Guided Lab: Connecting AWS Lambda to Amazon RDS

Description

In modern cloud applications, serverless computing with AWS Lambda is often combined with managed databases like Amazon RDS to build scalable, cost-efficient solutions. Connecting Lambda functions to RDS allows you to execute database queries, manage data, and build data-driven applications without managing servers. This lab demonstrates how to securely connect a Lambda function to an RDS instance using environment variables to manage database connection settings.

Why do this?
Connecting Lambda to RDS allows you to build dynamic, data-driven applications where Lambda handles the backend logic, and the data is stored and managed in RDS. This setup is ideal for use cases like API backends, serverless data processing, and real-time applications where you need scalable and secure access to a relational database.

Prerequisites

This lab assumes you have a basic understanding of AWS Lambda, Amazon RDS Database, basic networking in AWS, and Python programming.

If you find any gaps in your knowledge, consider taking the following lab:

- Creating an AWS Lambda function
- Using Environment Variables in AWS Lambda
- Creating an Amazon RDS database
- Security Group VS Network Access Control List

Objectives

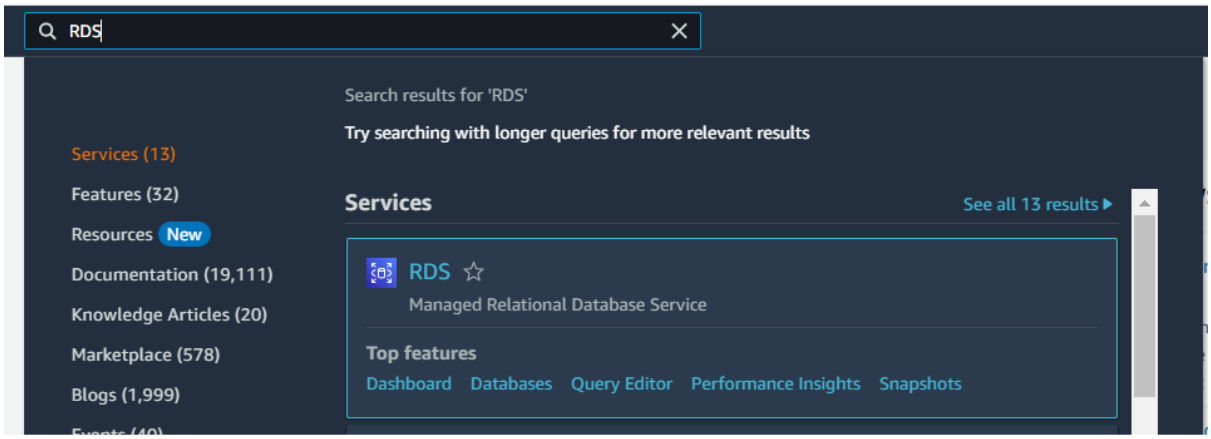
In this lab, you will:

- Create an RDS instance
- Configure a Lambda function to connect to the RDS instance.
- Securely store database credentials using environment variables.
- Write and execute a simple SQL query from the Lambda function.

Lab Steps

Create an Amazon RDS Instance

1. Navigate to the RDS Console.



2. **Create a Database with the following configuration:**

- **Choose a database creation method:** Standard create
- **Engine options:**
  - **Engine type:** MySQL
  - **Engine Version:** Leave it as default

Choose a database creation method Info

- (●) **Standard create**
  You set all of the configuration options, including ones for availability, security, backups, and maintenance.
- ( ) **Easy create**
  Use recommended best-practice configurations. Some configuration options can be changed after the database is created.

Engine options

Engine type Info

- ( ) Aurora (MySQL Compatible)
- ( ) Aurora (PostgreSQL Compatible)
- (●) MySQL
- ( ) MariaDB
- ( ) PostgreSQL
- ( ) Oracle
- ( ) Microsoft SQL Server
- ( ) IBM Db2

Edition

(●) MySQL Community

Engine version Info

View the engine versions that support the following database features.

▼ Hide filters

[ ] Show versions that support the Multi-AZ DB cluster Info
Create a A Multi-AZ DB cluster with one primary DB instance and two readable standby DB instances. Multi-AZ DB clusters provide up to 2x faster transaction commit latency and automatic failover in typically under 35 seconds.

[ ] Show versions that support the Amazon RDS Optimized Writes Info
Amazon RDS Optimized Writes improves write throughput by up to 2x at no additional cost.

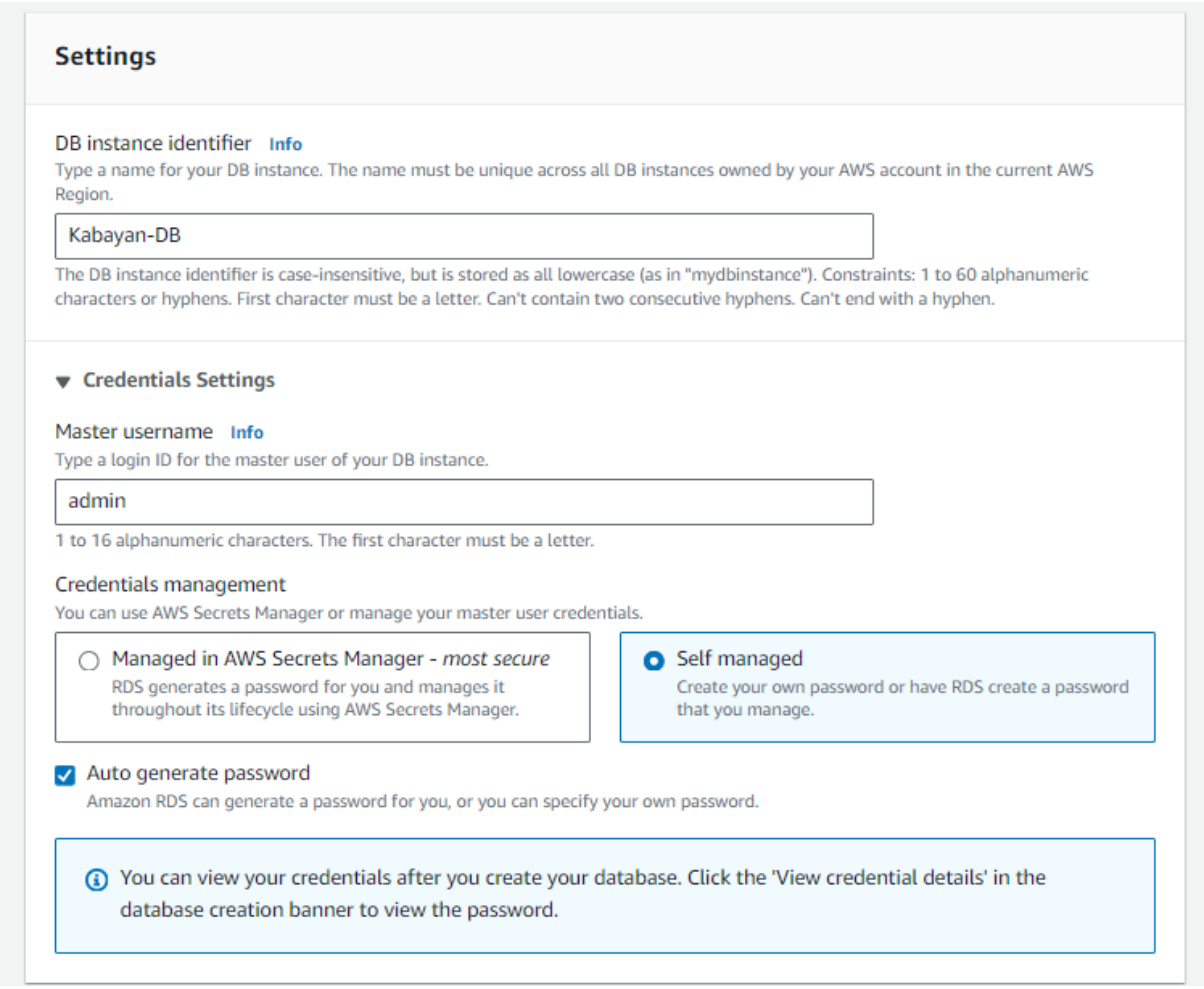
Engine Version

MySQL 8.0.35 ▼

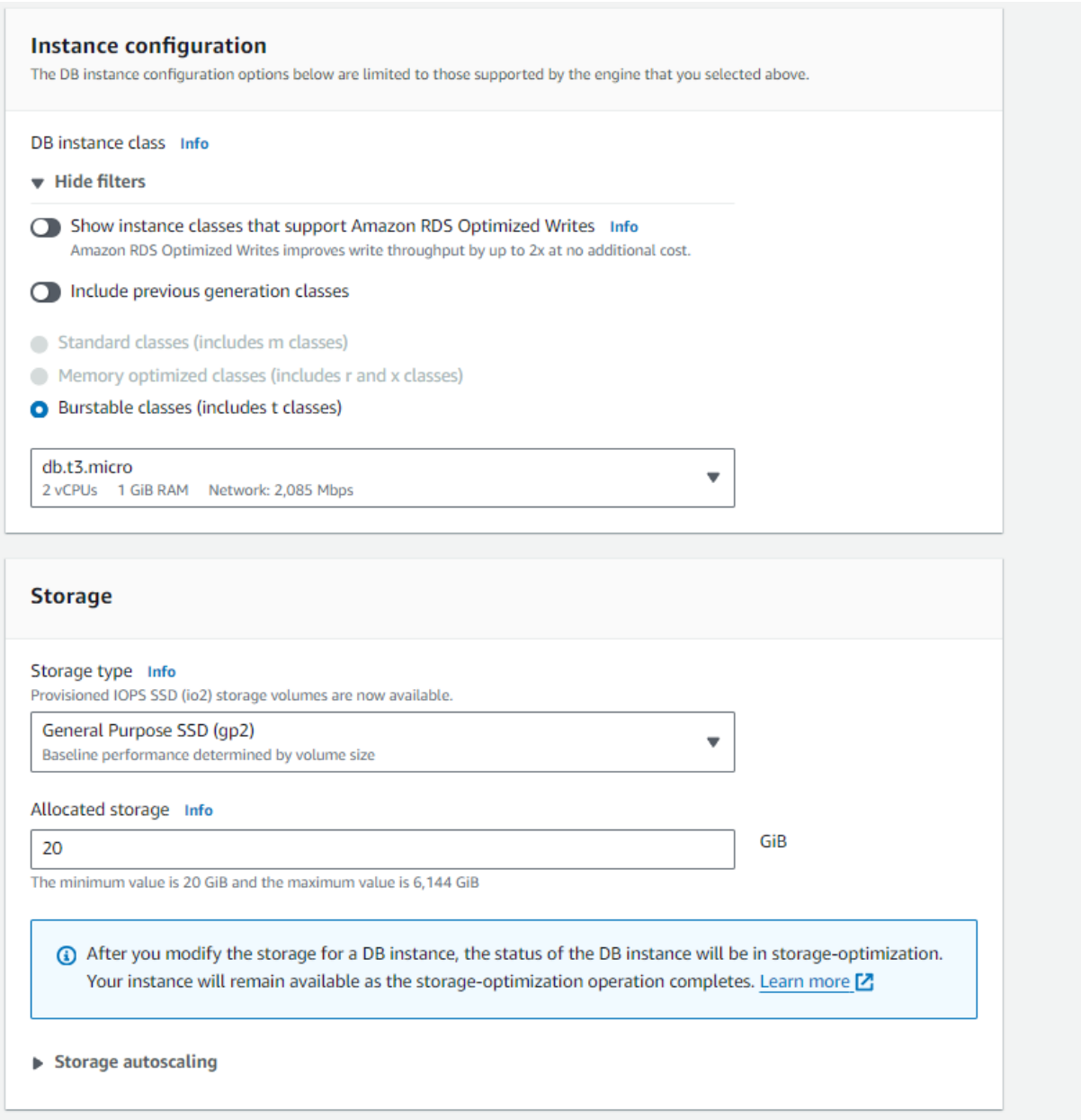
[ ] Enable RDS Extended Support Info
Amazon RDS Extended Support is a paid offering. By selecting this option, you consent to being charged for this offering if you are running your database major version past the RDS end of standard support date for that version. Check the end of standard support date for your major version in the RDS for MySQL documentation.

