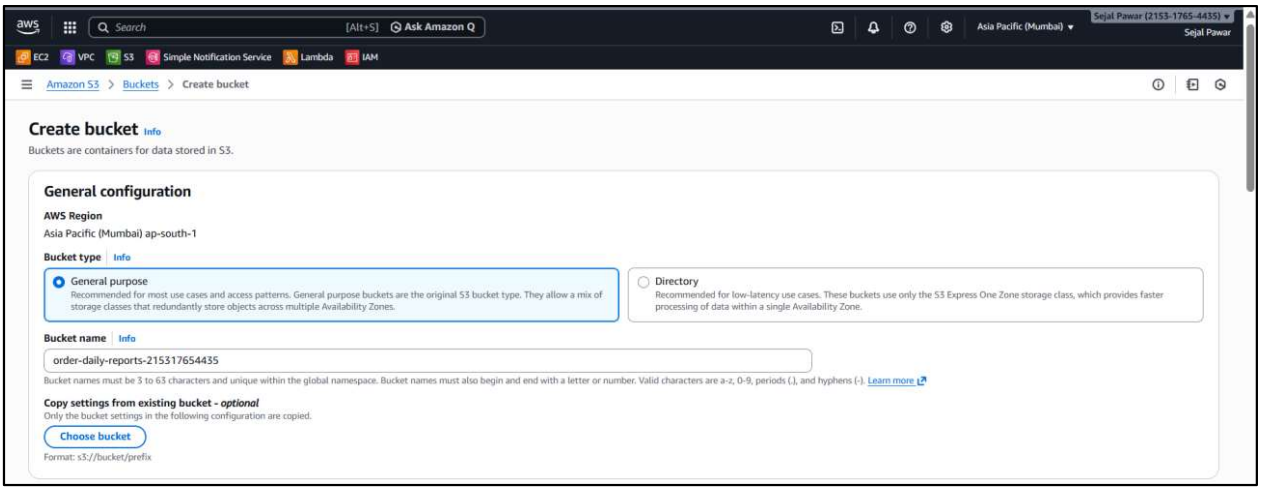
```
mysql> select * from orders;
+----------+---------------+----------+-----------+---------------------+
| order_id | customer_name | amount   | status    | created_time        |
+----------+---------------+----------+-----------+---------------------+
|        1 | SP            |  7777.00 | PROCESSED | 2026-02-12 11:07:15 |
|        2 | Sejal         | 56666.00 | PROCESSED | 2026-02-12 11:07:25 |
|        3 | Mruda         |   987.00 | PROCESSED | 2026-02-12 11:07:33 |
|        4 | Sejalp        |    10.00 | PROCESSED | 2026-02-12 11:12:47 |
+----------+---------------+----------+-----------+---------------------+
4 rows in set (0.00 sec)
```

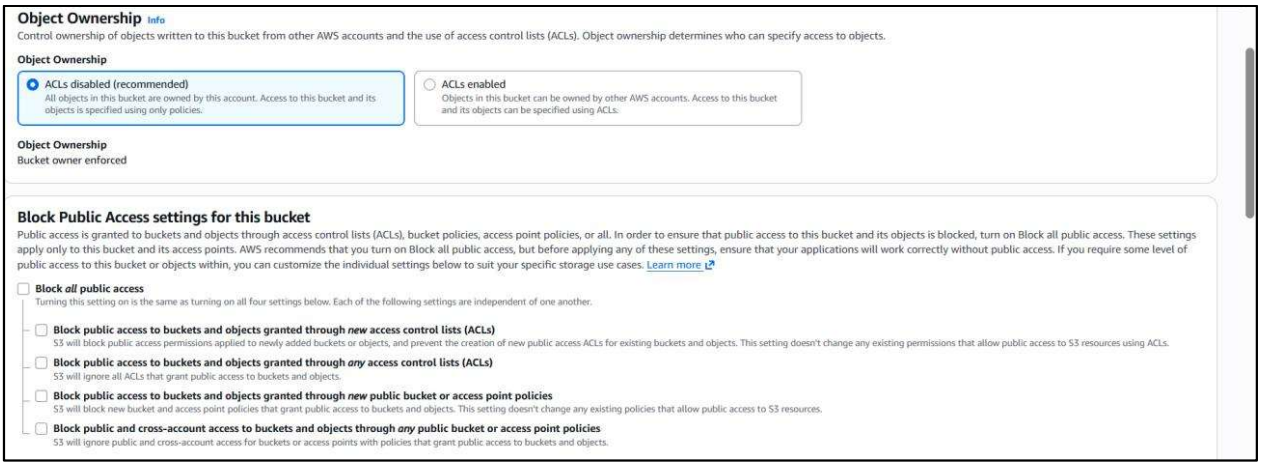
- When Value is less than or equal to Zero It shows **invalid amount**

