

Metropolitan State University

ICS 432 - 01: Distributed and Cloud Computing

Fall 2021

Lab 08: Using Cloud Relational Database System (RDS)

Total points: **35**

Out: Saturday, October 23, 2021

Due: 11:59 PM on Friday, October 29, 2021

What to submit?

The objective of this lab is to practice using virtual machines on the cloud. To complete this lab:

- Read this lab assignment carefully.
- At various parts of the lab, you are asked to take screen shots of your work. Open a word document and paste the screen shots in this document in the same order as mentioned in the lab. Make sure to highlight the screen shot number.
- After you complete all the lab exercises, upload the word document to the designated D2L folder by 11:59 PM on Friday, October 29, 2021.

NOTE: On Windows machines, you may consider using [Snip & Sketch](#) for screenshot handling.

Exercise 1: Relational Databases on AWS

Amazon Relational Database Service (Amazon RDS) makes it easy to set up, operate, and scale a relational database in the cloud. The goal of this exercise is to practice the following three different ways to interact with an Amazon RDS database:

- In part 1, you will connect to the database through a web application.
- In part 2, you will connect to the database through the command line from an EC2 instance
- In part 3, you will connect to the database through python script.

Part 1: Connecting to RDS through a web application

In this exercise, you will complete **Lab 5: Build a Database Server** in Module 8 on the AWS Academy course.

Note that, in this lab, you will only create the database server, however, the code to create the database tables and the implementation of the web application are already packaged inside the lab and you do not have the capability to add tables or do any interactions with the database server. In the next parts of the lab, you will practice creating database servers and you will have complete access to these RDS.

- 1- In Task 1, step 9, notice to the MySQL/Aurora rule as this is the first time to use this rule.
- 2- **Lab screenshot #1:** after step 10, take a screenshot to show the inbound rules in the security group.
- 3- In Task 3, after step 30, take **Lab screenshot #2** to show the details of your database instance. Make sure the Endpoint information is clearly displayed in the screenshot.
- 4- In Task 4, after step 35, take **Lab screenshot #3** to show the browser page showing the database table.
- 5- On the browser screen, click on [Add Contact](#) and add two more entries in the database table. Take Lab screenshot #4 showing the table with the new entries.

Part 2: Connecting to RDS through an EC2 instance

In this exercise, you will use your **AWS Educate** account to create a relational database server and, then, you will interact with your database using the command line from an EC2 instance.

- 1- Log in to AWS Educate and go to Services → RDS.
- 2- Create an RDS instance using the following parameters:
 - a. Standard Create
 - b. MySQL
 - c. FreeTier
 - d. Identifier: your-lase-name-database
 - e. Master username: admin
 - f. Master password: admin123
 - g. DB instance class: Burstable classes (db.t2.micro)
 - h. Uncheck Enable storage autoscaling
 - i. Public access: Yes
 - j. Security group: default
 - k. Password authentication
 - l. Uncheck enable automated backups

Lab screenshot #4: wait until the database sever creation is completed and then take a screenshot showing the details of your server. Make sure the Endpoint information is clearly displayed in the screenshot.

- 3- Copy your database server end point and paste it in your lab document. You will use this link to connect to the database.
- 4- Create an Amazon Linux EC2 instance.
- 5- Establish an SSH connection to your instance and execute the following commands to install the mysql client.

```
sudo yum update
sudo yum install mysql
mysql --version
```

Lab screenshot #5: take a screenshot of the command window showing the version of mysql.

- 6- Connect to your RDS using the following command. Replace the highlighted text with your endpoint.

```
mysql -u admin -p -h database-2.cbbfbaxspq8g.us-east-1.rds.amazonaws.com
```

- 7- Once connected to the database, you can start writing SQL queries to create tables and uploading them with data.
- Create a test database. Use your last name in the database name.

```
create database ghanemdb;
use ghanemdb;
```

- Create a test table.

```
create table ghanemtesttable (attr1 int, attr2 int,
attr3 varchar(10));
```

- Insert **three** rows in your table. This is an example insert statement.

```
insert into ghanemtesttable values (10,10,"Thanaa");
```

- Display the table.

```
select * from ghanemtesttable;
```

Lab screenshot #6: take a screenshot of the command window showing the execution of the previous SQL queries.

Part 3: Connecting to RDS using Python

- 1- Edit the RDS instance's security group by adding the following inbound rule.

