COURSEWARE

- Professional Skills
- Agile Fundamentals
- Jira
- Git
- Databases Introduction
- Java Beginner
- Maven
- Testing (Foundation)
- Java Intermediate
- HTML
- CSS
- Javascript
- Spring Boot
- Selenium
- Sonarqube
- Advanced Testing (Theory)
- Cucumber
- MongoDB
- Express
- NodeJS
- React
- Express-Testing
- Networking
- Security
- Cloud Fundamentals
- AWS Foundations
- AWS Intermediate
  - Virtual Private Cloud (VPC)
  - EC2 VPC Security Groups
  - EC2 VPC Subnets

AWS Serverless CRUD Solution

Contents

Overview

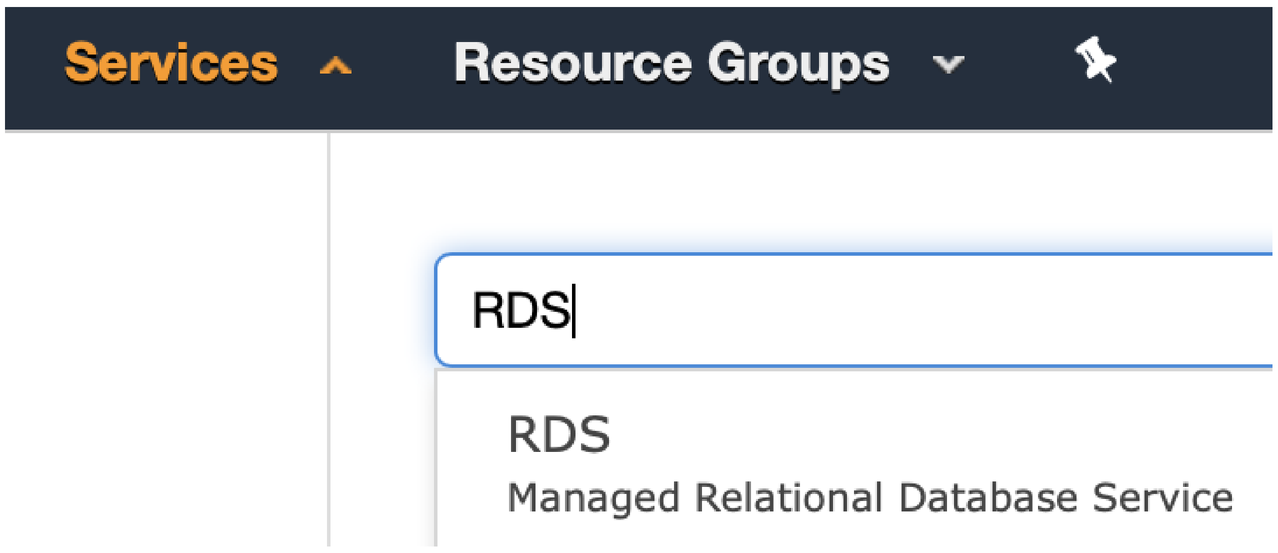
After discussing topics such as AWS Lambda, API Gateway and RDS, you may now realise that these tools are quite versatile and could be used to potentially create a serverless CRUD (Create, Read, Update, Delete) implementation.

In this tutorial, we will be going through the steps that will help you create an API that triggers lambda functions, in order to manipulate data on a database, or more specifically an RDS instance.