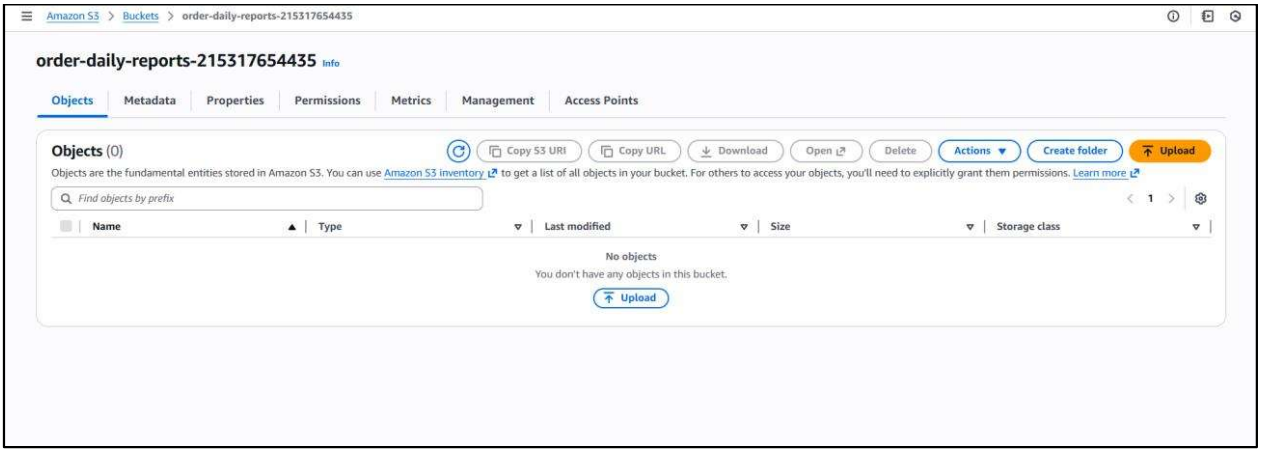


Step 9 : Create bucket

- Create bucket
- Select general purpose
- Enable public access


Search
[Alt+S]
Ask Amazon Q
Asia Pacific (Mumbai)
Sejal Pawar (2153-1765-4435)
Sejal Pawar
EC2
VPC
S3
Simple Notification Service
Lambda
IAM
Amazon S3 > Buckets > Create bucket
Create bucket Info
Buckets are containers for data stored in S3.
General configuration
AWS Region
Asia Pacific (Mumbai) ap-south-1
Bucket type Info
General purpose
Recommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.
Directory
Recommended for low-latency use cases. These buckets use only the S3 Express One Zone storage class, which provides faster processing of data within a single Availability Zone.
Bucket name Info
order-daily-reports-215317654435
Bucket names must be 3 to 63 characters and unique within the global namespace. Bucket names must also begin and end with a letter or number. Valid characters are a-z, 0-9, periods (.), and hyphens (-). Learn more
Copy settings from existing bucket - optional
Only the bucket settings in the following configuration are copied.
Choose bucket
Format: s3://bucket/prefix

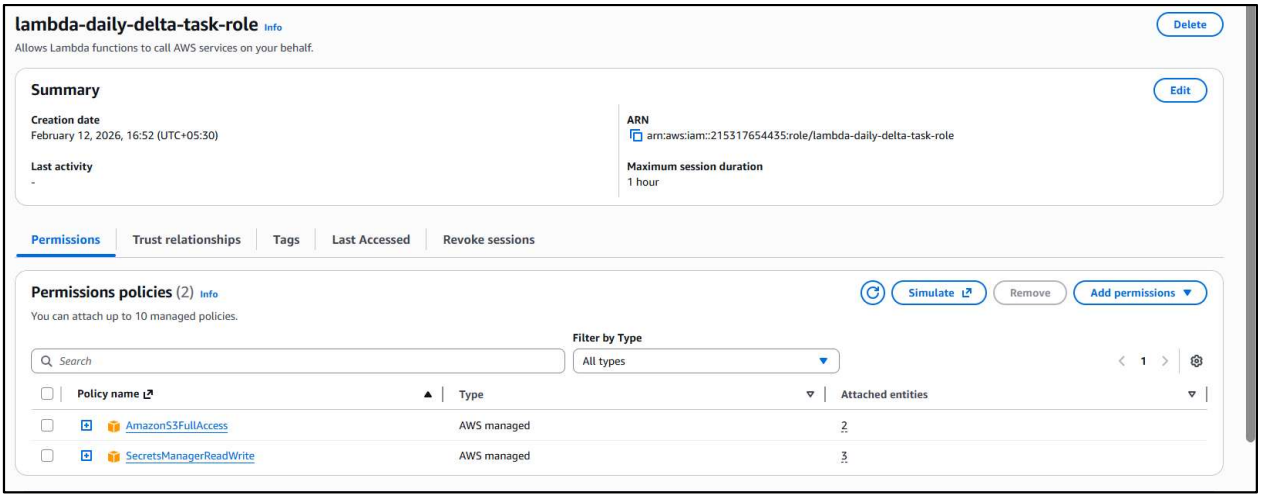

Object Ownership Info
Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.
Object Ownership
ACLs disabled (recommended)
All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.
ACLs enabled
Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.
Object Ownership
Bucket owner enforced
Block Public Access settings for this bucket
Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. Learn more
Block all public access
Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.
Block public access to buckets and objects granted through new access control lists (ACLs)
S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.
Block public access to buckets and objects granted through any access control lists (ACLs)
S3 will ignore all ACLs that grant public access to buckets and objects.
Block public access to buckets and objects granted through new public bucket or access point policies
S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.
Block public and cross-account access to buckets and objects through any public bucket or access point policies
S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.


Amazon S3 > Buckets > order-daily-reports-215317654435
order-daily-reports-215317654435 Info
Objects
Metadata
Properties
Permissions
Metrics
Management
Access Points
Objects (0)
Copy S3 URI
Copy URL
Download
Open
Delete
Actions
Create folder
Upload
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
Find objects by prefix
Name
Type
Last modified
Size
Storage class
No objects
You don't have any objects in this bucket.
Upload