-
    - **Templates:** Free tier
    - **Settings:**
        - **DB instance identifier:** Kabayan-DB
        - **Master username:** admin (you can change this as you desire)
        - **Credentials management:** Self managed
        - For the simplicity of this lab we will tick the checkbox of **Auto generate password**.

But take note that in PRODUCTION, you need to create a strong password for the security of your Database



- **Instance configuration**:
    - **DB instance size**:, db.t3.micro
- **Storage:**
    - **Storage type:** General Purpose SSD (gp2)
    - **Allocated storage:** 20

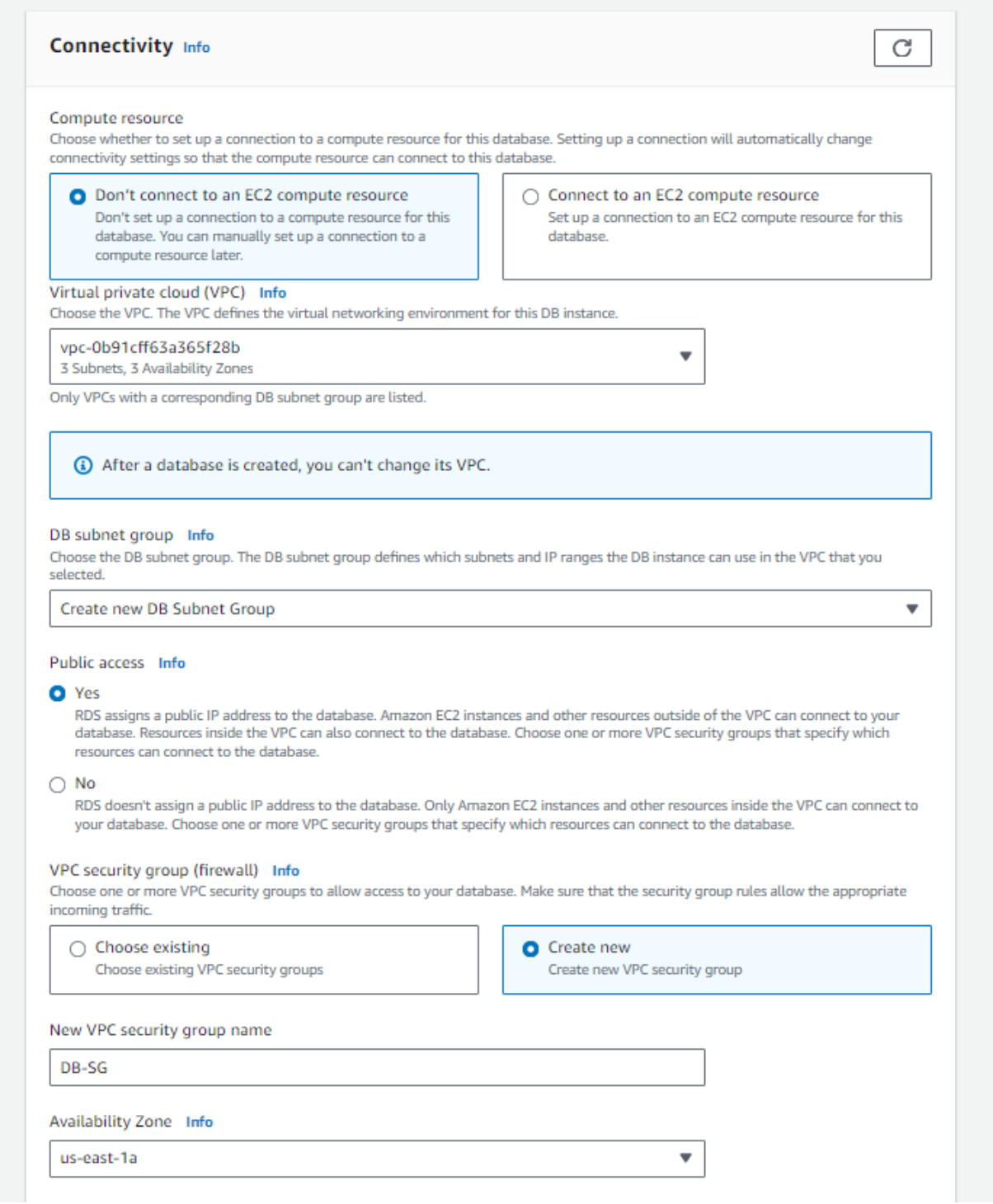


- **Connectivity:**
  - **Compute resource:** Don't connect to an EC2 compute resource
  - **Virtual private cloud (VPC):** leave it as the default
  - **DB subnet group:** Create new DB Subnet Group
  - **Public access:** Yes

Ensure the RDS instance is publicly accessible if you plan to connect from Lambda within the same VPC

-
  - **VPC security group (firewall):** Select Create new
    - *New VPC security group name:* DB-SG

- *Availability Zone:* Select your preferred one (e.g., us-east-1a)



- Leave the rest as default and click **Create Database**.

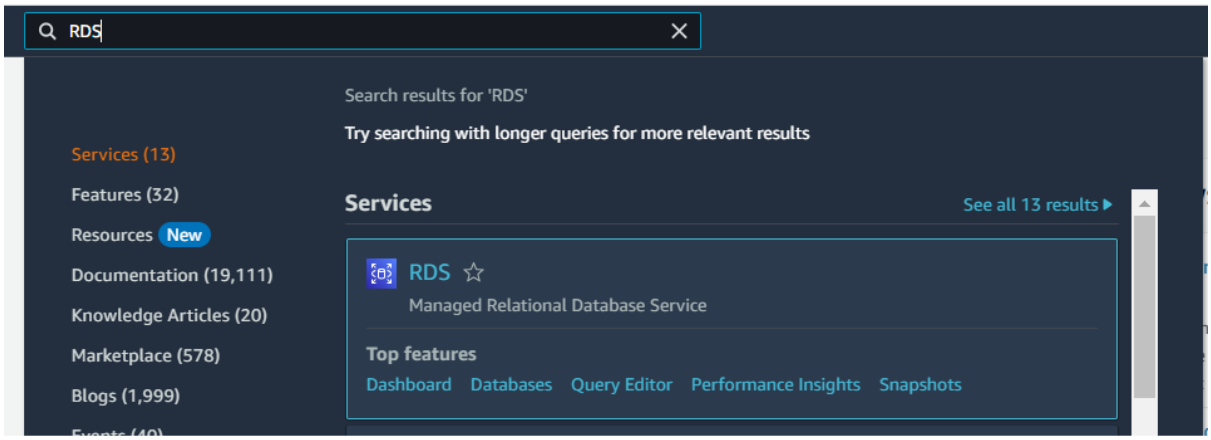
This will take a few minutes to finish.

3. **Note down the database endpoint, username, and password** for later use.

Lab Steps

Create an Amazon RDS Instance

1. Navigate to the RDS Console.



2. **Create a Database with the following configuration:**

- **Choose a database creation method:** Standard create
- **Engine options:**
  - **Engine type:** MySQL
  - **Engine Version:** Leave it as default

Choose a database creation method Info

- (●) **Standard create**
  You set all of the configuration options, including ones for availability, security, backups, and maintenance.
- ( ) **Easy create**
  Use recommended best-practice configurations. Some configuration options can be changed after the database is created.

Engine options

Engine type Info

- ( ) Aurora (MySQL Compatible)
- ( ) Aurora (PostgreSQL Compatible)
- (●) MySQL
- ( ) MariaDB
- ( ) PostgreSQL
- ( ) Oracle
- ( ) Microsoft SQL Server
- ( ) IBM Db2

Edition

(●) MySQL Community

Engine version Info

View the engine versions that support the following database features.

▼ Hide filters

[ ] Show versions that support the Multi-AZ DB cluster Info
Create a A Multi-AZ DB cluster with one primary DB instance and two readable standby DB instances. Multi-AZ DB clusters provide up to 2x faster transaction commit latency and automatic failover in typically under 35 seconds.

[ ] Show versions that support the Amazon RDS Optimized Writes Info
Amazon RDS Optimized Writes improves write throughput by up to 2x at no additional cost.