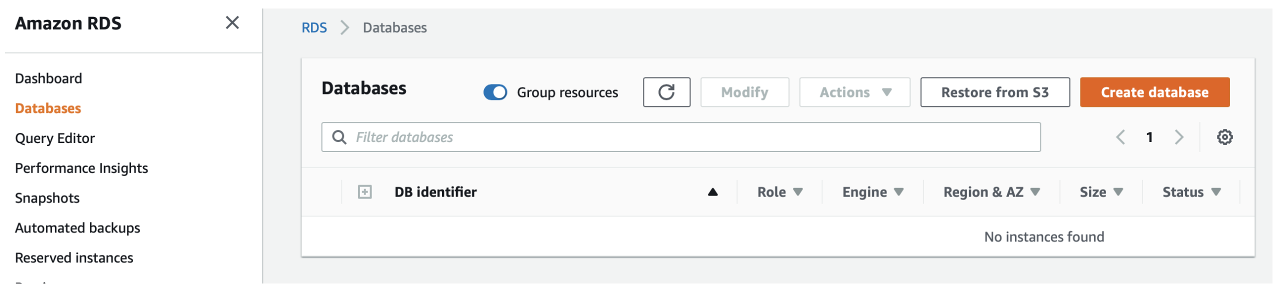
Tutorial

1. Navigate to the AWS Console and sign in here

2. Search for RDS under the services drop-down menu, and click on RDS



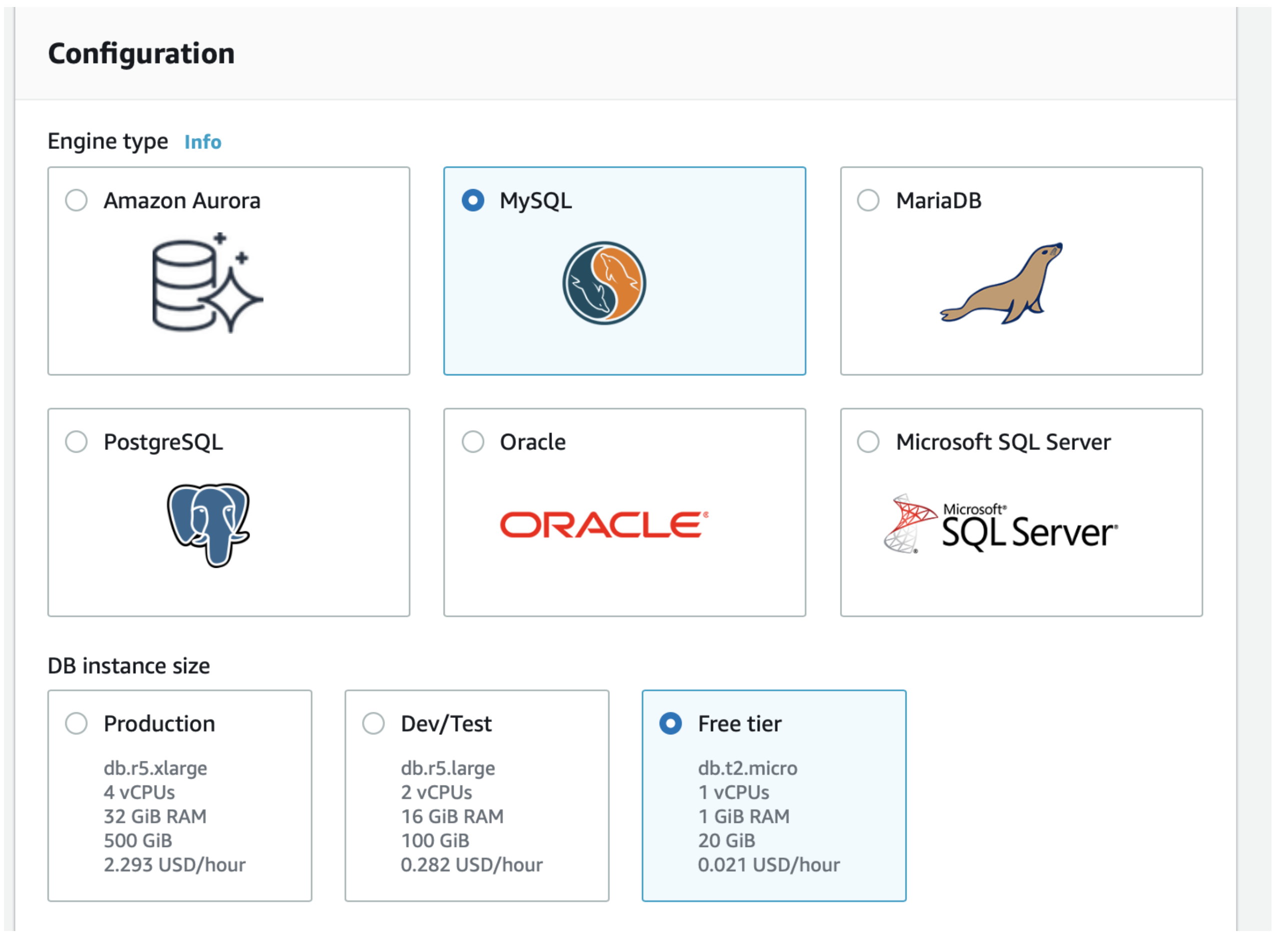
3. You will be redirected to the RDS homepage. On the left panel, click on **Databases**, this will show all of your running databases. Since you have no running databases, we will need to create one. So click on **Create database**



4. By default, the **Standard Create** is checked, in the interest of this speed, we will leave it on standard create. This allows us to tick the **Publicly Accessible Box** so that we don't have to create an EC2 instance in the same VPC in order to connect to the RDS instance.