Step 10 : Configure lambda function for Report generation

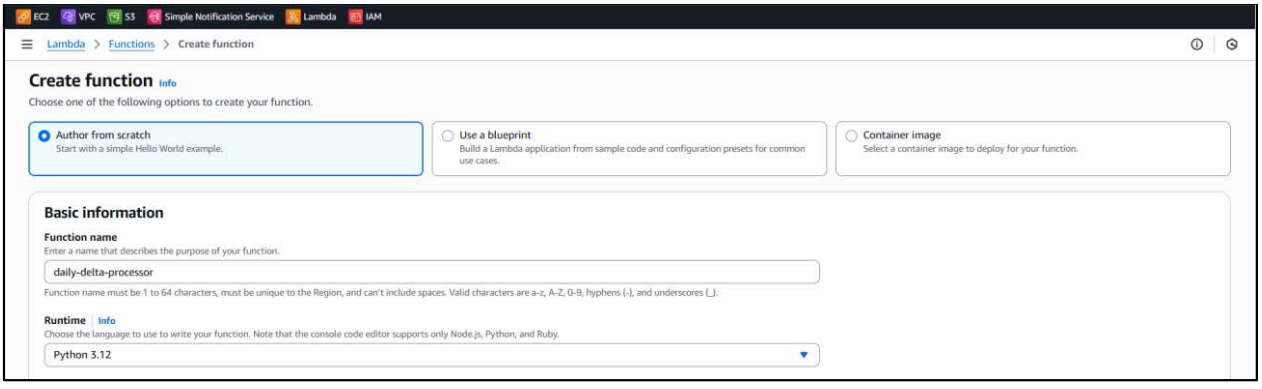
Step 10.1 : Create Role

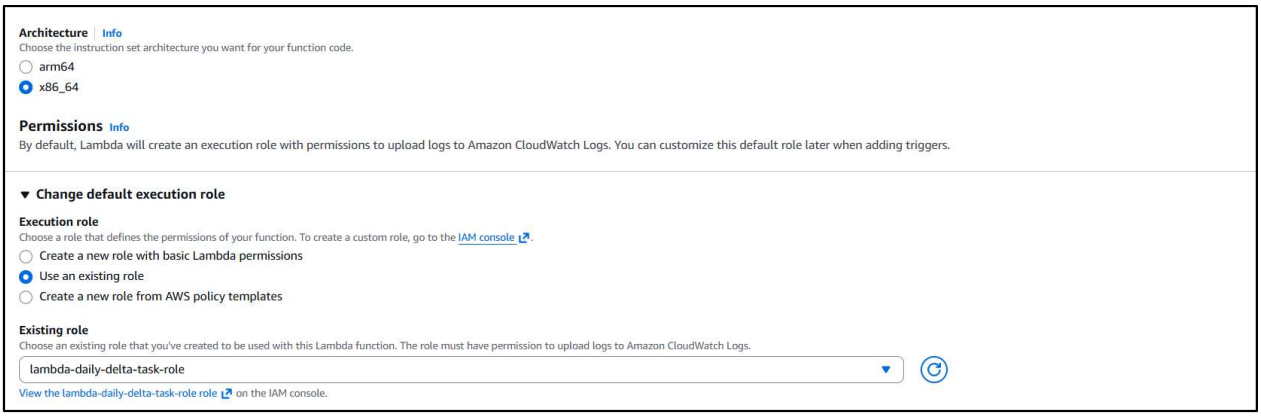
- Name : lambda-daily-delta-task-role
- Permissions:
    - AmazonS3FullAccess
    - SecretsManagerReadWrite



Step 10.2 : Create lambda function .

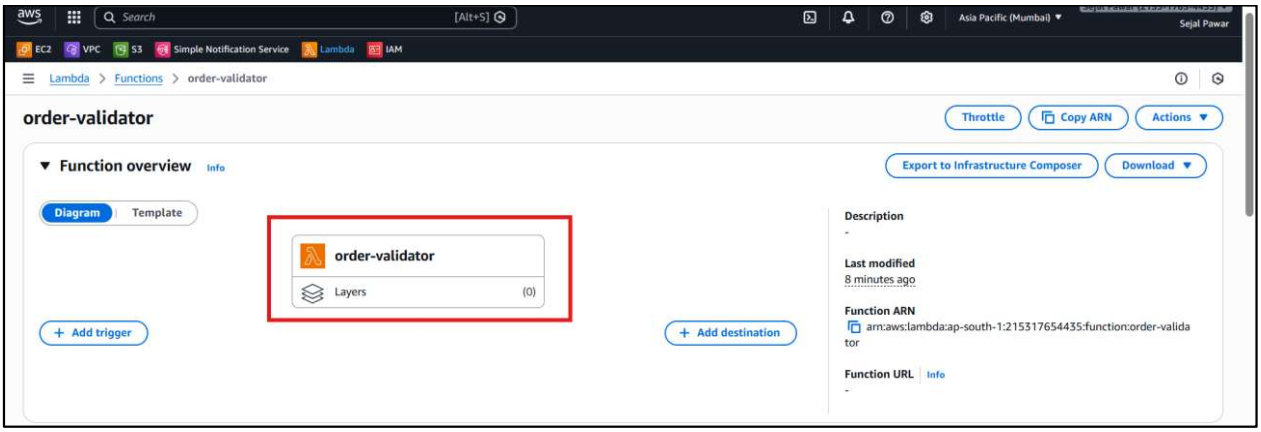
- Create Function
- Author from scratch
- Function name : daily-delta-processor
- Runtime : Python 3.12
- Attach role created above.

