Task 4: Introduction to Cloud SQL Database in Google Cloud Platform

Activity: Using Cloud SQL with a Sample JDBC Application on Google Cloud

1. **Create a Cloud SQL Instance** (MySQL)
2. **Configure Database and User**.
3. **Create a Compute Engine VM** and **install Java & MySQL Connector**.
4. **Write a Sample JDBC Application** to connect to Cloud SQL.
5. **Run and test the connection**.

Step 1: Enable Cloud SQL API

1. Open **Google Cloud Console**.
2. Navigate to **APIs & Services → Library**.
3. Search for **Cloud SQL Admin API** and click **Enable**.

Step 2: Create a Cloud SQL Instance

1. Go to **Cloud SQL** in the console.
2. Click **Create Instance** → Select **MySQL** (or PostgreSQL).
3. Configure:
    - **Instance ID**: `my-sql-instance`
    - **Root Password**: Set a password
    - **Region**: Choose a preferred region
    - **Configure Machine Type**: `db-f1-micro` (free tier eligible)
4. Click **Create** and wait for provisioning.

Copy instance ID, username & password and **Save** in notepad

Once created, **copy the Public IP and Connection Name** (`<PROJECT-ID>:<REGION>:my-sql-instance`). **Save in note pad**

Step 3: Create a Database and User

1. **Access Cloud SQL** via Cloud Shell:

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

3. **Create a new database**:

```
CREATE DATABASE testdb;
```

5. **Create a new user**:

```
CREATE USER 'testuser'@'%' IDENTIFIED BY 'testpassword';
GRANT ALL PRIVILEGES ON testdb.* TO 'testuser'@'%';
FLUSH PRIVILEGES;
```

Step 4: Create a Compute Engine VM

Since Cloud Shell doesn't support running Java applications persistently, create a VM:

Create a new Compute Engine instance:

```
gcloud compute instances create jdbc-vm \
  --machine-type=e2-medium \
  --image-family=debian-11 \
  --image-project=debian-cloud \
  --tags=allow-sql
```

SSH into the VM:

```
gcloud compute ssh jdbc-vm
```

Step 5: Allow Compute Engine VM's IP in Cloud SQL

1. Go to **Cloud SQL** in the Google Cloud Console -> Cloud SQL Instances
2. Click on your **Cloud SQL instance**.
3. Navigate to **Connections → Authorized Networks**.
4. Click **"Add Network"** and enter your Compute Engine VM's external IP.
    - Find your VM's external IP:
5. Save changes and restart the instance.

Step 6: Install Java and MySQL Connector

On the VM, install Java:

```
sudo apt update && sudo apt install -y openjdk-17-jdk wget
```

Download MySQL **JDBC driver** and upload to VM using **Upload** Button

```
sudo dpkg -i  mysql-connector-j_9.2.0-1debian12_all.deb
```

Copy the `.jar` file:

```
$ sudo cp /usr/share/java/mysql-connector-java-9.2.0.jar  /usr/lib/
```

Step 7: Write a Sample Java JDBC Application

Create a Java file:

```
nano CloudSQLJDBC.java
```

Add the following Java code:

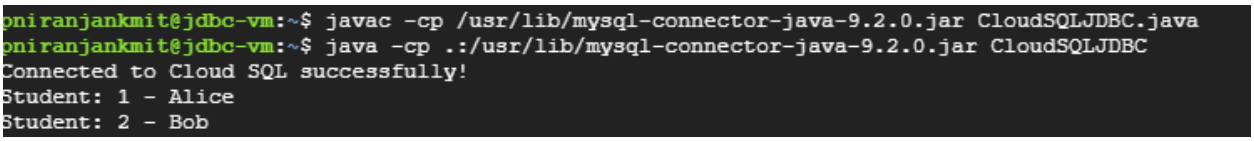
```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

public class CloudSQLJDBC {
    public static void main(String[] args) {
        String jdbcUrl = "jdbc:mysql://<PUBLIC_IP>:3306/testdb";
        String user = "testuser";
        String password = "testpassword";

        try {
            Connection conn = DriverManager.getConnection(jdbcUrl, user,
password);
            System.out.println("Connected to Cloud SQL successfully!");

            Statement stmt = conn.createStatement();
            stmt.executeUpdate("CREATE TABLE IF NOT EXISTS students (id
INT PRIMARY KEY AUTO_INCREMENT, name VARCHAR(50))");
            stmt.executeUpdate("INSERT INTO students (name) VALUES
('Alice'), ('Bob')");

            ResultSet rs = stmt.executeQuery("SELECT * FROM students");
            while (rs.next()) {
                System.out.println("Student: " + rs.getInt("id") + " - " +
rs.getString("name"));
            }

            conn.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Compile and Run the Program:

```
javac -cp /usr/lib/mysql-connector-java-9.2.0.jar CloudSQLJDBC.java
java -cp .:/usr/lib/mysql-connector-java-9.2.0.jar CloudSQLJDBC
```

Output

```
Connected to Cloud SQL successfully!
Student: 1 - Alice
Student: 2 - Bob
```



Step 8: Firewall Rules

If the connection fails, allow Cloud SQL traffic:

```
gcloud compute firewall-rules create allow-cloud-sql \
    --allow tcp:3306 --source-ranges=0.0.0.0/0 \
    --target-tags=allow-sql
```

Step 9: Clean Up

After testing, delete resources to avoid costs:

```
$gcloud compute instances delete jdbc-vm --quiet

$gcloud sql instances delete my-sql-instance --quiet
```