IMPORTANT NOTE: Please don't follow this if you're planning on using this is in the future and are storing sensitive data on the database, you would normally not have this selected, you would need to configure your security groups correctly to allow inbound traffic from your IP address, so you can logon to the db instance**

5. Select **MySQL** as your Database Engine tool. And this is only an example, therefore select **Free tier**



Configuration

Engine type Info

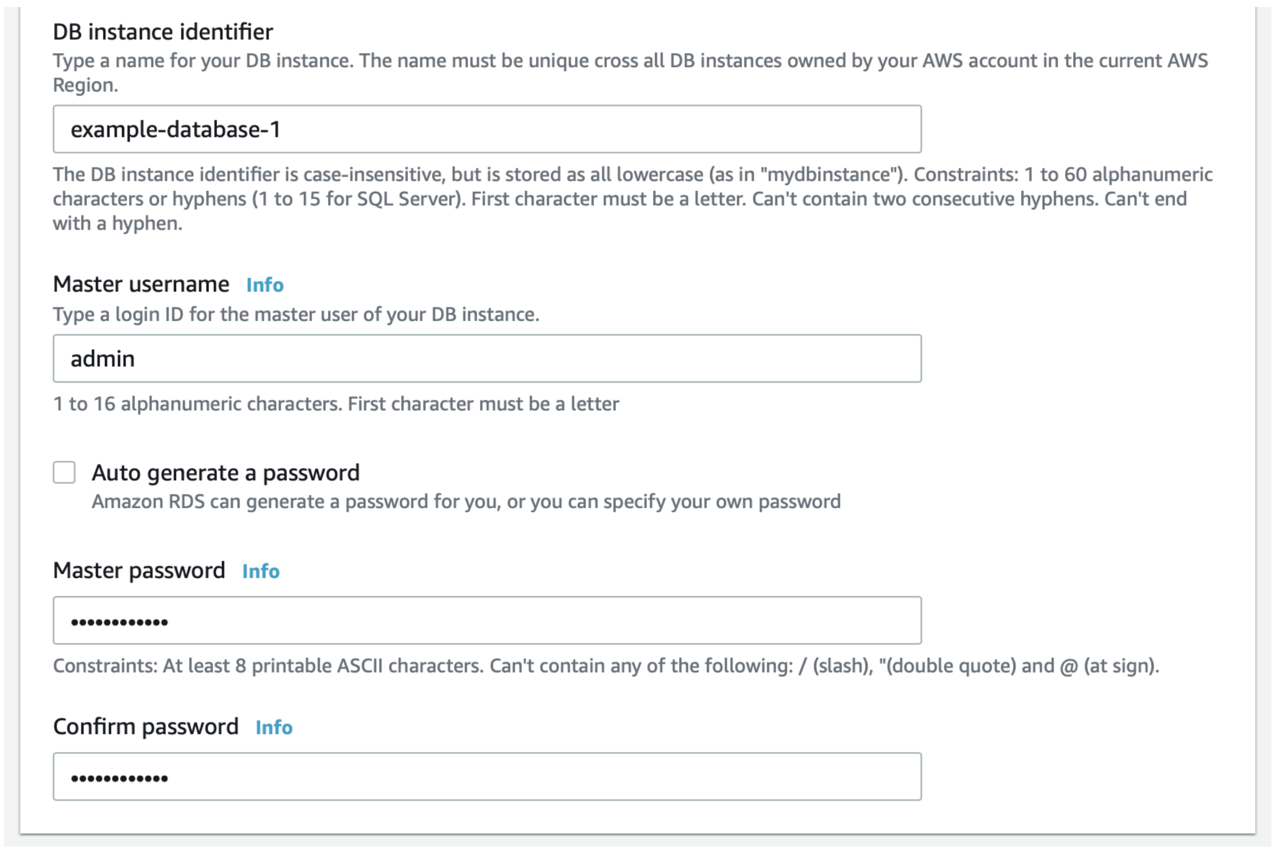
Amazon Aurora [✓ MySQL] MariaDB PostgreSQL Oracle Microsoft SQL Server