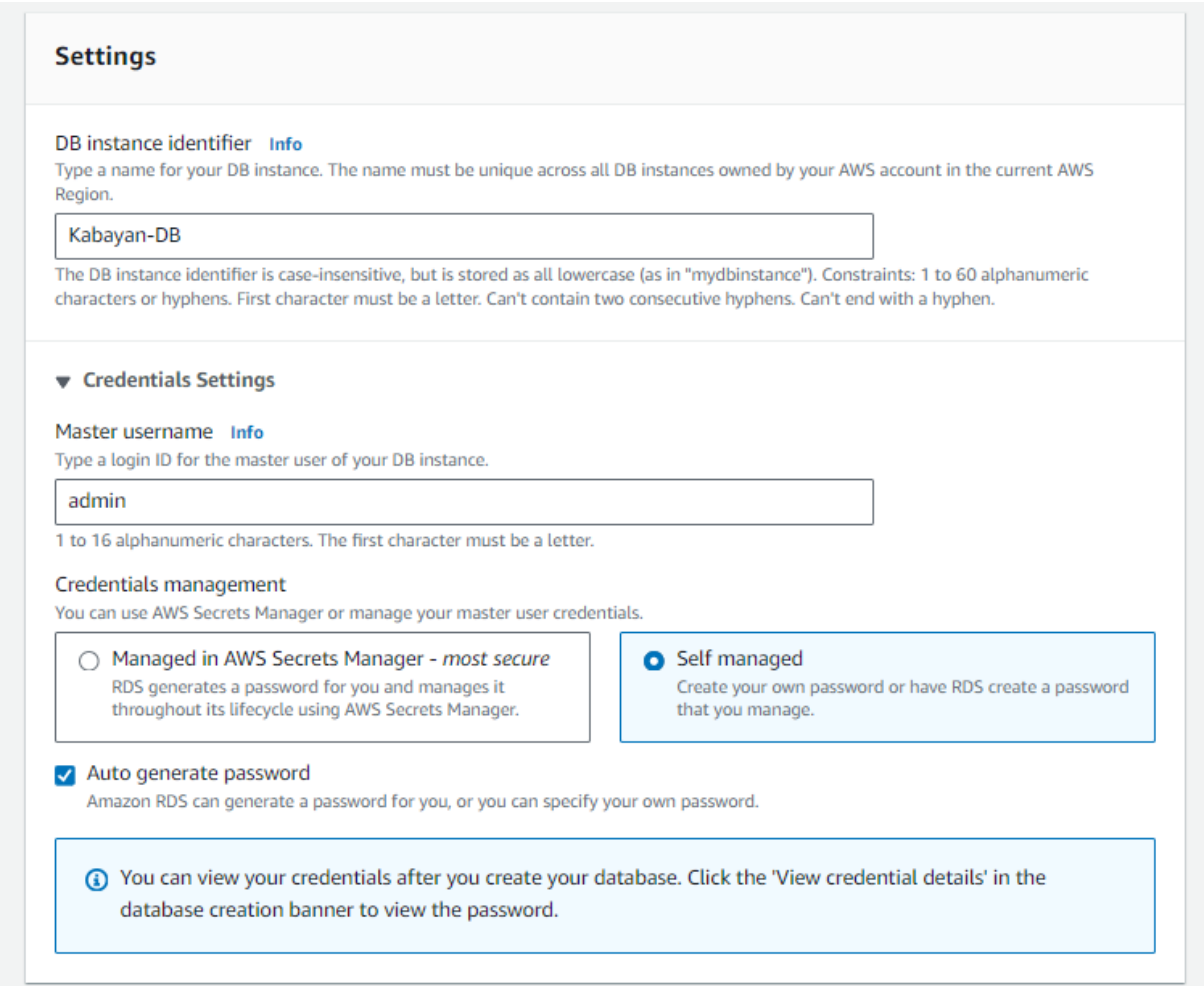
Engine Version

MySQL 8.0.35 ▼

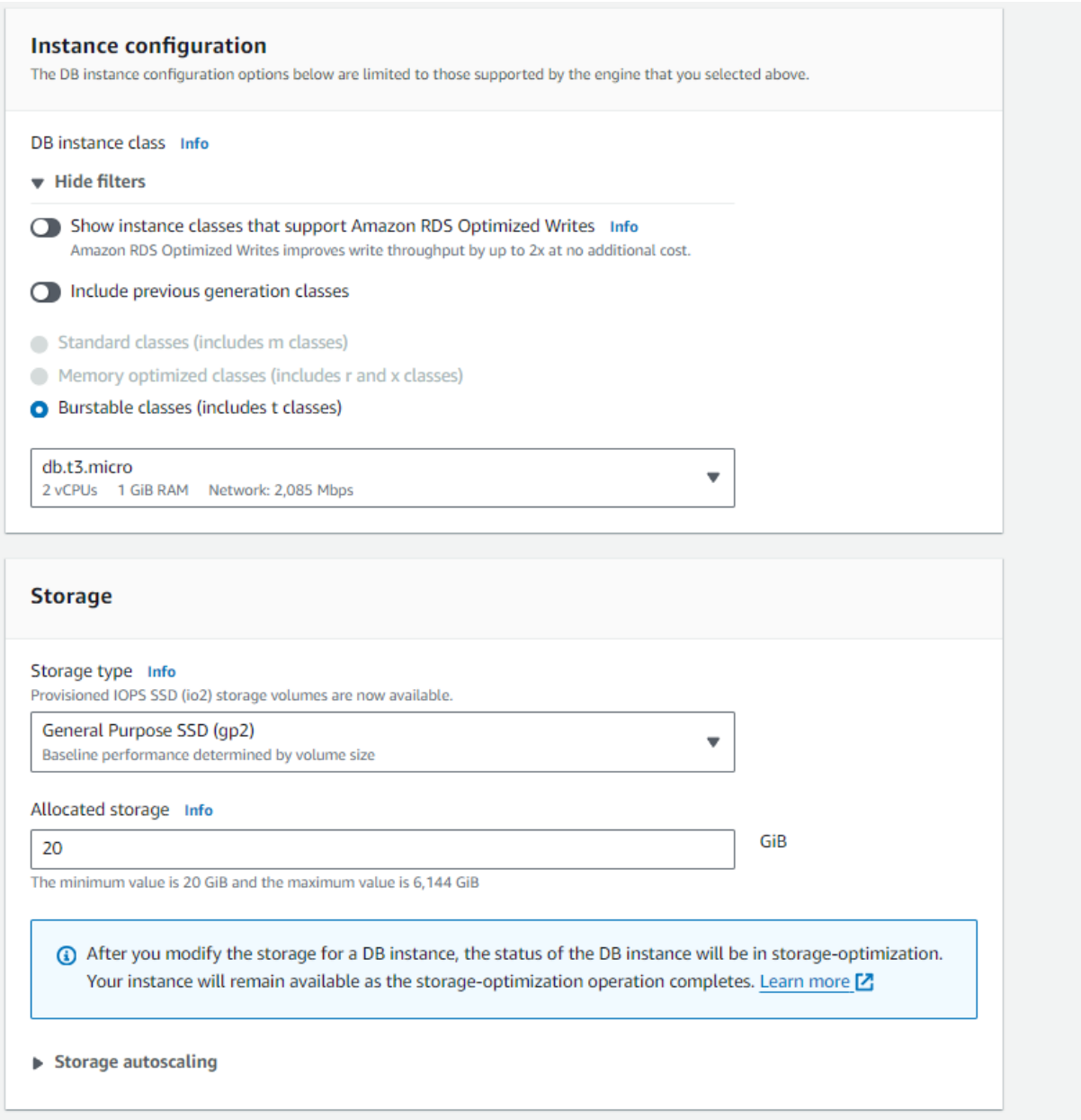
[ ] Enable RDS Extended Support Info
Amazon RDS Extended Support is a paid offering. By selecting this option, you consent to being charged for this offering if you are running your database major version past the RDS end of standard support date for that version. Check the end of standard support date for your major version in the RDS for MySQL documentation.

-
    - **Templates:** Free tier
    - **Settings:**
        - **DB instance identifier:** Kabayan-DB
        - **Master username:** admin (you can change this as you desire)
        - **Credentials management:** Self managed
        - For the simplicity of this lab we will tick the checkbox of **Auto generate password**.

But take note that in PRODUCTION, you need to create a strong password for the security of your Database



- **Instance configuration**:
    - **DB instance size**:, db.t3.micro
- **Storage:**
    - **Storage type:** General Purpose SSD (gp2)
    - **Allocated storage:** 20

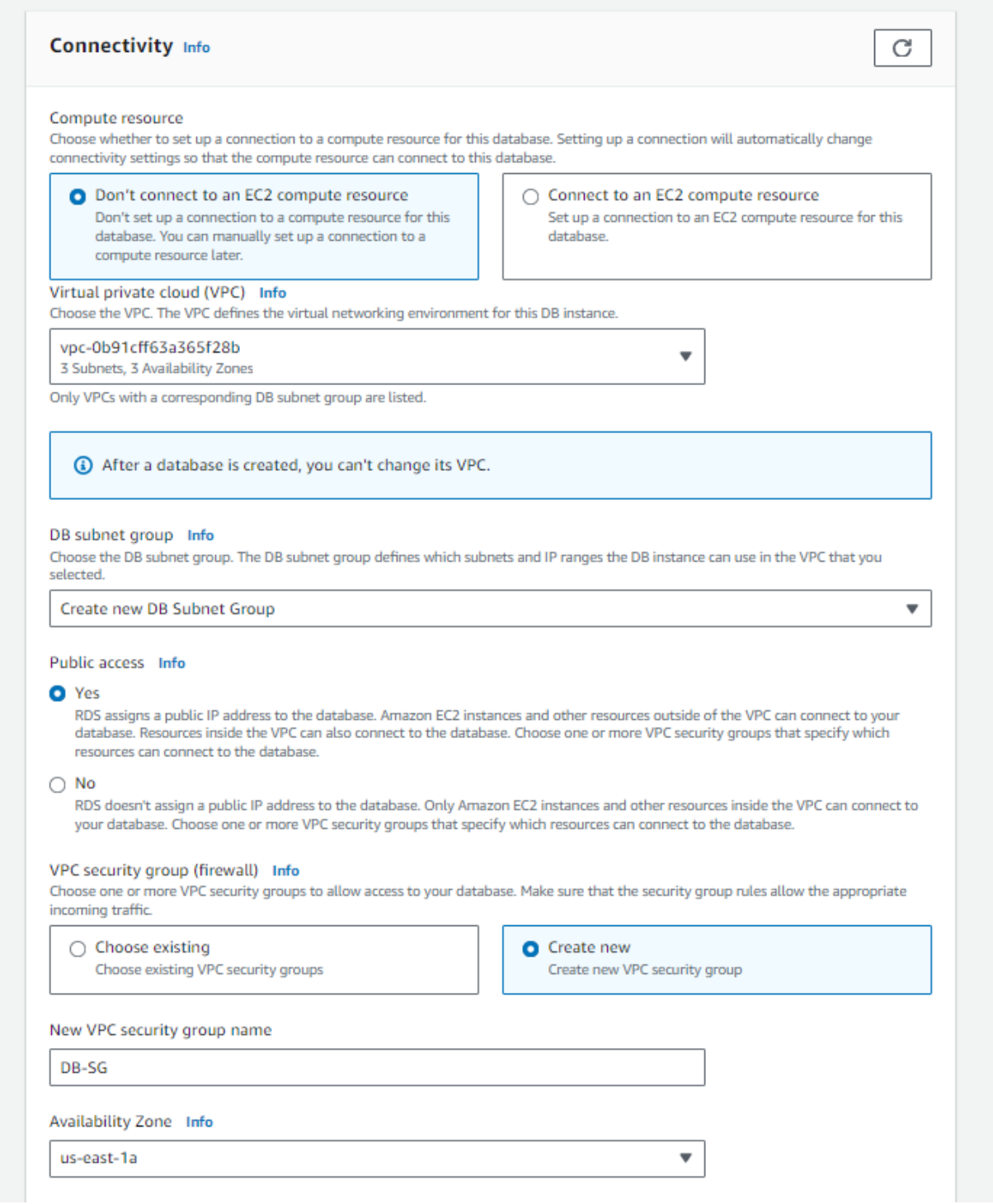


- **Connectivity:**
  - **Compute resource:** Don't connect to an EC2 compute resource
  - **Virtual private cloud (VPC):** leave it as the default
  - **DB subnet group:** Create new DB Subnet Group
  - **Public access:** Yes

Ensure the RDS instance is publicly accessible if you plan to connect from Lambda within the same VPC

-
  - **VPC security group (firewall):** Select Create new
    - *New VPC security group name:* DB-SG

- *Availability Zone:* Select your preferred one (e.g., us-east-1a)



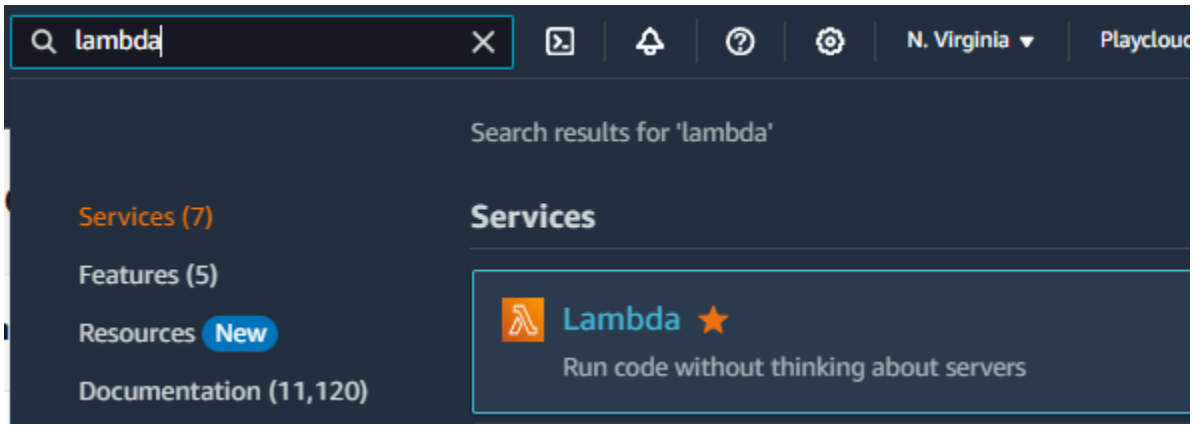
- Leave the rest as default and click **Create Database**.

This will take a few minutes to finish.

3. **Note down the database endpoint, username, and password** for later use.