The screenshot shows the 'Inbound Rules' section of an AWS Security Group. A rule is being added or edited with the following details:

- Rule Name: MySQL/Aurora
- Protocol: TCP
- Port Range: 3306
- Source: Anywh... (0.0.0.0/0)
- Action: Allow
- There is a 'Delete' button next to the rule.

- Create a Cloud9 environment or open an existing environment.
- In the Cloud9's command line, run the following command to install the MySQL connector Python library

```
pip install mysql-connector-python
```

Lab screenshot #7: take a screenshot of Cloud9's command window showing the correct installation of the Python library.

- 4- Create a python file with the following code to establish a connection to your database. Replace the host with your database endpoint.

```
accessdb.py
1 import mysql.connector
2 from mysql.connector.constants import ClientFlag
3
4 config = {
5     'user': 'admin',
6     'password': 'admin123',
7     'host': 'database-1.cwb57hbfc8ix.us-east-1.rds.amazonaws.com'
8 }
9
10 # establish a connection to the database
11 db_connection = mysql.connector.connect(**config)
12
13 # initialize connection cursor
14 cursor = db_connection.cursor()
15
16 #execute a query to list existing databases.
17
18 cursor.execute("show databases")
19 out = cursor.fetchall()
20 for row in out:
21     print(row)
22
23
24 # close connection
25 db_connection.close()
26
```

Lab screenshot #8: take a screenshot of your python code.

Lab screenshot #9: take a screenshot of the command window showing the output of running your code. You should see a list of databases on your server including the database you created in step 2.

Part 4: Executing more queries on your database

- 1- Examine the python code you wrote in the previous step. To run an SQL query:
 - a. Write the SQL query string as input in `cursor.exeucte`.
 - b. Collect the query output using `cursor.fetchall()`
 - c. Display the output using a for loop.

- 2- Add a select query to your code to display all rows from the table you created in step2. Note that you have to execute `use databasename` before executing the `select` query.

Lab screenshot #10: take a screenshot of your python code and the output of running the code.

- 3- Create another python file and establish a connection to the database server.
- 4- Write a for loop to insert 3 more rows in the table. Your loop should ask the user to enter values for the three attributes and then execute a query to insert the values in the table. Make sure to correctly construct the string of the insert query using the three values that are entered by the user. Before sending the insert query to the database, print the string on the screen first to make sure the commas and quotations around the string attribute are placed correctly.
- 5- Run a select query to show all the rows in your table.

Lab screenshot #11: take a screenshot of your python code.

Lab screenshot #12: take a screenshot of the command window showing the output of running your

Lab screenshot #13: go to the EC2 instance and execute a select query to show all rows in the table including the rows inserted by your python code. Take a screenshot of the output of the select query.

Exercise 2: Relational Databases on GCP

In this exercise, you will work with MySQL on GCP. You may choose to complete the following

optional Qwiklab: Cloud SQL for MySQL: Qwik Start:

<https://www.qwiklabs.com/focuses/936?parent=catalog>

Part 1: Creating a database and uploading data

- 1- Log in to your GCP account, create a new project, and make sure you are using the Billing Account for Education.
- 2- From the main menu, select **Databases → SQL**.
- 3- Click **CREATE INSTANCE**.
- 4- You will be prompted to choose a database engine. Select **MySQL**.
- 5- Enter **instance ID** (use your last name) and enter a secure password in the **Password** field (make sure to remember this password). Remember instance ID because you are going to use to connect to the instance.
- 6- Choose single zone
- 7- Click **CREATE INSTANCE**:

Lab screenshot #14: wait until the instance creation process is completed (which may take up to 5 minutes) and then take a screenshot showing your database instance.

- 8- Copy your instance's Public IP address because you will use it later.
- 9- Activate cloud shell

In the command window, run the following command to connect to your MySQL instance. The cloud shell will automatically install required tools to connect to your instance.

```
gcloud sql connect instance ID --user=root
```

If you cannot connect, run the following command to check the current environment's project:

```
gcloud config list project
```

If you are not in the same project as the database instance, run the following command to change the environment to point to the current project:

```
gcloud config set project your-project-ID
```

10- When prompted, enter your database password.

11- Once connected to the database server, create a database to upload movie rating data. Use your last name in the database name.

```
create database ghanemmovierating;
```

12- Create the users table:

```
use ghanemmovierating;

create table users(
    userid int,
    age int,
    gender varchar(5),
    occupation varchar(100),
    zipcode varchar(10));
```

13- Upload the `users.csv` file (from assignment 1) to a bucket on Google storage.

14- In your Cloud SQL instance dashboard click **IMPORT**.