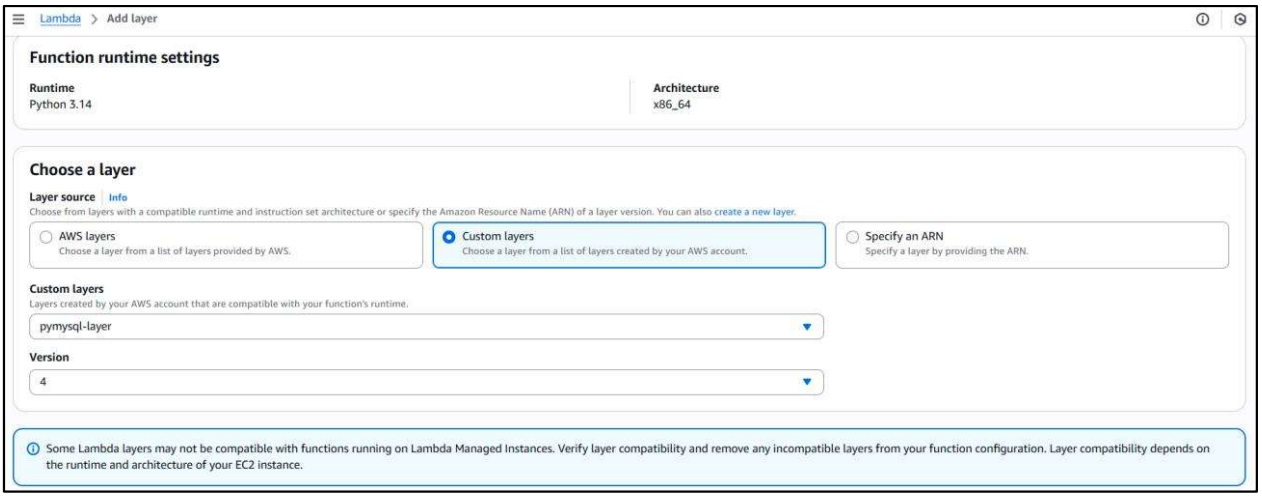




Step 10.6 : Attach layer to function

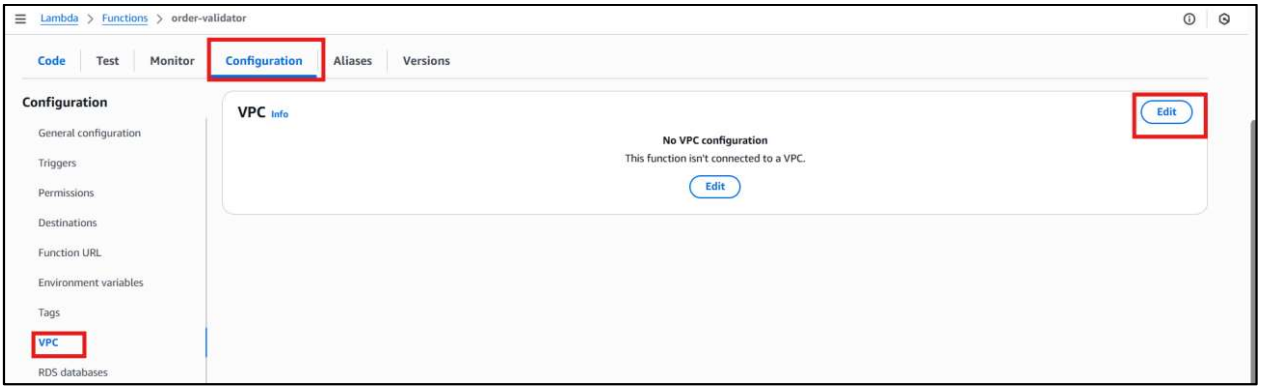
- Click on layers.
- Add layers
- Custom layers
- Select layer created above.
- Select Version.
- Create

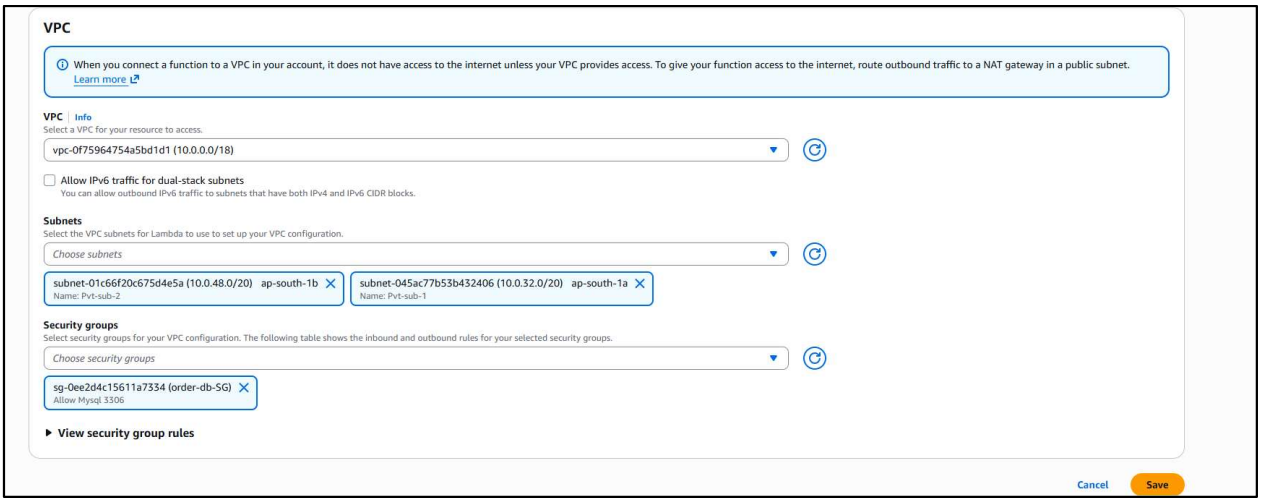


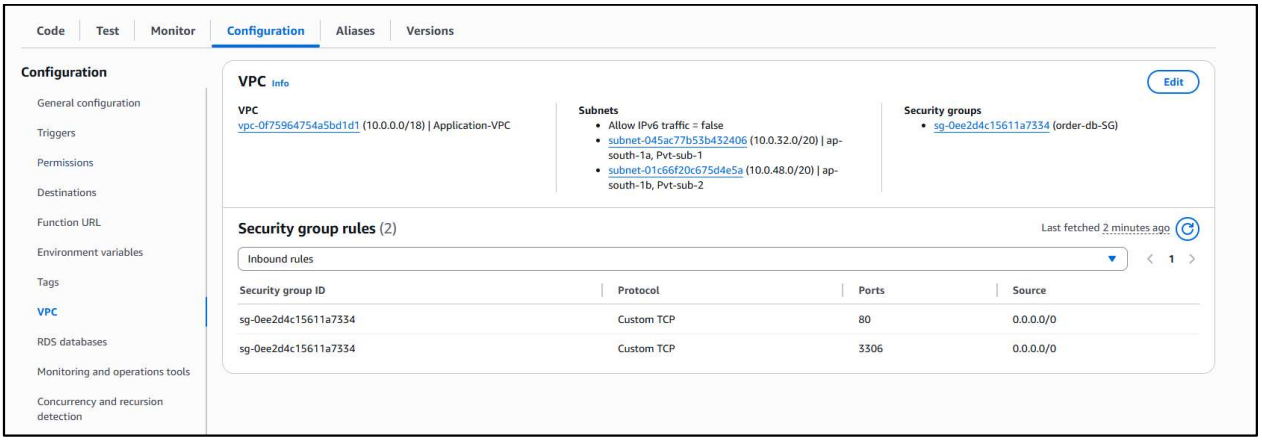


Step 10.4 : Configure networking part for Lambda.