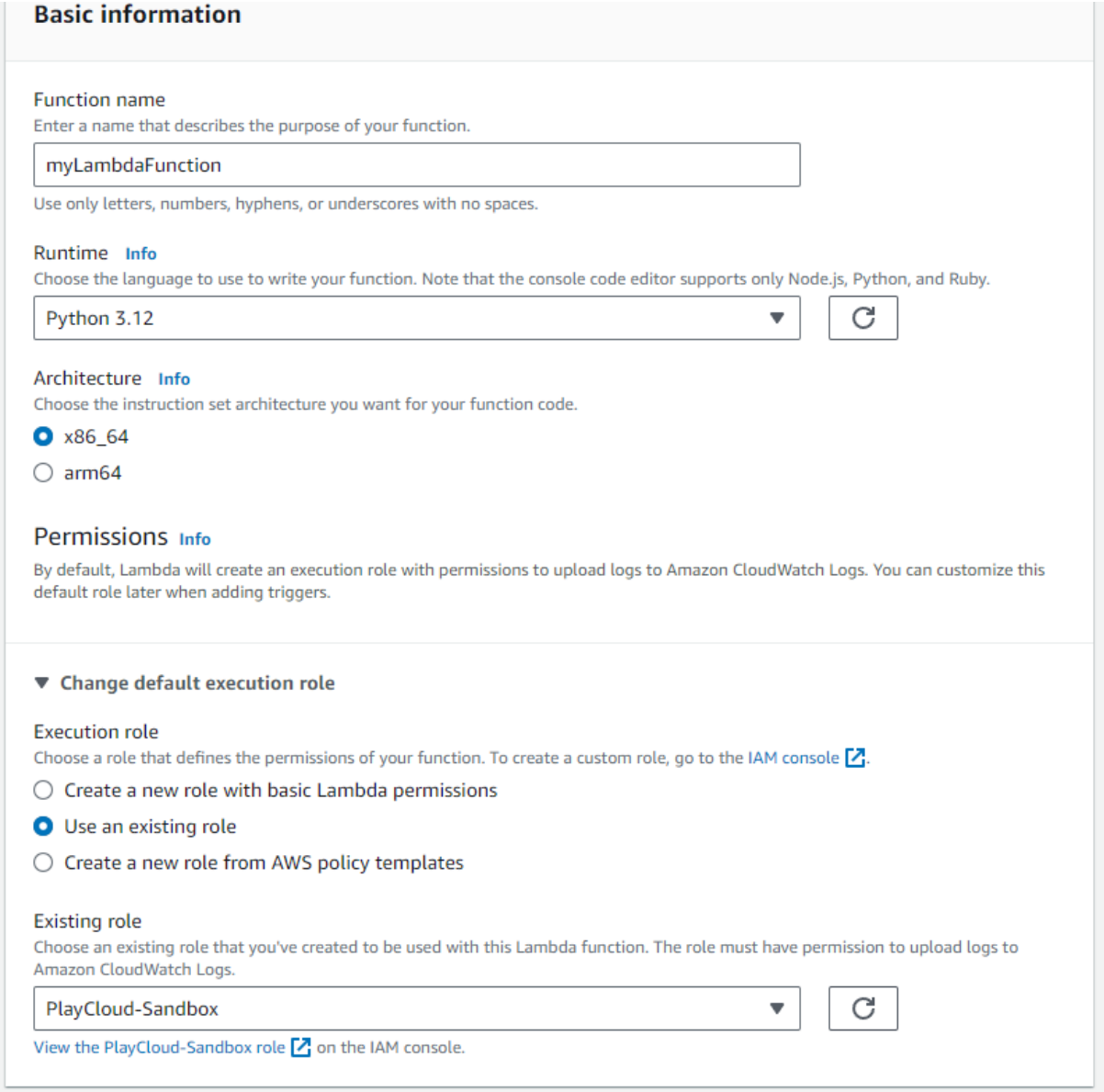
Create the Lambda Function

1. Navigate to the AWS Lambda Console



2. Create a new Lambda function using the following configurations:

- Choose **Author from scratch.**
- Function name: myLambdaFunction
- Select Python 3.12 as the runtime.
- **Execution role:**
    - Select Use an Existing Role: PlayCloud-Sanbox



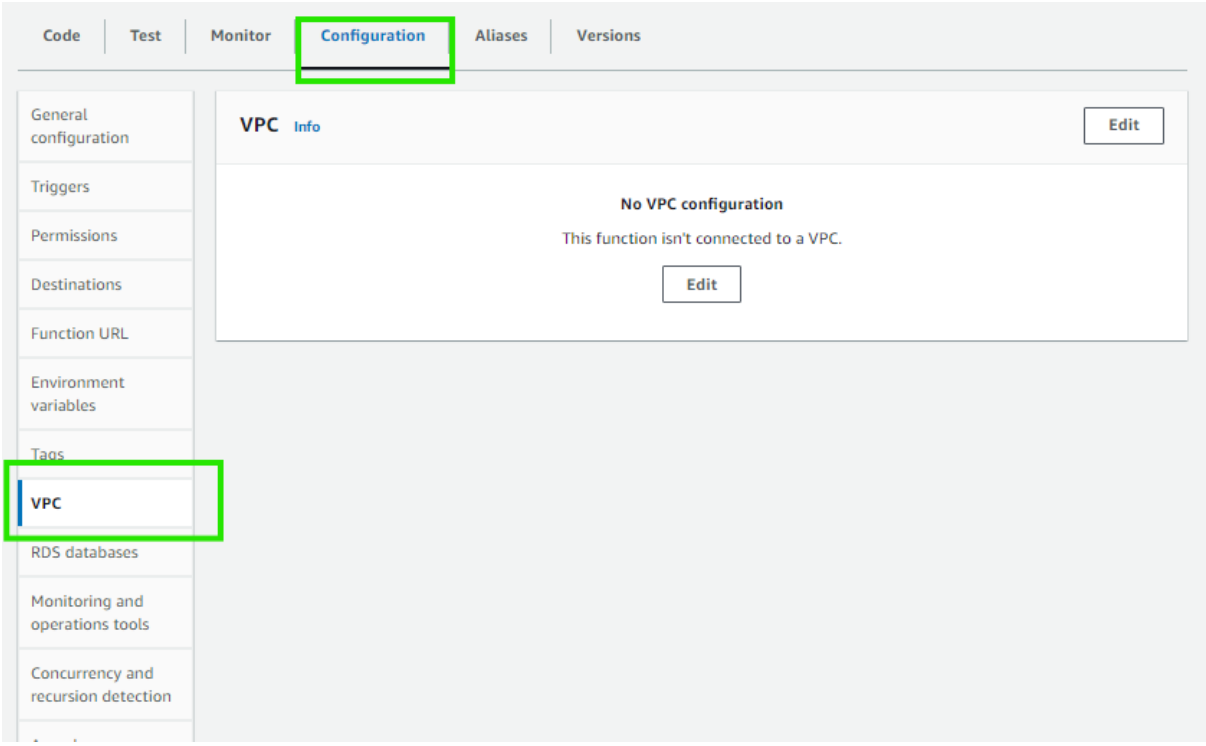
- Click **Create function**

Configure VPC and Security Groups

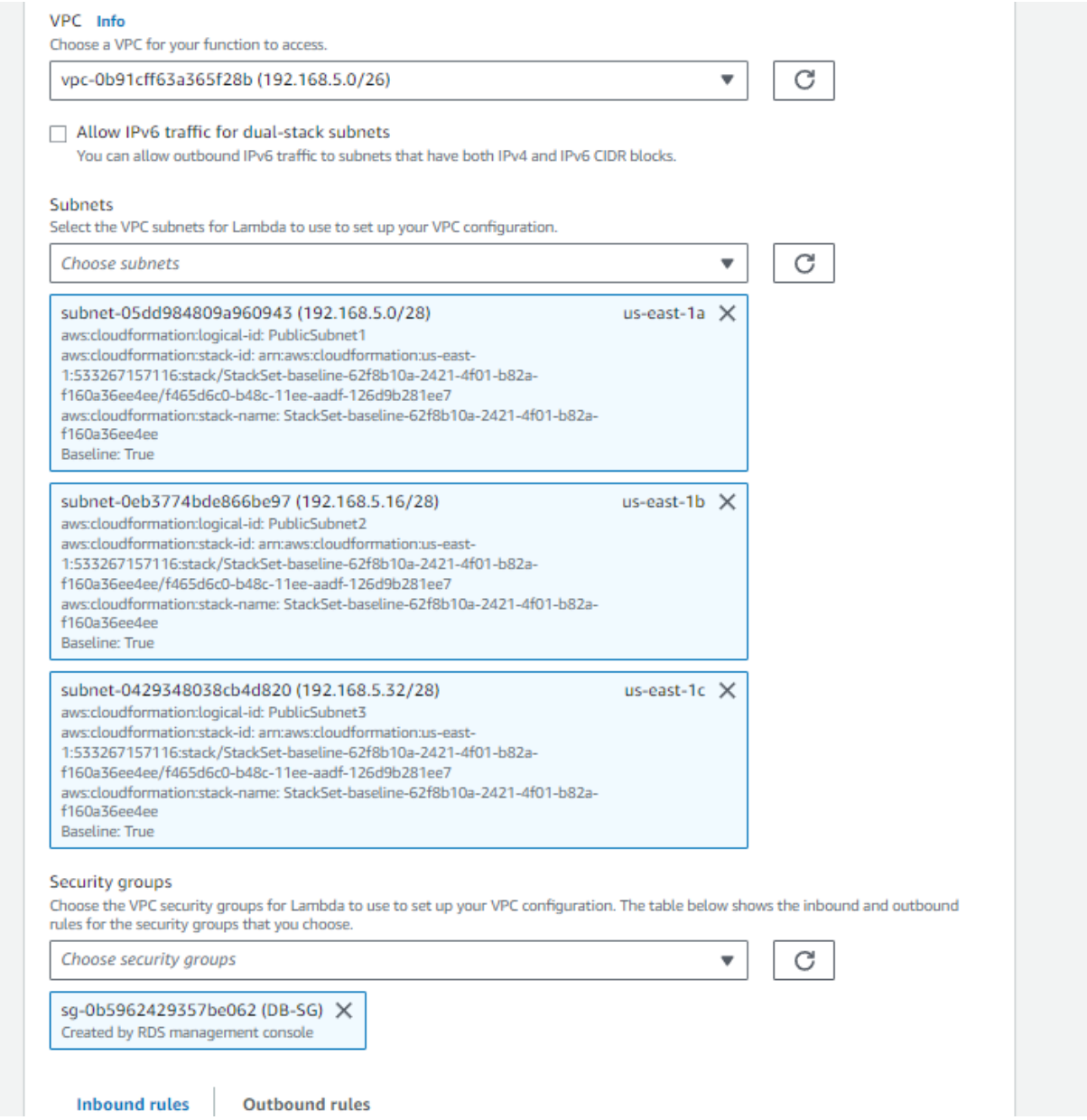
1. Ensure that your Lambda function and RDS instance are in the same VPC.

2. Modify the Lambda function's VPC settings to include the necessary subnets and security groups to communicate with the RDS instance.

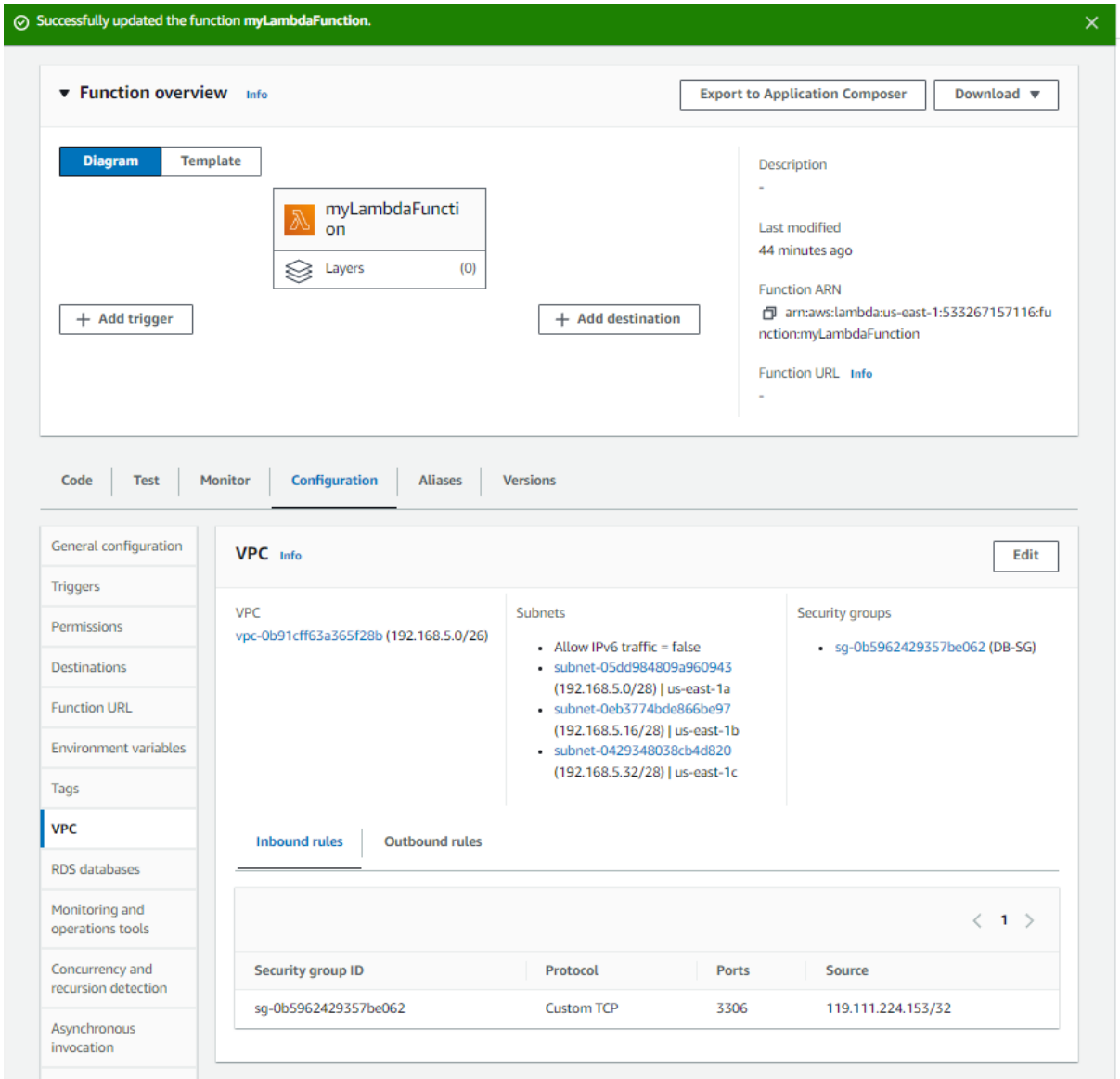
- Navigate to the **Configuration tab** and click on **VPC.**