15- In the Cloud Storage file field, click **Browse**, and then click the arrow opposite your bucket name, and then click `users.csv`. Click **Select**.

16- Select **CSV** as File format.

17- Select your database and type in `users` as your table.

18- Click **Import**.

19- Go back to the command window and run the following query to make sure the data is successfully imported.

```
select count(*) from users;
```

Lab screenshot #15: take a screenshot of the command window showing your SQL command and the output.

Part 2: Writing python code to connect to the database

- 1- In order to connect to your database server from python, you first need to set up the connection.
- 2- **Enabling IP Access:** in your Instance Details page, click on the **Connections** tab (on the left-taskbar) and click + **Add network** under Public IP. Give the network a name and enter the IP address range you would like to allow. For this assignment, enter **0.0.0.0/0** to allow entry to all IP addresses. Of course, this is a pretty big security issue, which is why we use SSL encryption to restrict access to our instance.
- 3- While still on the **Connections** tab, scroll down to the **SSL** section. Uncheck **Allow only SSL connections** and then scroll down to Manage Client Certificates and click on **Create a client certificate** — give it a name (e.g., lab7sqlcertificate) and click **create**. Download all three .pem files because you will use them when connecting with Python. Also copy the `mysql` connection command that is similar to. Note that the IP address should be the same as your instance's IP address that was copied in a previous step.

```
mysql -uroot -p -h 34.69.37.239 --ssl-ca=server-ca.pem --ssl-cert=client-cert.pem --ssl-key=client-key.pem
```

SSL

SSL connections
SSL encryption is recommended when using Public IP to connect to your instance.

⚠️ Unsecured connections are allowed to connect to this instance. **Allow only SSL connections** 1

Configure SSL server certificates
The server Certificate Authority (CA) certificate is required in SSL connections.

Create new certificate Rotate certificate Rollback certificate

	Created	Expires
Upcoming		No certificate
Active	53 minutes ago	29 Aug 2030, 09:57:13
Previous		No certificate

Download SSL server certificates
You can download a server-ca.pem file of all available SSL server certificates.

Download

Configure SSL client certificates
An SSL certificate is composed of a client certificate and client private key. Both are required for SSL connections. For existing client certificates, you can access only the client certificate. The client private key is only visible during certificate creation.

Create a client certificate 2

Reset SSL configuration
Resetting the SSL configuration of the server revokes all client certificates and creates a new server CA certificate.

Reset SSL configuration

Download client-key.pem

```
-----BEGIN RSA PRIVATE KEY-----
MIIEogI
p1YZa9w
-----
```

Download client-cert.pem

```
-----BEGIN CERTIFICATE-----
MIIDbTC
NTdkHjV
-----
```

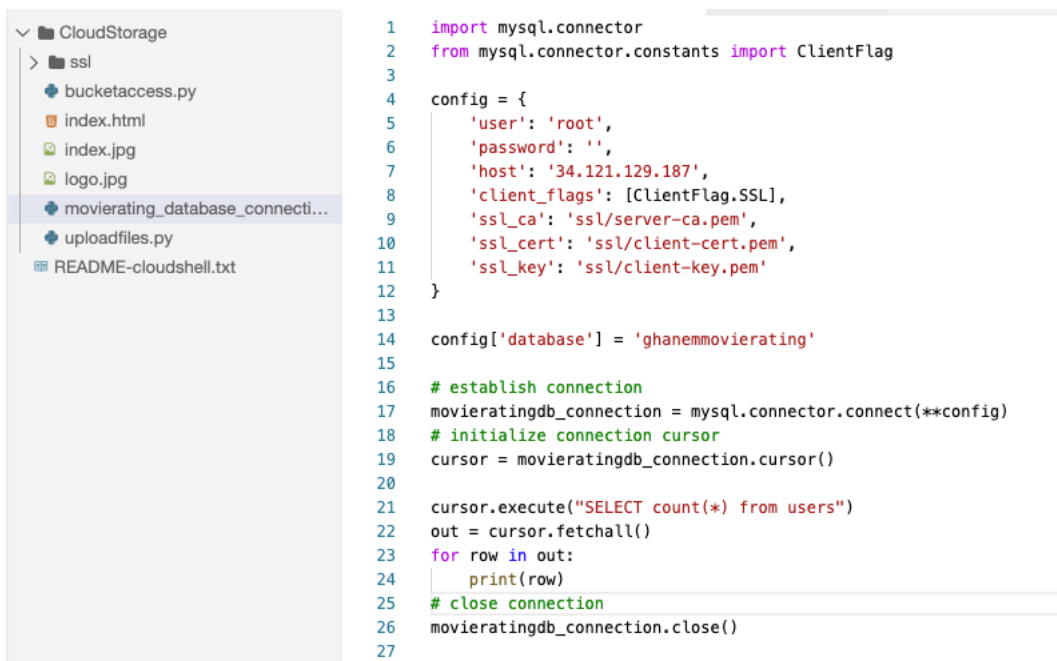
Download server-ca.pem

```
-----BEGIN CERTIFICATE-----
MIIDFzC
MTkxM11
-----
```