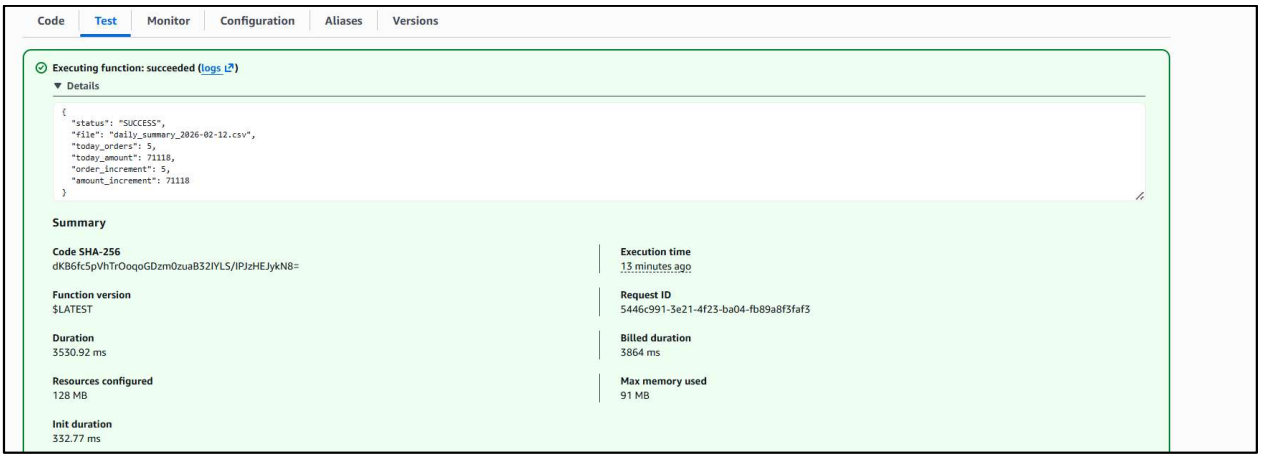
- Click on configuration
- Select VPC > Edit
- Select same VPC as RDS and EC2
- Select Private subnets
- Attach security group
    - Inbound allow : 3306 (MySql)





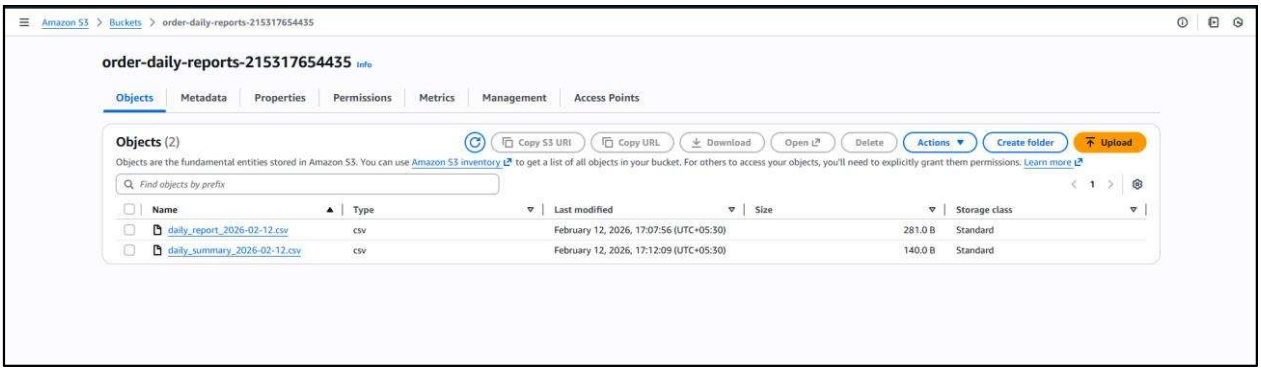


- Test lambda function.