DB instance size

Production	Dev/Test	[✓ Free tier]
db.r5.xlarge	db.r5.large	db.t2.micro
4 vCPUs	2 vCPUs	1 vCPUs
32 GiB RAM	16 GiB RAM	1 GiB RAM
500 GiB	100 GiB	20 GiB
2.293 USD/hour	0.282 USD/hour	0.021 USD/hour

6. You will need to fill the rest of the credentials. You can leave the master username as **admin** or you can change it to whatever you want, keep in mind that you will need these credentials when writing your lambda function.

Note: Please remember your credentials for the account and password, as you will need this to login to your database and to use the pymysql library to logon and interact with your database within your lambda function. Then click **Create Database**.



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

example-database-1

The DB instance identifier is case-insensitive, but is stored as all lowercase (as in "mydbinstance"). Constraints: 1 to 60 alphanumeric characters or hyphens (1 to 15 for SQL Server). First character must be a letter. Can't contain two consecutive hyphens. Can't end with a hyphen.

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

admin

1 to 16 alphanumeric characters. First character must be a letter

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

Master password Info

••••••••••••

Constraints: At least 8 printable ASCII characters. Can't contain any of the following: / (slash), "(double quote) and @ (at sign).

Confirm password Info

••••••••••••

7. You should be able to run the following command to install mysql on your machine.

```
sudo apt install mysql-client-core-5.7 -y
```

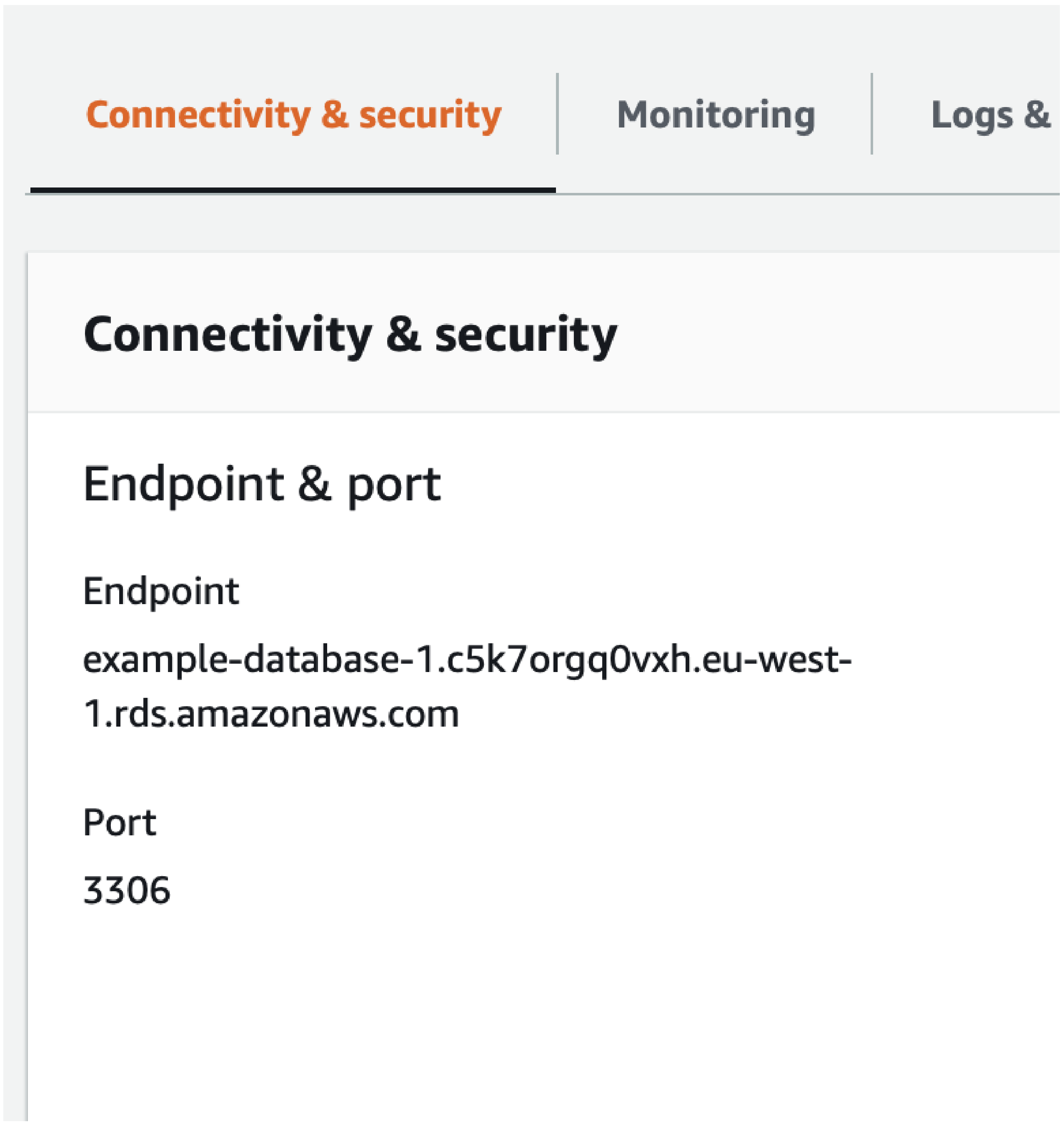
8. Now you can run **mysql** commands in the terminal. To connect to your database, you will need to run the following command:

NOTE: Please use your own RDS instances' endpoint as the host, this comes after the **-h**.

```
mysql -h <your_db_endpoint> -P 3306 -u <your_master_user> -p
```

The endpoint can be found by going to the list of databases, click on the name which you gave to your database and you will be shown more details about your database. You will then need to find your endpoint under **Connectivity and security**.

The username is what you created during the process in step 6. And after running that command you will be prompted to enter your password.



Connectivity & security

Endpoint & port

Endpoint

example-database-1.c5k7orgq0vxh.eu-west-1.rds.amazonaws.com

Port

3306

9. You have just been able to connect to the RDS instance through the terminal, meaning you can start running commands that can add data to your database without having to manage an instance that has MySQL running in it.

10. With reference to the example python script created for this task found within this current file path **/post_lambda.py**, you're going to need to create a database on the mysql shell. Run the following command:

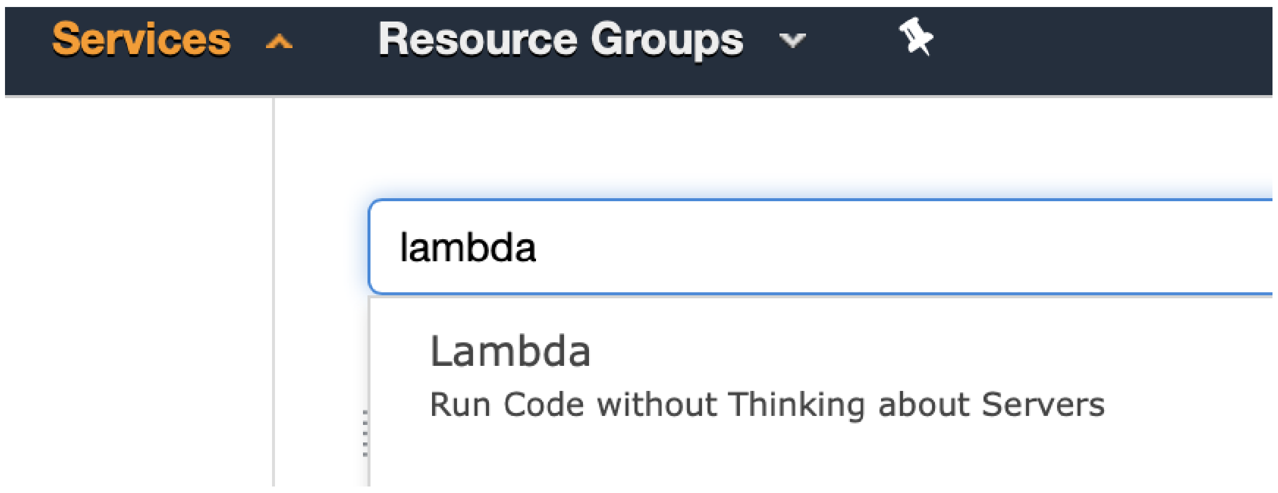
```
CREATE DATABASE IF NOT EXISTS <your_chosen_database_name>
DEFAULT CHARACTER SET utf8
DEFAULT COLLATE utf8_general_ci;
```