- 4- Next, you will write Python code to connect to your database. Open editor. Create a new file called `databaseconnection.py`.
- 5- In the editor file system, create a new folder called `ssl` and upload the three `.pem` files into that folder.
- 6- In the command window, run the following command to install the mysql connector python library. If you are still inside MySQL, type `exit` to go back to the command prompt.

```
pip install mysql-connector-python
```

- 7- To establish a connection to the database from a Python script, you create a config variable that include the following information:
 - a. Username — this should be `root`.
 - b. Password — we set this up earlier.
 - c. Host — the public IP address of our SQL instance. You can read this from the database instance details dashboard.
 - d. SSL certification — the three files, `server-ca.pem`, `client-cert.pem`, and `client-key.pem`.
- 8- Write the following code in `databaseconnection.py`.



```

1  import mysql.connector
2  from mysql.connector.constants import ClientFlag
3
4  config = {
5      'user': 'root',
6      'password': '',
7      'host': '34.121.129.187',
8      'client_flags': [ClientFlag.SSL],
9      'ssl_ca': 'ssl/server-ca.pem',
10     'ssl_cert': 'ssl/client-cert.pem',
11     'ssl_key': 'ssl/client-key.pem'
12 }
13
14 config['database'] = 'ghanemmovierating'
15
16 # establish connection
17 movieratingdb_connection = mysql.connector.connect(**config)
18 # initialize connection cursor
19 cursor = movieratingdb_connection.cursor()
20
21 cursor.execute("SELECT count(*) from users")
22 out = cursor.fetchall()
23 for row in out:
24     print(row)
25 # close connection
26 movieratingdb_connection.close()
27

```

- 9- In the command window, type `ls` and make sure the `databaseconnection.py` and the `ssl` folder are listed in your current directory. Then, run your python script using the following command.


```
python3 databaseconnection.py
```

Lab screenshot #16: take a screenshot of your python code and the command window showing the output of the query.

Part 3: Writing python code to run database queries

In this part, you will add code to your python file to run queries on the database. First, familiarize yourself with the code to connect to the database and run queries which is the same as the code you wrote in Exercise 1.

- 1- Write code to ask the user to enter a value for age then retrieve from the database all users younger than the entered age.

Lab screenshot #17: take a screenshot of your python code and the command window showing the output of the query.

- 2- Note that the output of `fetchall` is a two-dimensional array and you can read the individual attributes of each row. For example, the following for loop prints only the occupation of users because occupation is the fourth attribute in the `users` table.

```
for row in out:
    print(row[3])
```

Add code to your loop to count how many **students** are there is the output from the query.

Lab screenshot #18: take a screenshot of your python code and the command window showing the output of the query.

- 3- Write a `for` loop to enter more users to the table. Your code should first ask how many users to enter, read the number, and then use a `for` loop to read input from the user and insert rows to the `users` table. Note that, for each user, you need to read id, age, sex, occupation, and zip code. When you enter users, make sure to use id values > 1000 . To test your code, run a select query to display all users with $id > 1000$.

Lab screenshot #19: take a screenshot of your python code and the command window showing the output of the query.

Part 4: Connect to your GCP database instance from AWS

In this part, you will connect to your GCP database instance from Python code that is running in AWS Cloud9.

- 1- Create a python file in AWS Cloud9 and establish a connection to the Cloud SQL database the same way as in `databaseconnection.py`. Add code to run one query:

```
select count(*) from users
```

- 2- In Cloud9's command window, run the following command to install mysql connector. If you are using the same Cloud9 environment as Exercise 1 then the connector should be already installed.

```
pip install mysql-connector-python
```

- 3- In Cloud9's editor, create a folder called `ssl` and upload the three `.pem` files to that folder.
- 4- Run the code and you should be able to read data from the `users` table that is hosted in GCP Cloud SQL database.
- 5- Run another Two queries of your choice on the `users` table.

Lab screenshot #20: take a screenshot of your python code in Cloud 9 and the command window showing the output of the queries.