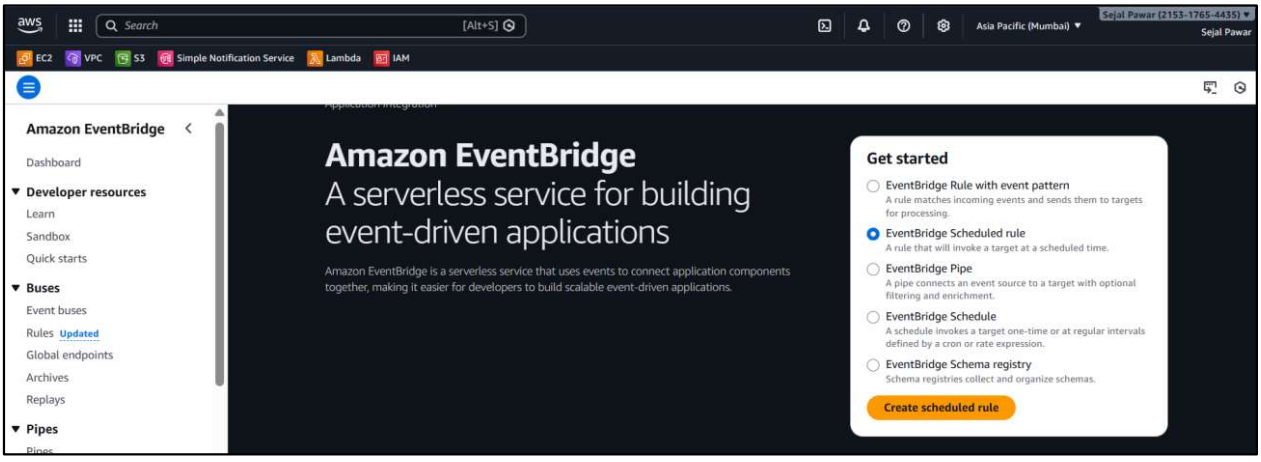
Step 11 : Verify file is upload in S3

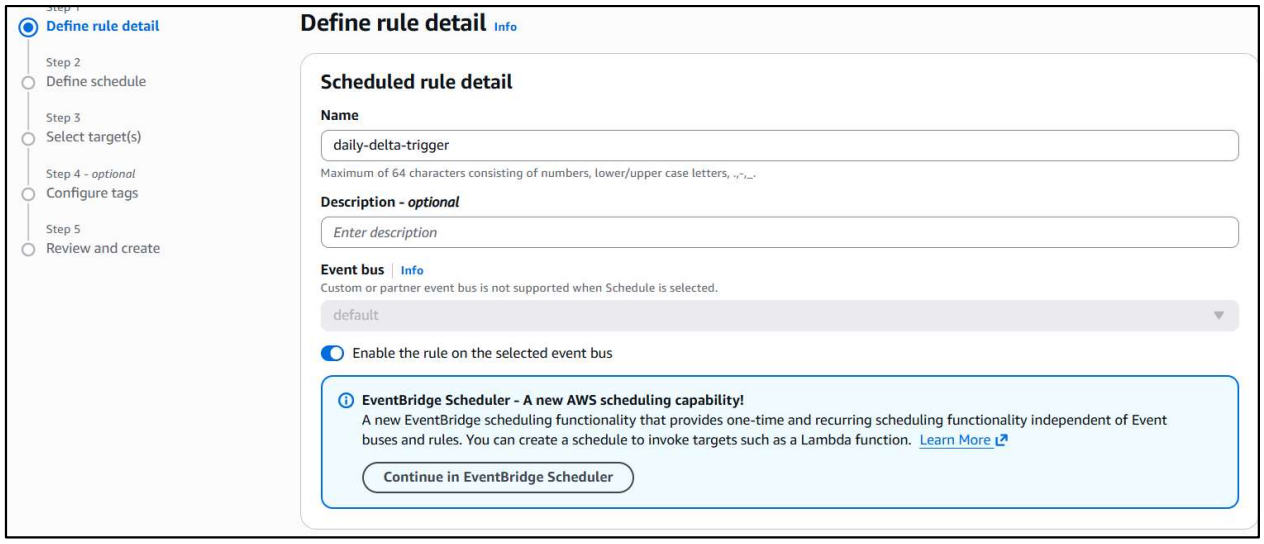
- Open S3 Bucket Created above.
- Click on bucket name.
- Files are generated.