11. You have now created a database, if you're planning on using the python script already created, referenced in step 11. You may notice that you need to create a table called "test". Run the following command:

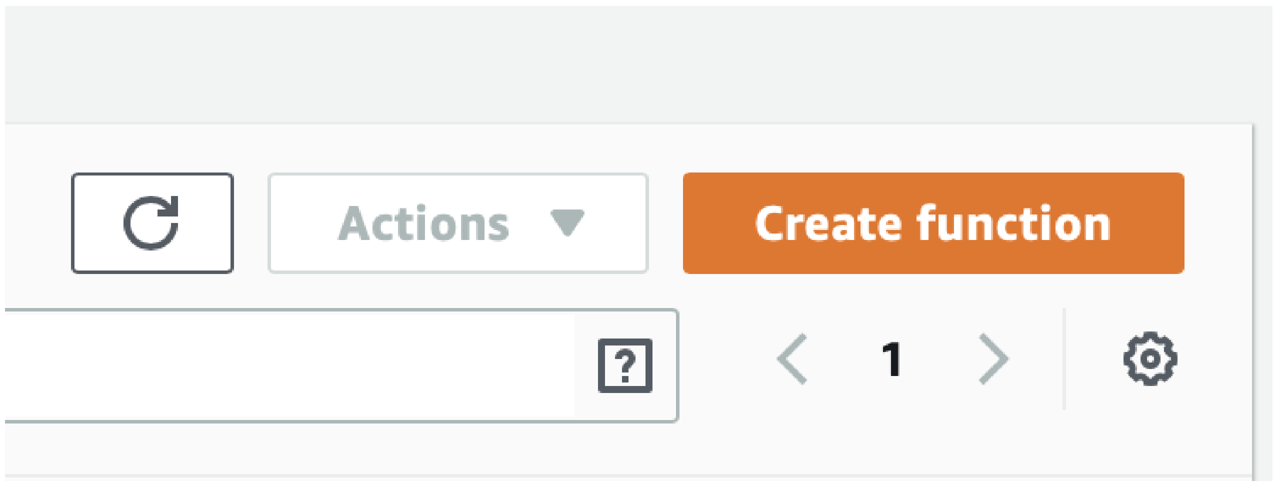
```
create table test (id int NOT NULL, name varchar(255), primary key (id));
```

NOTE If you're not using the script noted in step 11, then please adjust the previous commands accordingly.