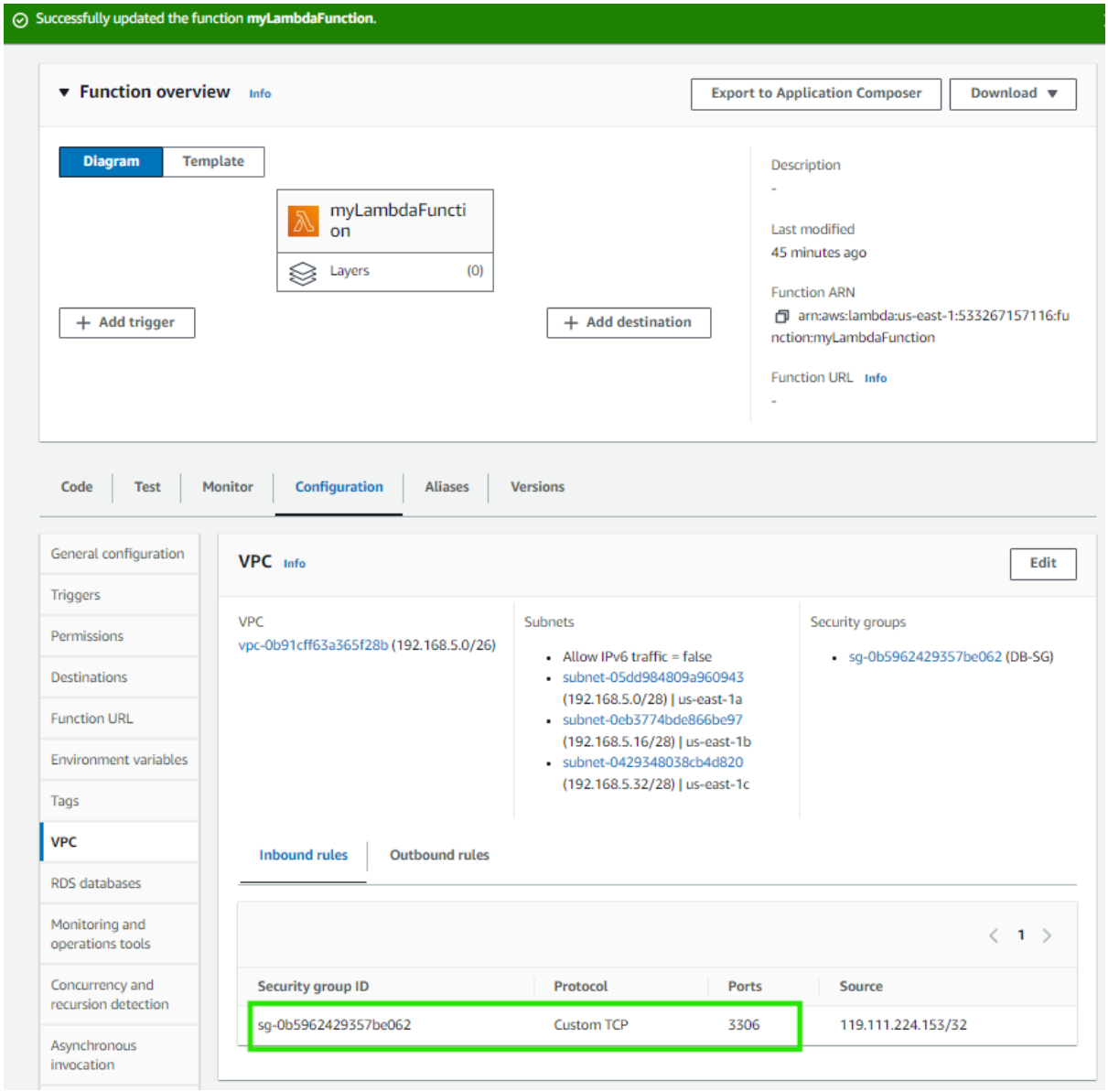
- Click on Edit.
- Add the following setting:
    - **VPC:** Choose the default
    - **Subnets:** Choose the same subnets where you set the RDS Instance.
    - **Security groups:** Choose the DB-SG, the same SG as the RDS Instance.



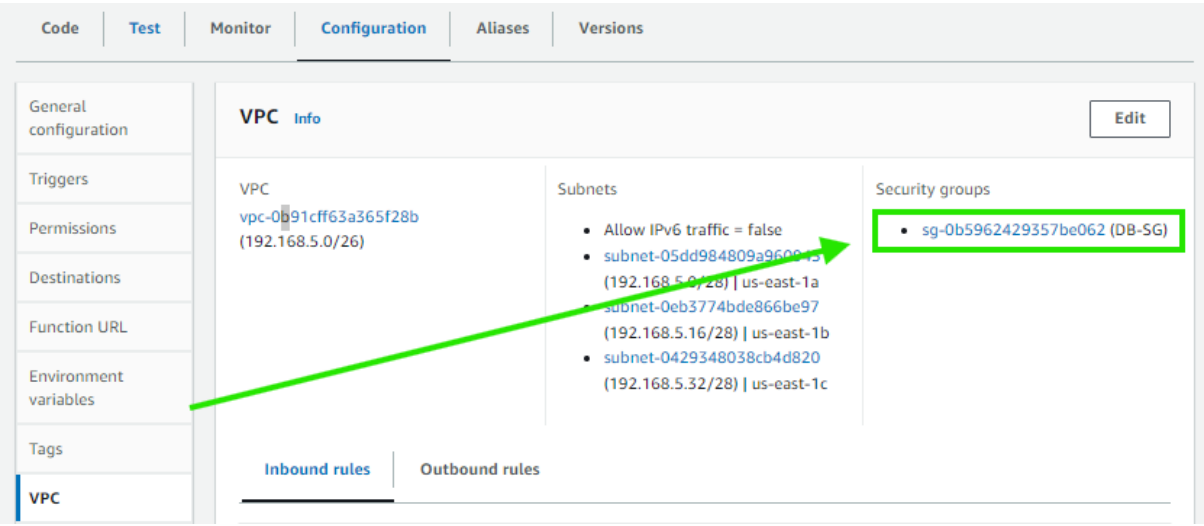
-
    - **Click Save** and wait for the function to update. Ensure the RDS security group allows inbound connections on port **3306** for MySQL.



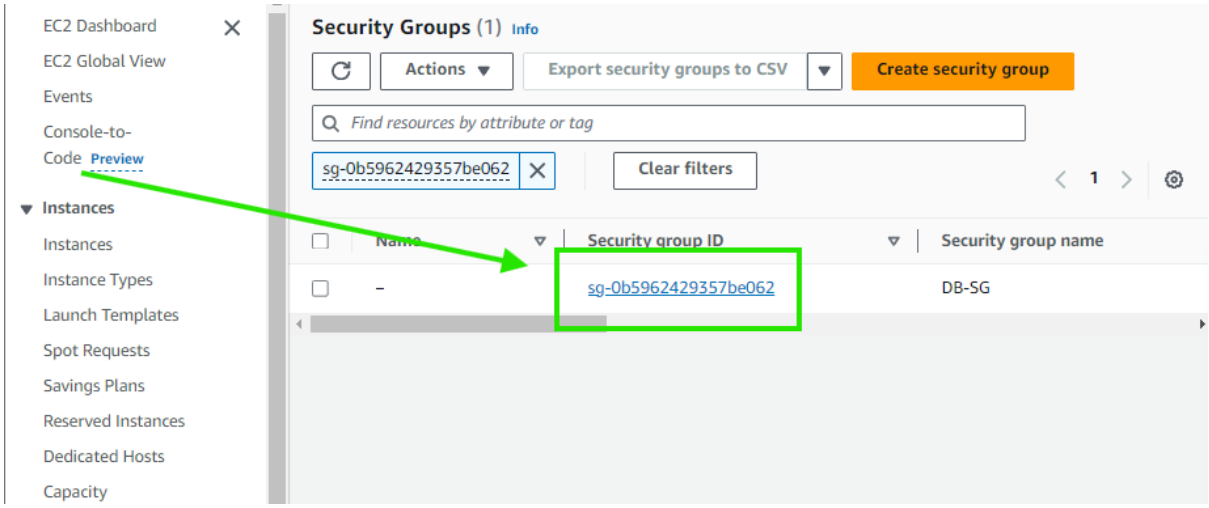
3. Ensure the RDS security group allows inbound connections on the database port (e.g., 3306 for MySQL).



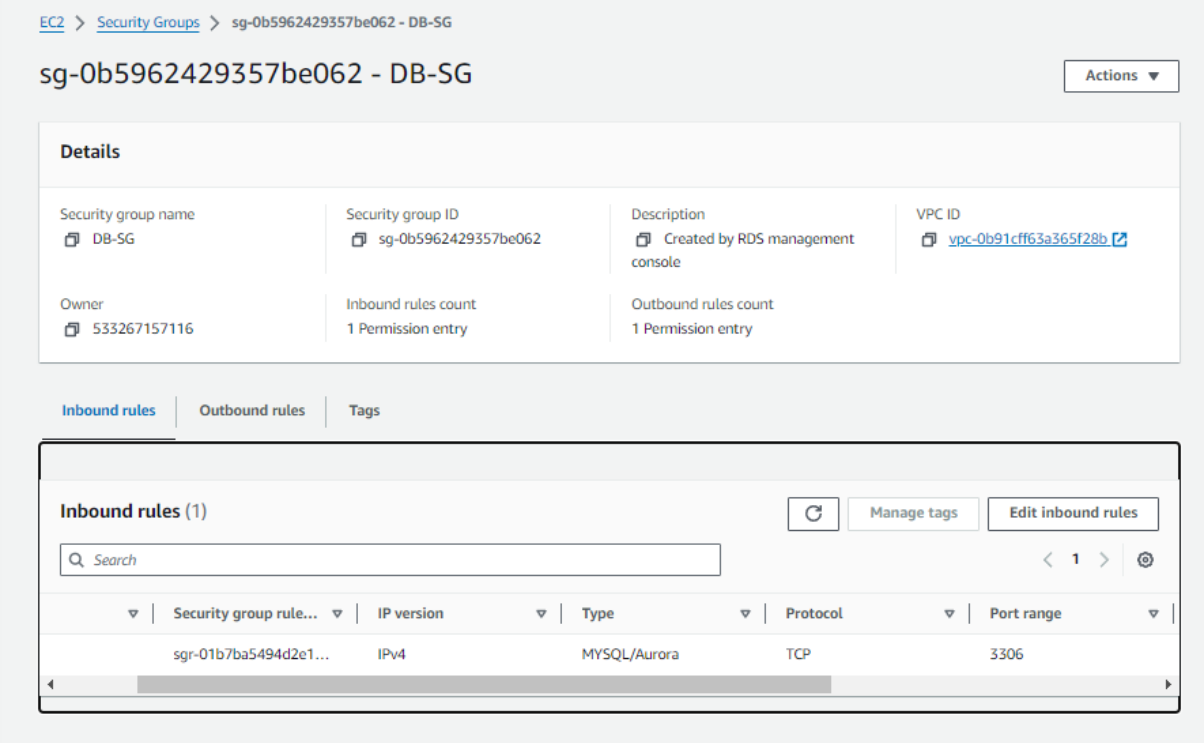
- You can also double-check this by clicking the security group.



- You will then be redirected to the security group tab. Click on the Security group ID.



- You should see the current Inbound rule, Port range 3306 for MySQL/Aurora, with your IP as the source. If you cannot see this, add the inbound rules for this setting. You can also set the source as **anywhere IPv4** for this lab.