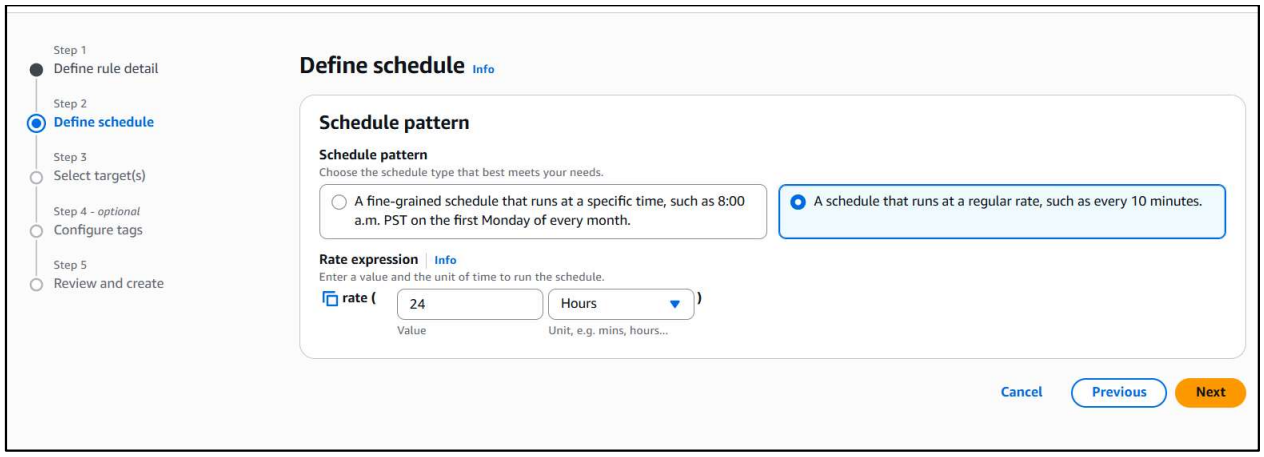


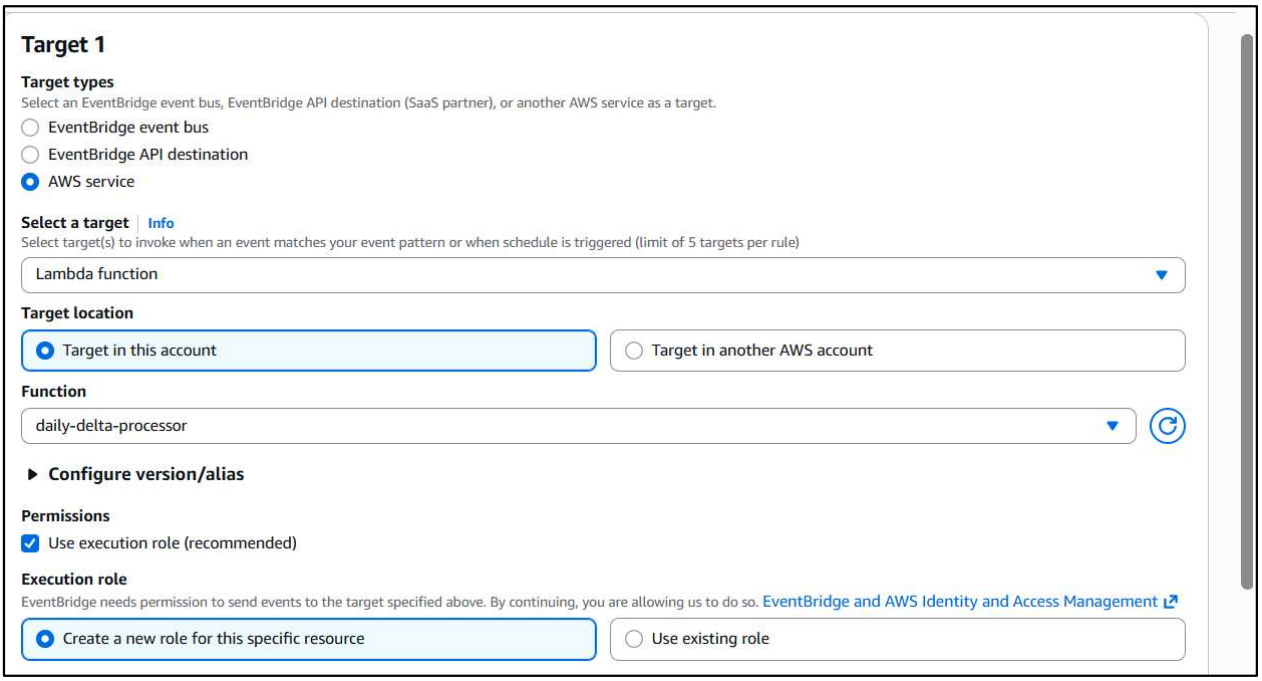
Step 12 : Create Event bridge Rule.

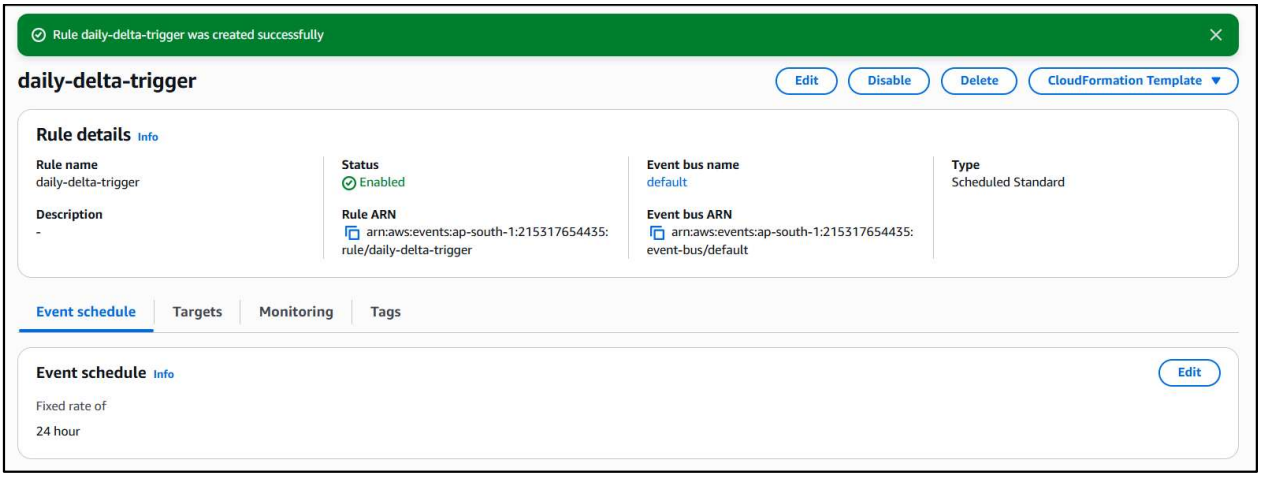
- AWS console > EventBridge
- Select EventBridge scheduled rule
- Click on create schedule rule.
- Name : daily-delta-trigger
- Defne Schedule : Schedule pattern
    - Set Schedule for 24 hours
- Target : AWS services
- Select targete : lambda Function
- Target this account
- Select Function : daily-delta-processor (Created above)



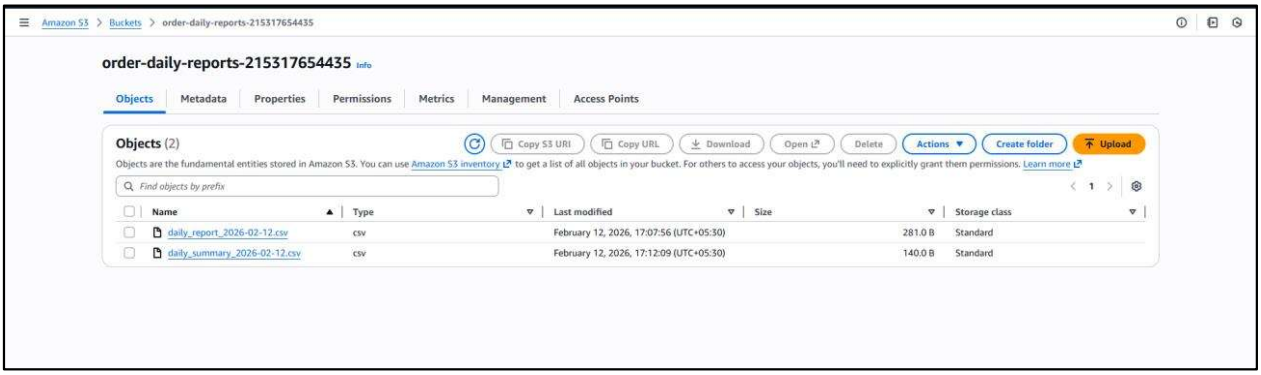

Define rule detail
Step 2
Define schedule
Step 3
Select target(s)
Step 4 - optional
Configure tags
Step 5
Review and create
Define rule detail Info
Scheduled rule detail
Name
daily-delta-trigger
Maximum of 64 characters consisting of numbers, lower/upper case letters, .,-,_.
Description - optional
Enter description
Event bus Info
Custom or partner event bus is not supported when Schedule is selected.
default
Enable the rule on the selected event bus
EventBridge Scheduler - A new AWS scheduling capability!
A new EventBridge scheduling functionality that provides one-time and recurring scheduling functionality independent of Event buses and rules. You can create a schedule to invoke targets such as a Lambda function. Learn More
Continue in EventBridge Scheduler