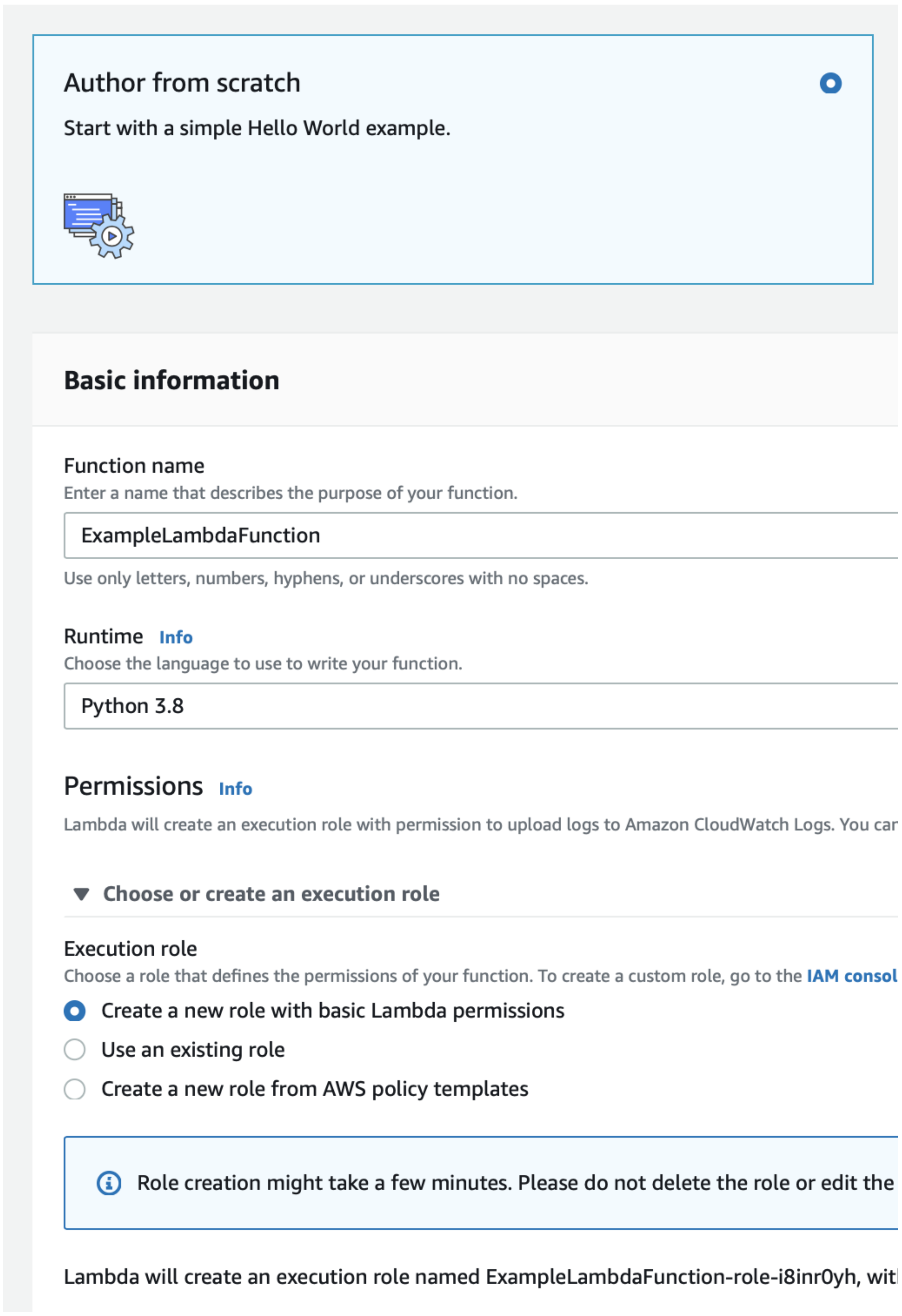
12. Now that we have configured the database to meet our needs, search for **Lambda** under the services drop-down menue



13. This next window, you will need to click the **Create function** button



14. You will then be presented with options on how you would like to create your Lambda function.
    1. Check **Author from scratch**
    2. Function name, choose your own function name, here it will be, **ExampleLambdaFunction**, but I strongly suggest you adhere to naming conventions that will help you identify the function related to the HTTP method it will be performing e.g. **Lambda-Gateway-Post-Create-User**
    3. Runtime, you have a large selection, but we will be making a python function, select **Python 3.8**
    4. When choosing an IAM Role for your function, make sure choose an IAM Role that atleast has Read and Write access to RDS, as well as the Lambda Execute Permission, you can then use this role later when you go on to create the rest of the lambdas for the remaining HTTP methods. **You can choose the default and then modify your IAM Role afterwards**


Author from scratch

Start with a simple Hello World example.

Basic information

Function name

Enter a name that describes the purpose of your function.

ExampleLambdaFunction

Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime Info

Choose the language to use to write your function.

Python 3.8

Permissions Info

Lambda will create an execution role with permission to upload logs to Amazon CloudWatch Logs. You car

▼ Choose or create an execution role

Execution role

Choose a role that defines the permissions of your function. To create a custom role, go to the IAM consol

[✓ Create a new role with basic Lambda permissions]

Use an existing role

Create a new role from AWS policy templates

Role creation might take a few minutes. Please do not delete the role or edit the

Lambda will create an execution role named ExampleLambdaFunction-role-i8inr0yh, wit

15. Proceed by clicking **Create function**
16. This next window, we will be focusing on the **Function Code** section. We are going to need to need to provide some code for the lambda, so that it actually performs some sort of "POST" action.
17. The containers that lambda runs on only have access the built-in python libraries and so we are going to need to create a deployment package for our lambda, so that it can perform "pymysql" functions to interact with the rds instance.