Upload the Lambda Function Code

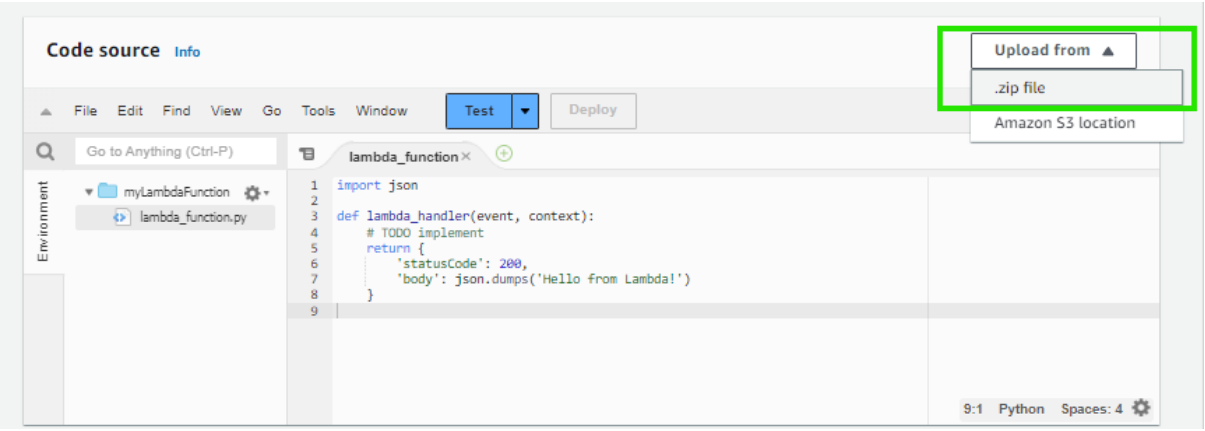
1. Navigate back to the Lambda Function Console to the **Code tab.**

Note*: We are using the* ***old console editor*** *for this lab. You're welcome to use either the old or new editor, whichever you prefer; the steps remain the same, though the interface may have a slightly different appearance in the new editor.*

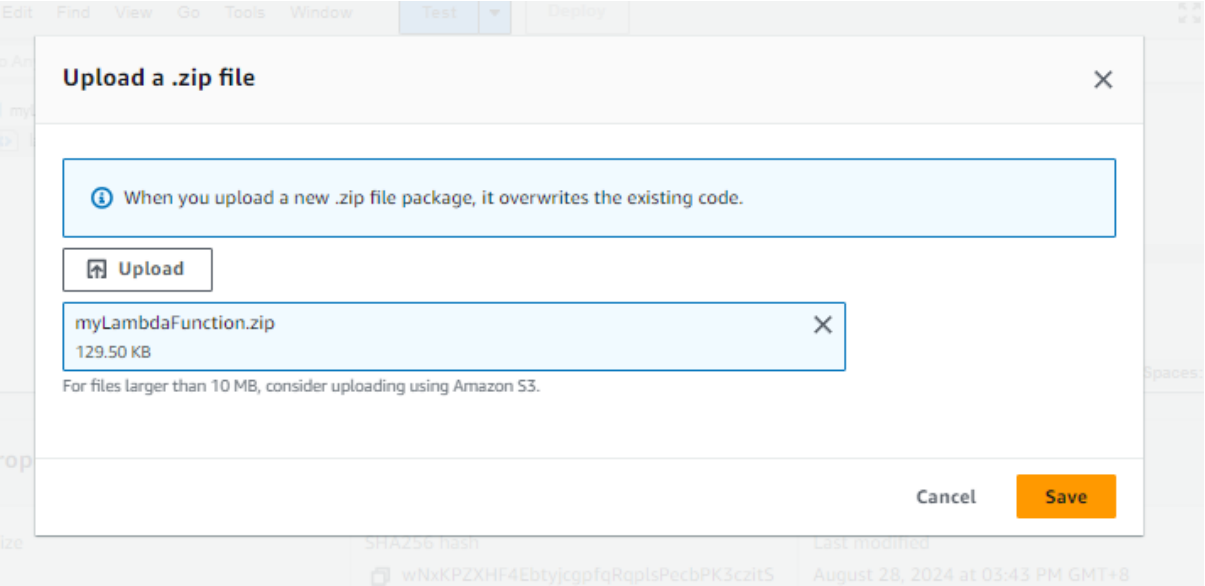
2. Download the following zip file:

https://media.tutorialsdojo.com/public/myLambdaFunction.zip

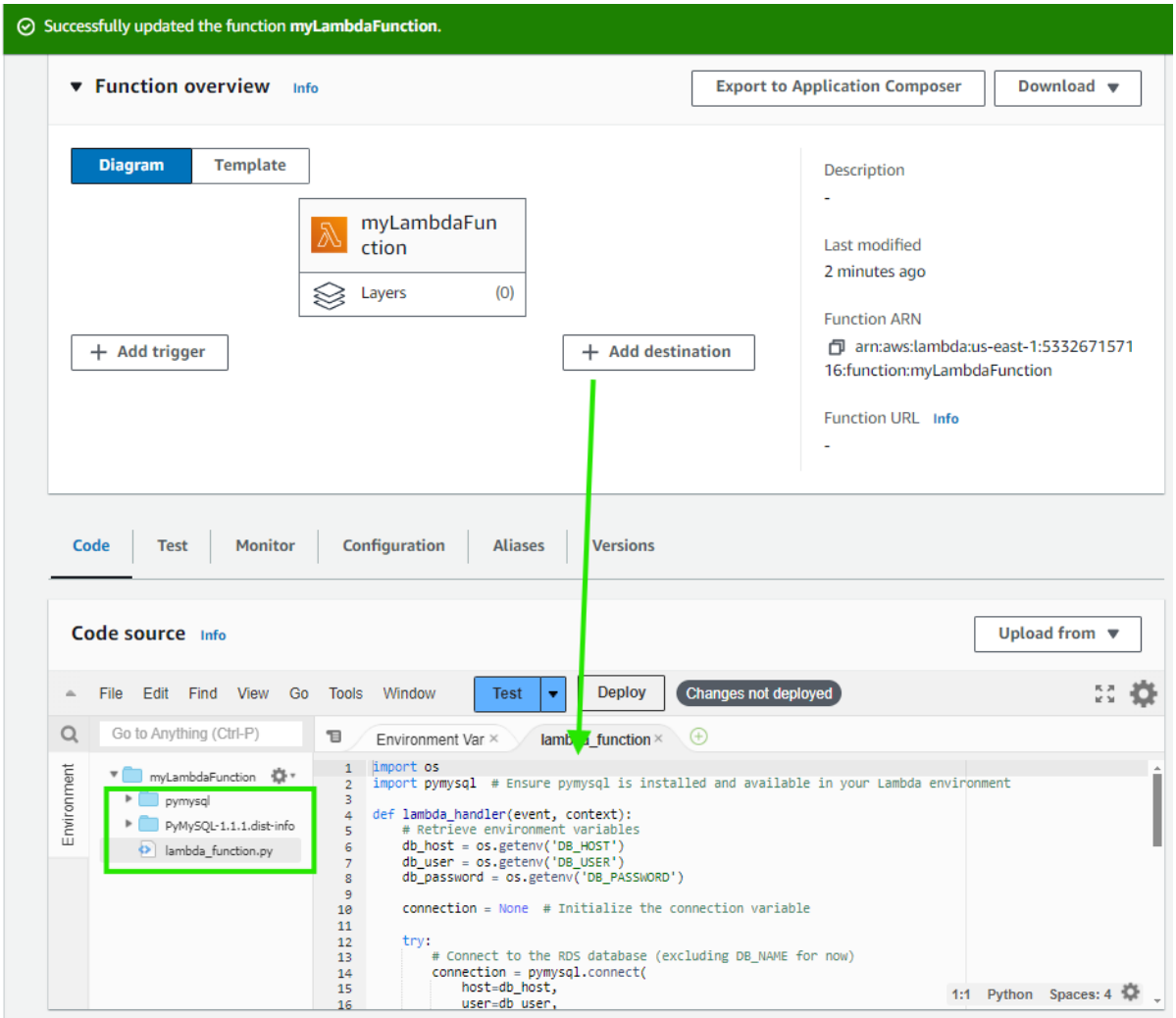
3. Click the upload from the button and select the .zip file



4. Click upload, select where you downloaded the zip file earlier from your local machine, and then click on save.



5. Your code should now be updated.



6. Take your time to review what is added to your code and lambda environment.

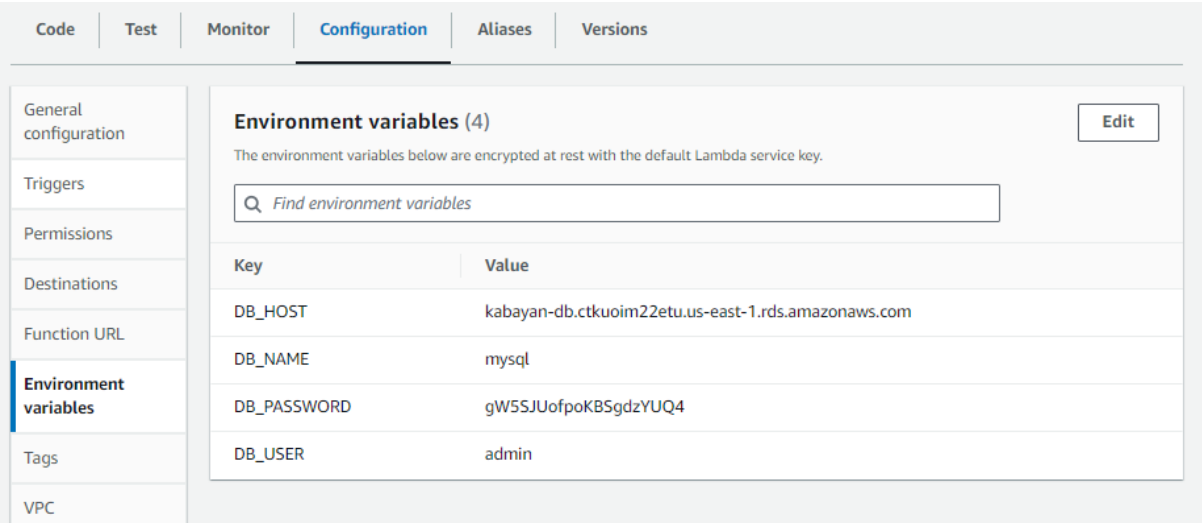
This AWS Lambda function, utilizing the pymysql module, connects to a MySQL database using credentials stored in environment variables. It performs a simple query, SHOW DATABASES;, to retrieve a list of all databases on the MySQL server and checks if a specified database exists. Based on the result, it returns an HTTP 200 status code if the database is found, a 404 status if not, or a 500 status in case of a connection error. The connection is safely closed after the operation.