Step 1
Define rule detail
Step 2
Define schedule
Step 3
Select target(s)
Step 4 - optional
Configure tags
Step 5
Review and create
Define schedule Info
Schedule pattern
Schedule pattern
Choose the schedule type that best meets your needs.
A fine-grained schedule that runs at a specific time, such as 8:00 a.m. PST on the first Monday of every month.
A schedule that runs at a regular rate, such as every 10 minutes.
Rate expression Info
Enter a value and the unit of time to run the schedule.
rate (
24
Value
Hours
Unit, e.g. mins, hours...
)
Cancel
Previous
Next

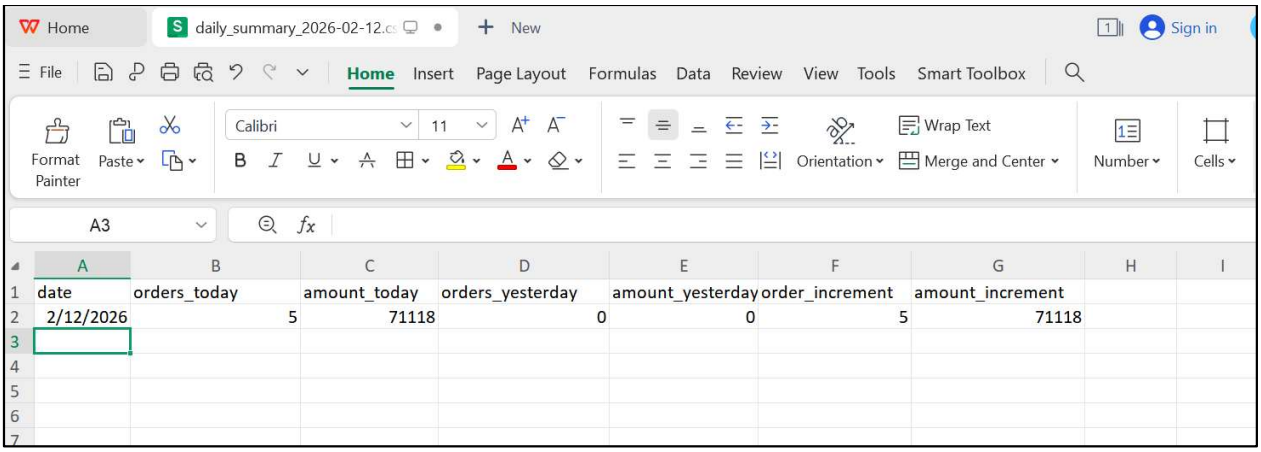

Target 1
Target types
Select an EventBridge event bus, EventBridge API destination (SaaS partner), or another AWS service as a target.
EventBridge event bus
EventBridge API destination
AWS service
Select a target | Info
Select target(s) to invoke when an event matches your event pattern or when schedule is triggered (limit of 5 targets per rule)
Lambda function
Target location
Target in this account
Target in another AWS account
Function
daily-delta-processor
Configure version/alias
Permissions
Use execution role (recommended)
Execution role
EventBridge needs permission to send events to the target specified above. By continuing, you are allowing us to do so. EventBridge and AWS Identity and Access Management
Create a new role for this specific resource
Use existing role


Rule daily-delta-trigger was created successfully
daily-delta-trigger
Edit
Disable
Delete
CloudFormation Template
Rule details Info
Rule name
daily-delta-trigger
Description
-
Status
Enabled
Rule ARN
arn:aws:events:ap-south-1:215317654435:rule/daily-delta-trigger
Event bus name
default
Event bus ARN
arn:aws:events:ap-south-1:215317654435:event-bus/default
Type
Scheduled Standard
Event schedule
Targets
Monitoring
Tags
Event schedule Info
Edit
Fixed rate of
24 hour

- After 24 hrs new report is generated.



- Report generated in S3.



Troubleshooting

During implementation of the Lambda + RDS integration, I faced several issues. Below are the problems and how I resolved them.

1. Lambda Error: No module named 'pymysql'

Issue

- My Lambda function failed with:
- Runtime.ImportModuleError: No module named 'pymysql'

Reason

- The required Python dependency was not included in the Lambda deployment package.

Resolution

- I installed pymysql locally and packaged it with the function:
    - pip install pymysql -t .
    - zip -r function.zip .

Then I uploaded the zip file to Lambda.
Alternatively, I created a Lambda Layer and attached it to the function.

2. Secret Key Error: KeyError 'dbname'

Issue

- Lambda Show:
    - KeyError: 'dbname'

Reason

- The database name key was missing in AWS Secrets Manager.

Resolution

- I updated the secret JSON to include:
  - "dbname": "orders"

3. Lambda Timeout

Issue

- Task timed out after 3.00 seconds

Reason

- Lambda could not connect to RDS in time.

Resolution

- I increased the timeout:
  - Lambda → Configuration → General configuration
  - Timeout: 15 seconds

4. Lambda Cannot Connect to RDS

Issue

- Function kept timing out.

Checks I performed

- Lambda and RDS are in the same VPC
- Security group allows port 3306
- RDS inbound rule allows Lambda security group
- Correct rule:
  - Type: MySQL
  - Port: 3306
  - Source: Lambda security group

After fixing security groups, the connection worked.

5. IAM Permission Error

Issue

- The execution role does not have permission to call CreateNetworkInterface

Reason

- Lambda role lacked VPC permissions.

Resolution

- I attached this policy to the Lambda role:
    - AWSLambdaVPCAccessExecutionRole