    1. Traverse to the directory in whcih you have written or copied your python code.
    2. Run the following command:

    ```
    pip3 install pymysql -t .
    ```

    NOTE The "." represents the current working directory you are in, which should be where your function code is, if you are not within that directory, then instead of "." write the relative or absolute path.

    3. You will now see that the pymysql library has been installed to whatever directory you specified and should be alongside your python code.

   4. Select all of the pymysql folders, as well as your function code and zip/compress them. You have now created your lambda deployment package.
   5. Go back to the lambda console and select the drop down menu titled **Code entry type** in the **Function code** section. "Select Upload .zip file" and choose the deployment package you just created.
18. You should remember from the lambda tasks that you need to ensure that the handler is specified correctly, in order to invoke your function. Check that the handler is **post_lambda.main** if you have copied the python code or ensure that it meets the standards of **.**

NOTEThis may take a little longer than usual as you're not just uploading your code, but some extra libraries as well

19. Your lambda function should now be ready to be triggered, if you have added the correct credentials for the variables passed into the "pymysql.connect" method. If you have copied the code, you can create a test event:
    ```
    {"id": "1",
    "name": "Bob"
    }
    ```
    You should see that Bob has been added to the database with an id of 1. Obviously, if you have not copied the code then change the test event to match your configuration.

NOTEYou can double check this by running **select * from test;** from your mysql shell if you have followed the tutorial completely, otherwise adjust test to whatever your table name is.

20. Now you have created your lambda and your rds instance, it's time to move on to our api-gateway configuration.

21. Navigate to the AWS Console and sign in here

22. Search for API Gateway under the services drop-down menu, and click on API Gateway

![api-gateway-search][api-gateway-search]

23. You will be redirected to the **API Gateway** homepage.
    This will only show your current API's which you have created.
    Click on **Create API**

24. We are going to build a **REST API**.
    Click **Build** under this section to get started.

25. Leave everything as default and only provide a name for the API you are creating.
    Continue to **Create**

26. Click on actions and **create resource**, give your resource/endpoint a relative name, in this case we are making a POST method, so maybe something **post_user**

27. Click on action and **create method**.
    This will then give you an option of the type of REST API method you want to make. In this case **POST**.

28. You will then be met with the "Integration Request" menu, make sure lambda function is selected and in the lambda function drop down, start typing your lambda function name and then you should be able to select it.

You've now configured an endpoint on your API that performs the actions configured in your lambda function.

29. In order for **API Gateway** to be properly utilised, you will need to deploy this API.
    Click on **Action** and you have the option to **Deploy API**.

![api-gateway-deploy-api][api-gateway-deploy-api]

30. You will be prompted to select an existing stage or create a new stage for your API.

![api-gateway-new-stage][api-gateway-new-stage]

31. Continue to **Deploy**

32. The next window will show you the stage which you have just created.
    And at the top you are given the **API URL**.
    When the URL is called it will be as if you called the API for your application, which you created previously when creating your **POST** Method.

I would advise download a tool to perform HTTP requests, like postman which can be downloaded here:

```
https://www.getpostman.com/downloads/
```

Just select the right OS for your machine and skip the sign in step.

33. If you combine the **API URL** and the **resource endpoint** in postman then you will be able to send events to your lambda function.
    If you copied the python script made available to you then you can just send a message body like this:

```
{"id": "2",
 "name": "Bill"
}
```

If you remember that you triggered your test event before, which should have added a user id of "1" tied to a user called "Bob".

Then if you perform select * from then you should see two users have been created, Bob and Bill.

NOTE This obviously depends on how you have configured your database, but if you write the correct key pairs to match what you have written then you should be able to add data to your database.

Resource

- RDS
- Lambda
- API Gateway

Exercises

Expand on your API Gateway, by creating a GET, PUT and DELETE method that are tied to individual Lambda Functions that communicate with your RDS database to create a full serverless CRUD Implementation. Please make your lambda functions relevant to the type of HTTP method used, otherwise this can become confusing.