---

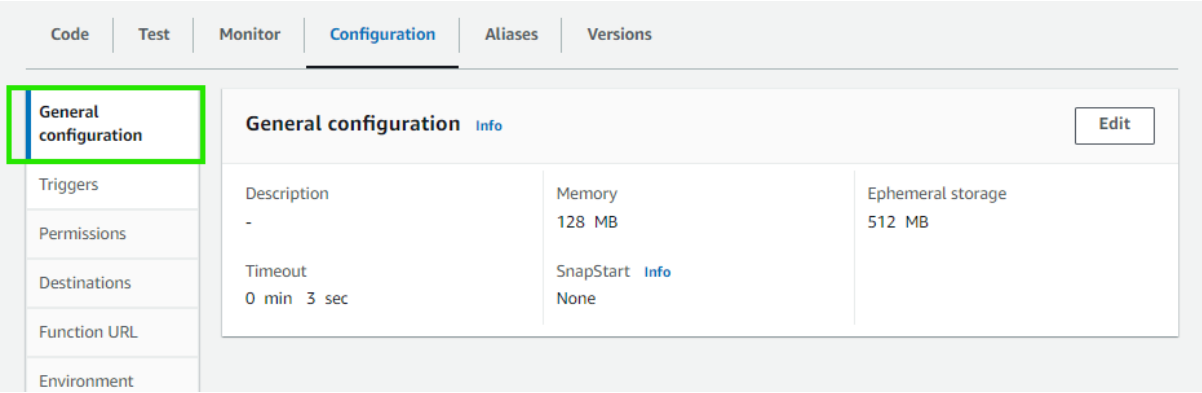
7. Add Environment Variables to the Lambda Function

- In the Lambda console, navigate to the “Configuration” tab and add the following environment variables:
    - DB_HOST: The endpoint of your RDS instance.
    - DB_USER: The database username.
    - DB_PASSWORD: The database password.
    - DB_NAME : **MySQL**

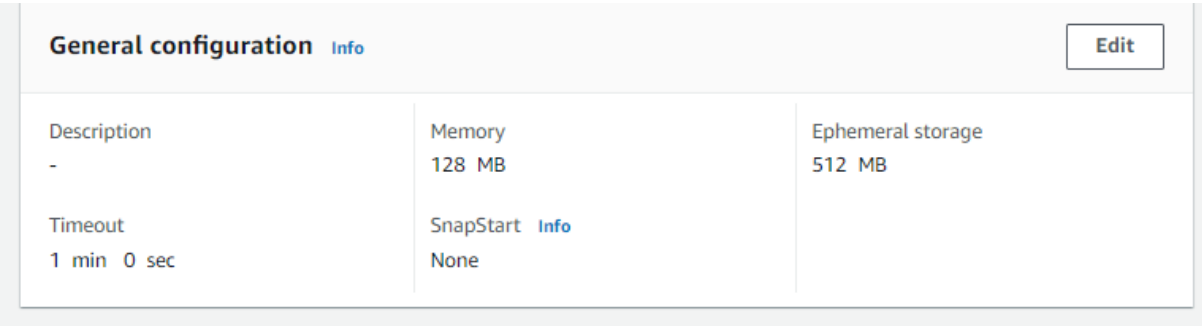
- Click **Save**



8. Lastly, click on the General configuration.

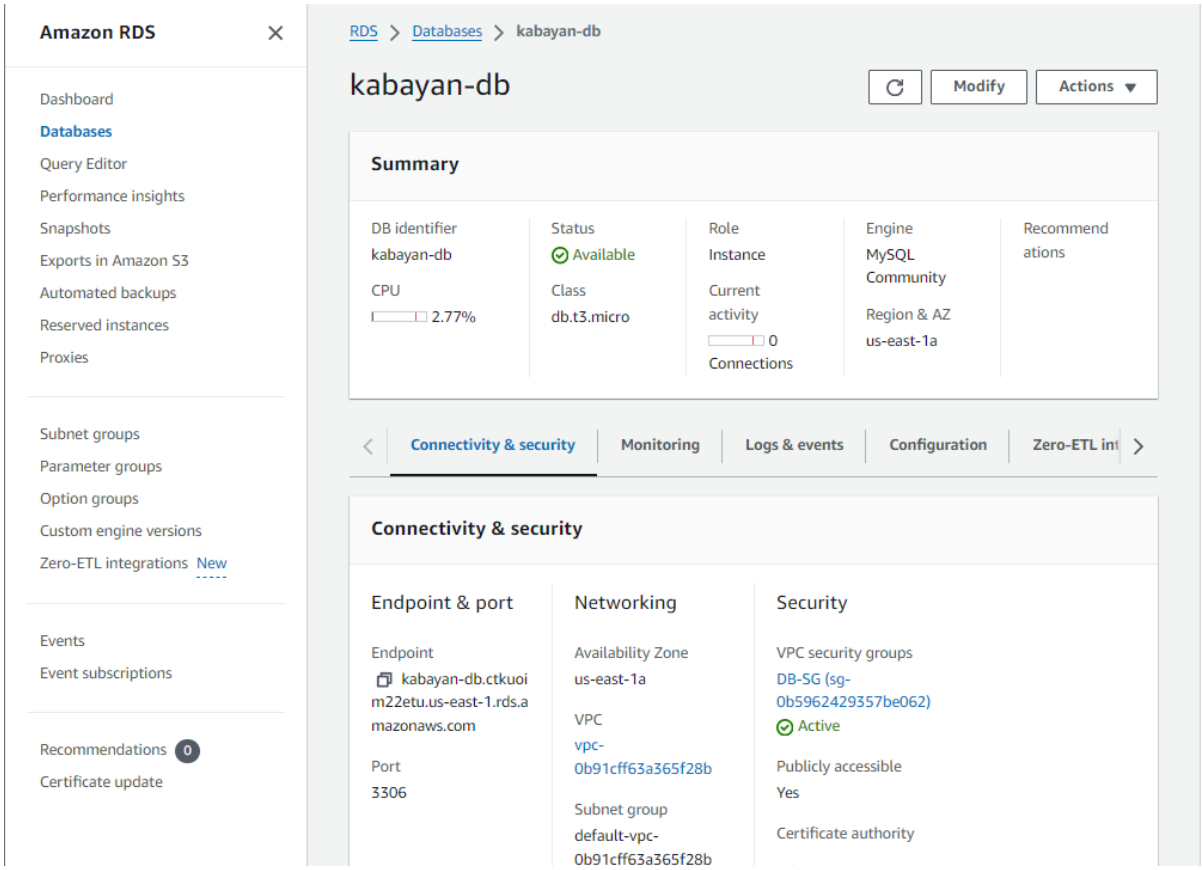


- change the **Timeout** to 1 min and save it

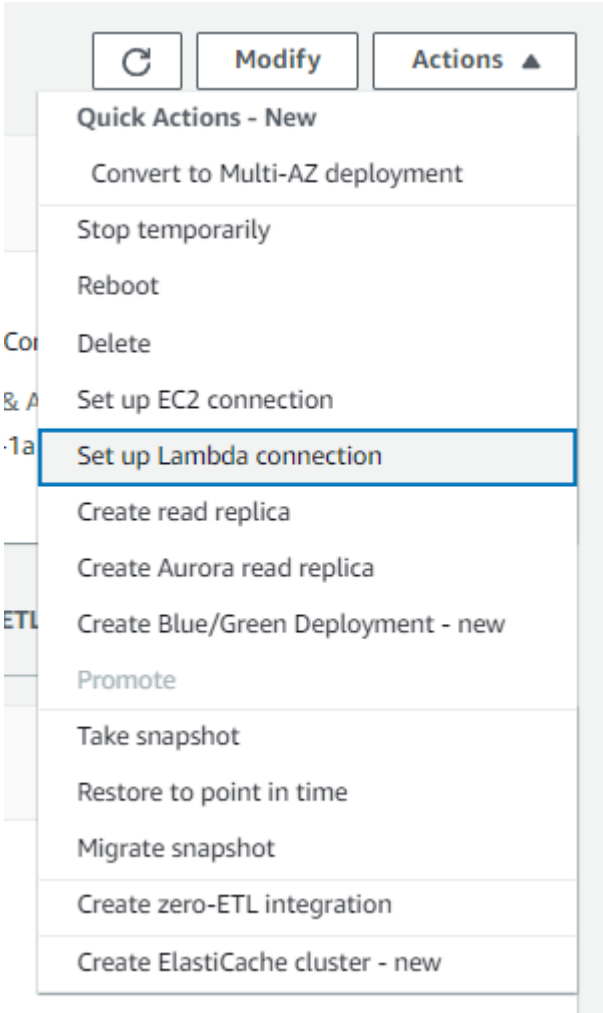


Set up Lambda connection

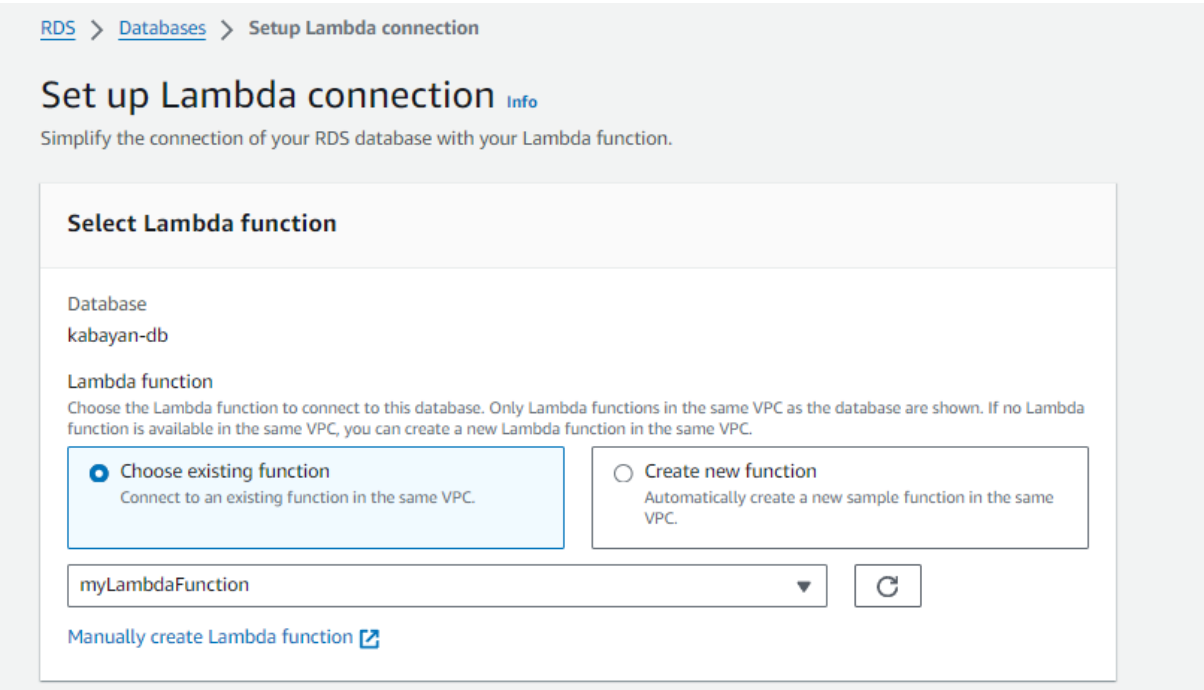
1. Navigate back to the RDS Instance you created earlier.



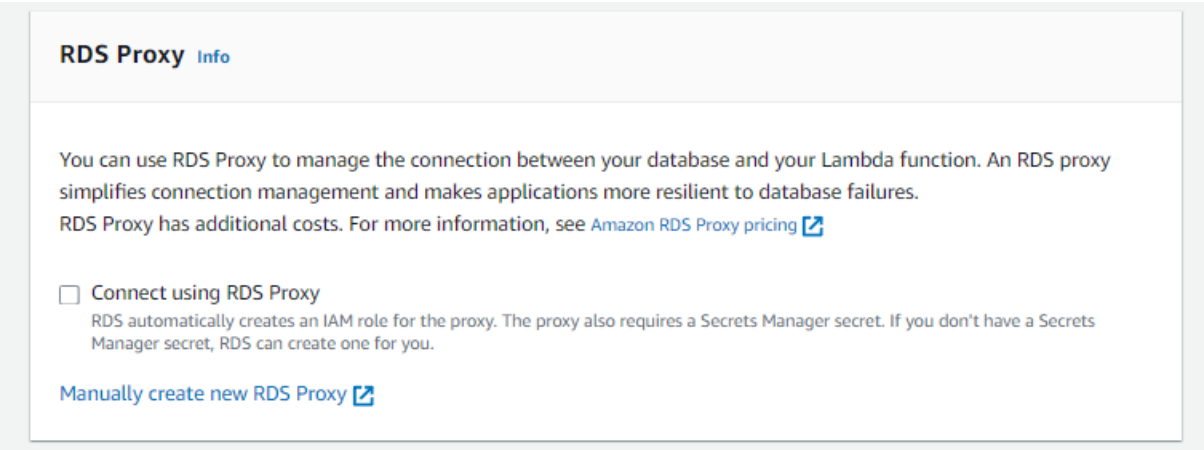
2. Click on the **Actions** dropdown and select **Set up Lambda connection.**



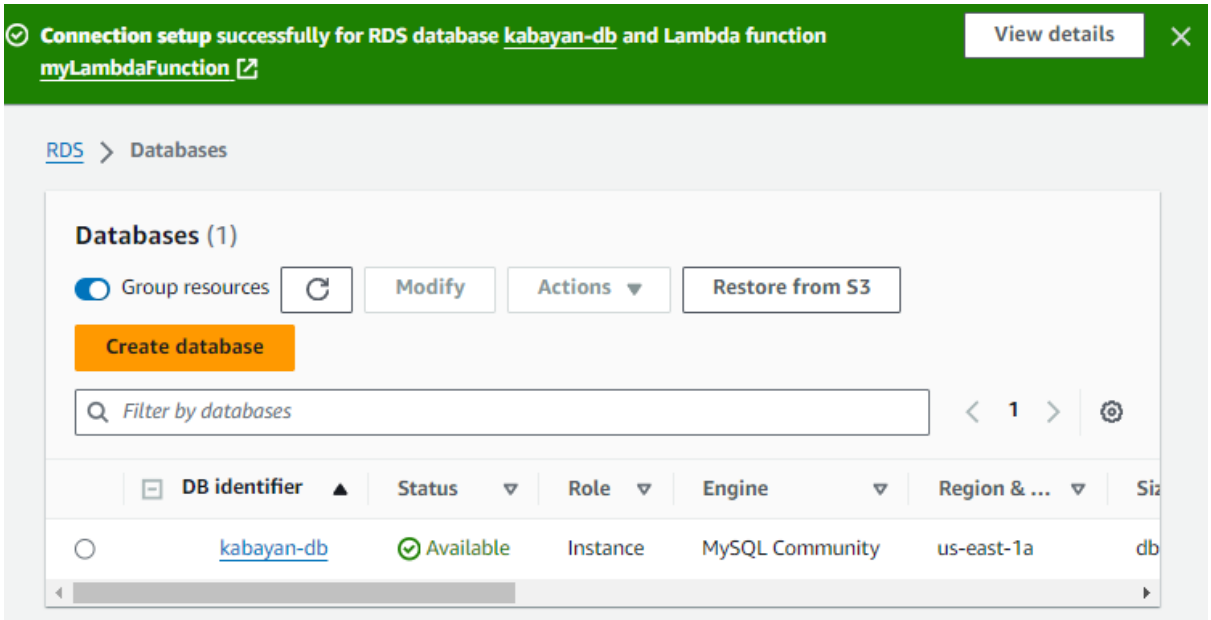
3. Add the Lambda Function we created earlier (i.e myLambdaFunction)



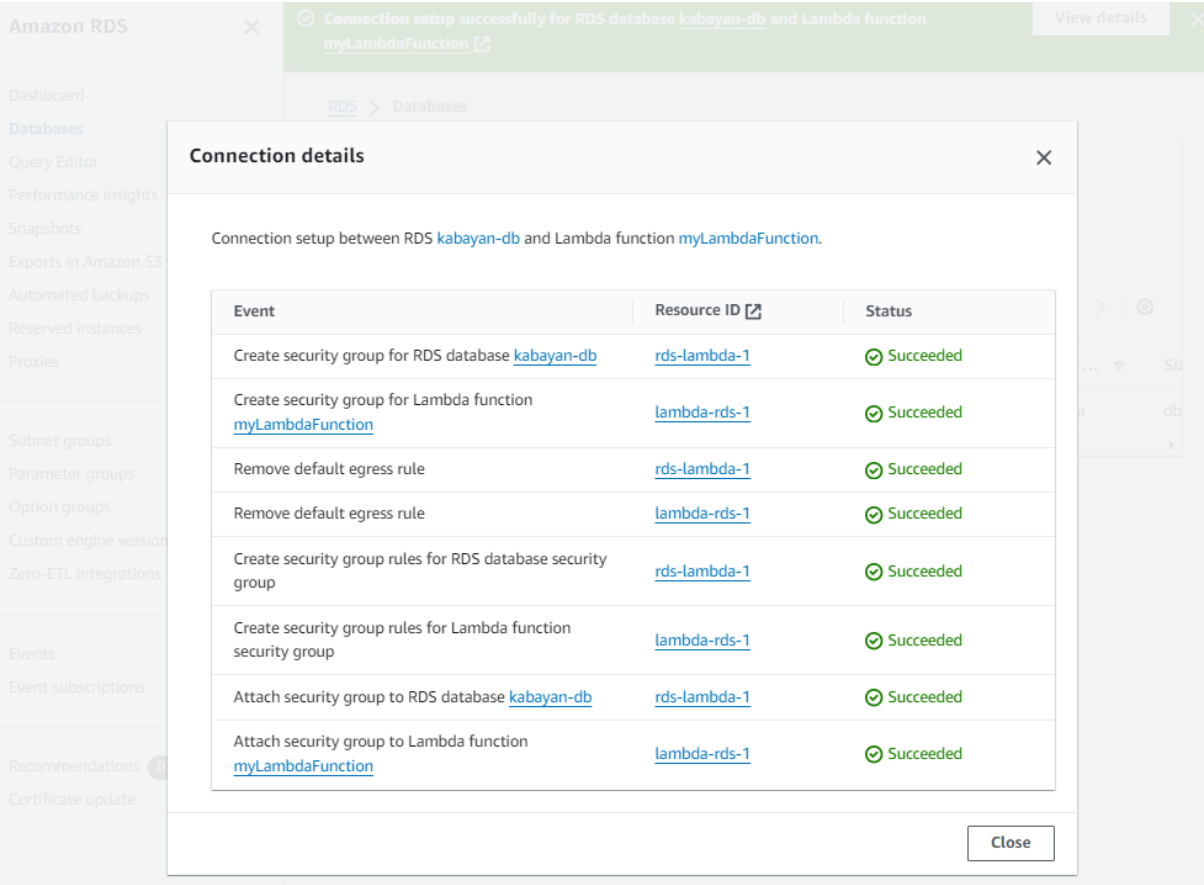
4. For this lab, **uncheck** the **Connect using RDS proxy**



5. Click **Set up** and wait for the connection setup to succeed.



- You can also view the details.

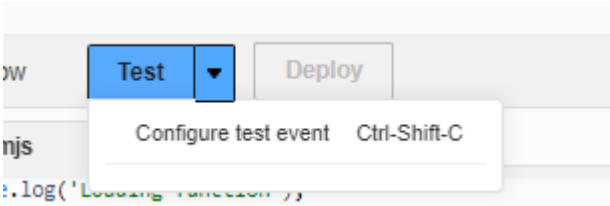


6. Navigate back to the Lambda Function Console and **wait for the Lambda Function to finish updating.**

Take Note that the Lambda Function will also be updated

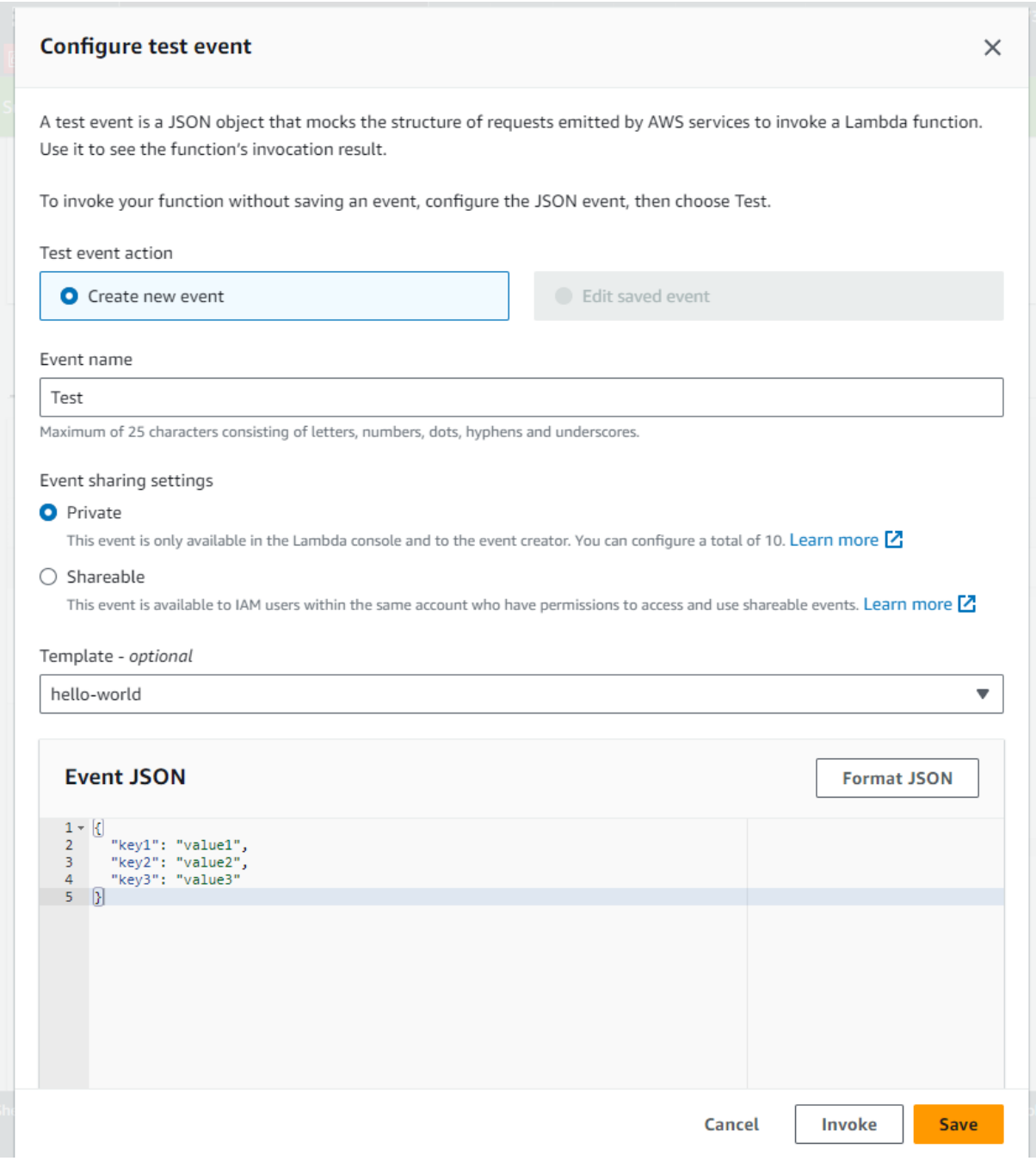
Test the Lambda Function

1. Navigate to the Code tab and click the arrow dropdown of the BLUE **Test button**



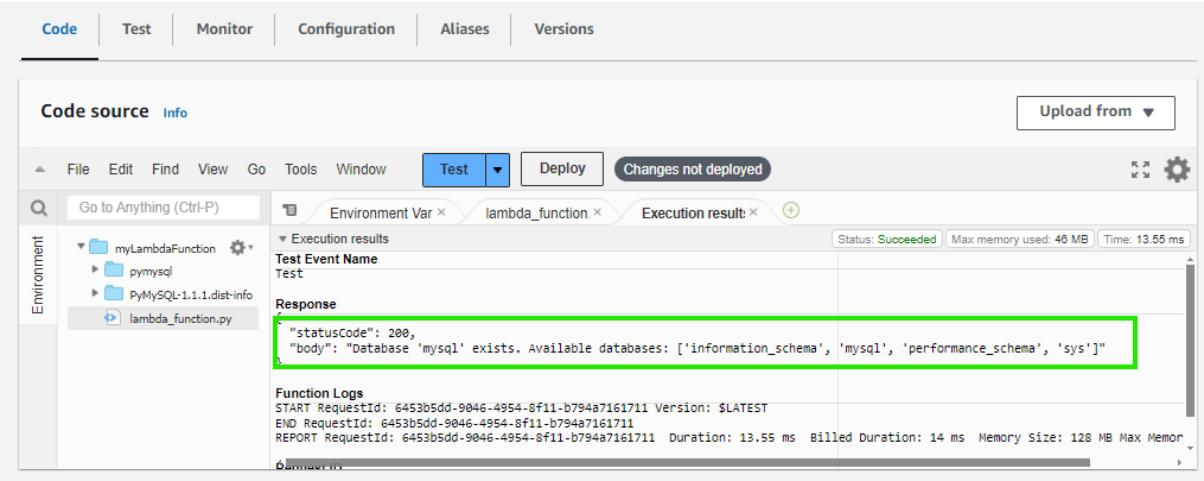
2. Click on **Configure test event,** and follow the configuration below:

- Event name: Test
- Template- optional: hello-world
    - Leave the rest as default



- Click on Save

3. Now, click on **Test** and check the output to verify the successful execution of the SQL query.



That's it! Congratulations! You have gained hands-on experience in securely connecting an AWS Lambda function to an Amazon RDS database. You've learned how to manage database credentials using environment variables, ensuring that sensitive data like passwords are not hardcoded into your application code and enhancing security and maintainability.

This setup is crucial for building scalable, serverless applications that interact with relational databases, enabling use cases such as real-time data processing, API backends, and other data-driven applications. By understanding how to configure Lambda and RDS in a secure and efficient manner, you are now equipped to create dynamic, data-driven solutions in a serverless environment.