

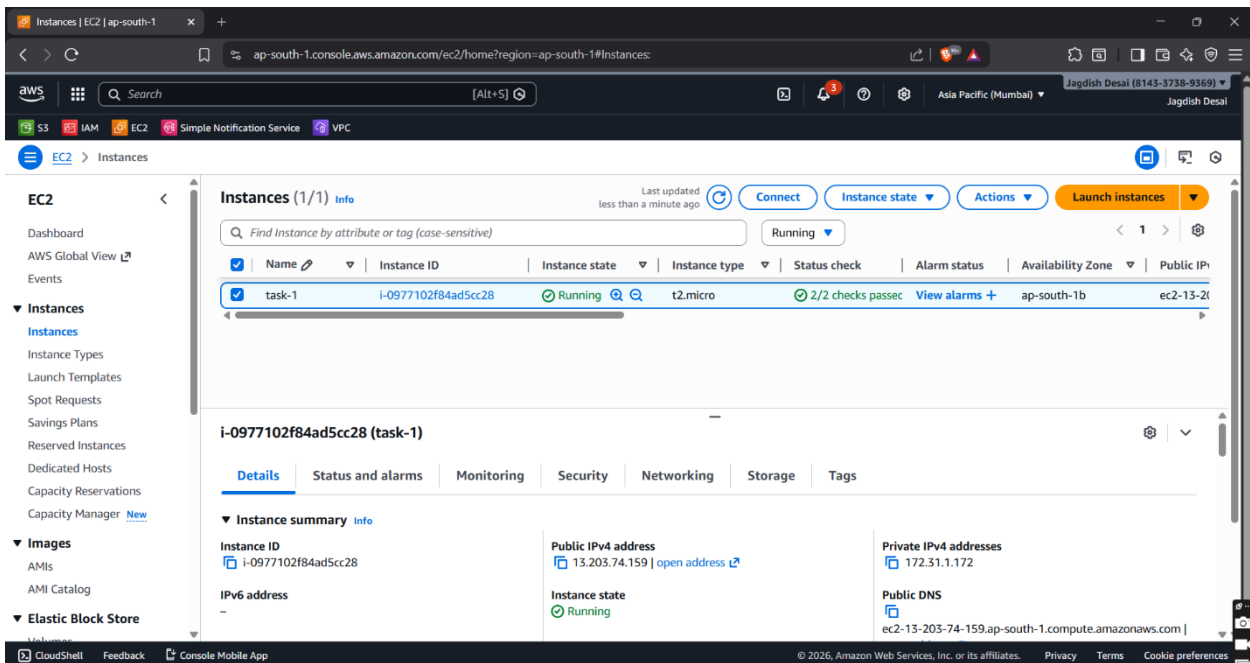
Task 1-

1. Application

- ❑ Simple frontend + backend app
- ❑ Use one database (MySQL / MongoDB / Influx DB)
- ❑ App must connect to the database successfully

Steps:

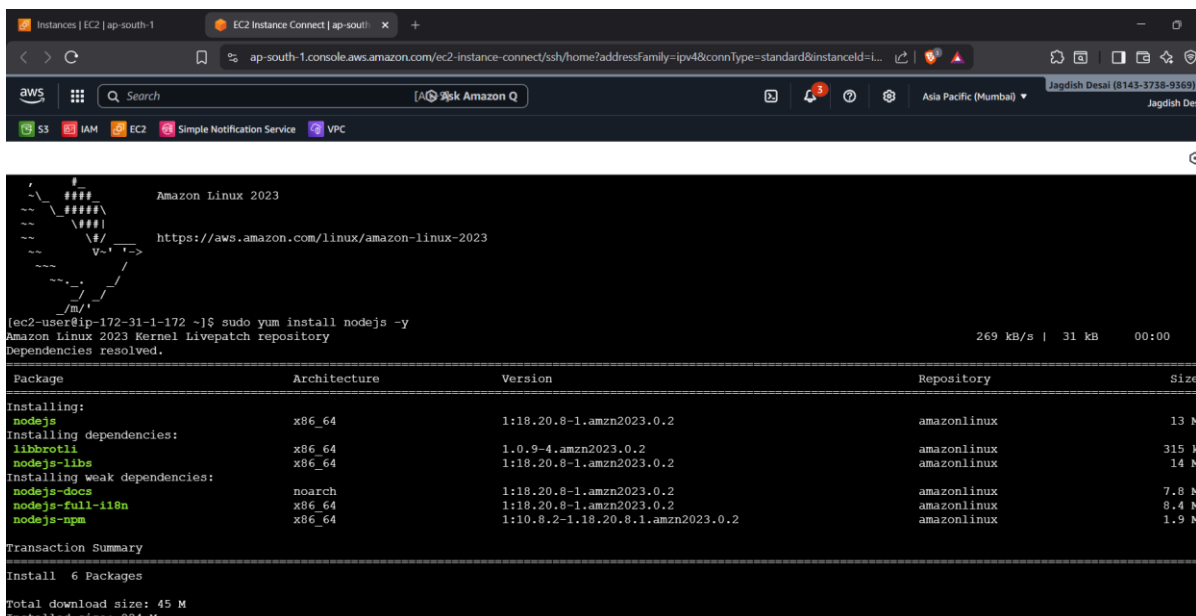
Step 1: Created an instance task-1



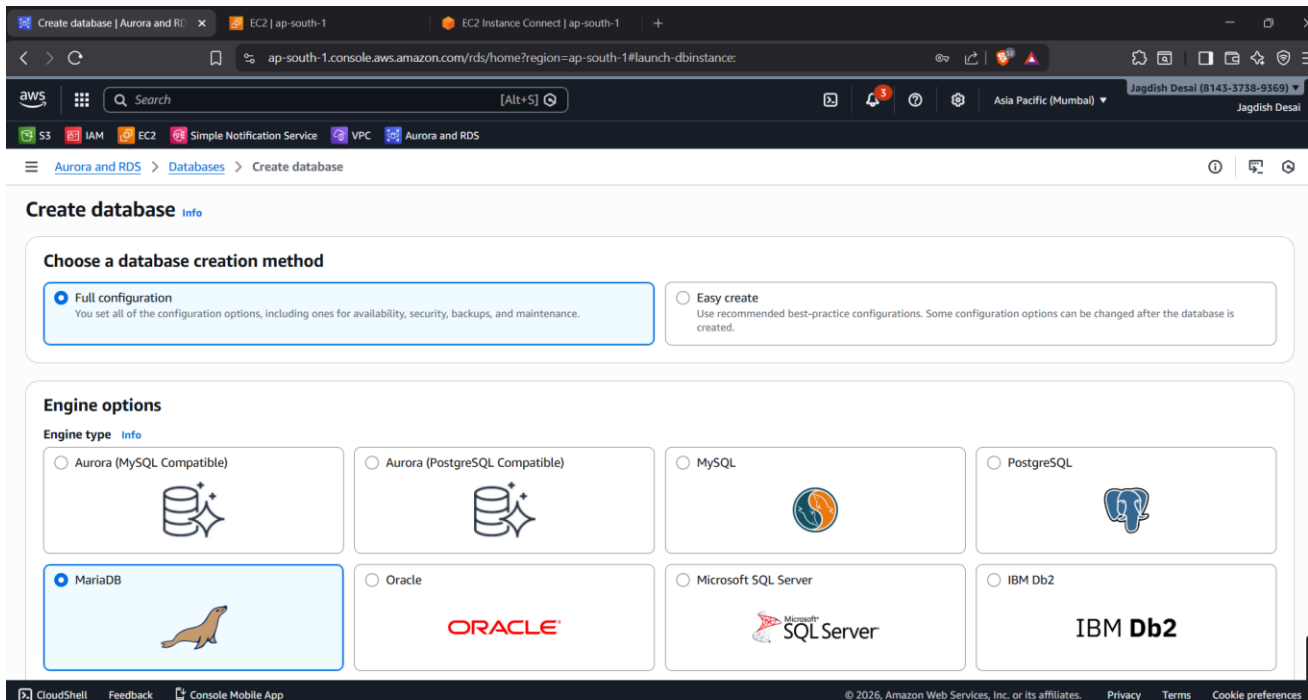
we

Step 2: Installing Node-js on task-1 instance

`sudo yum install nodejs -y`



Step 3: Going to RDS and creating a mariadb database

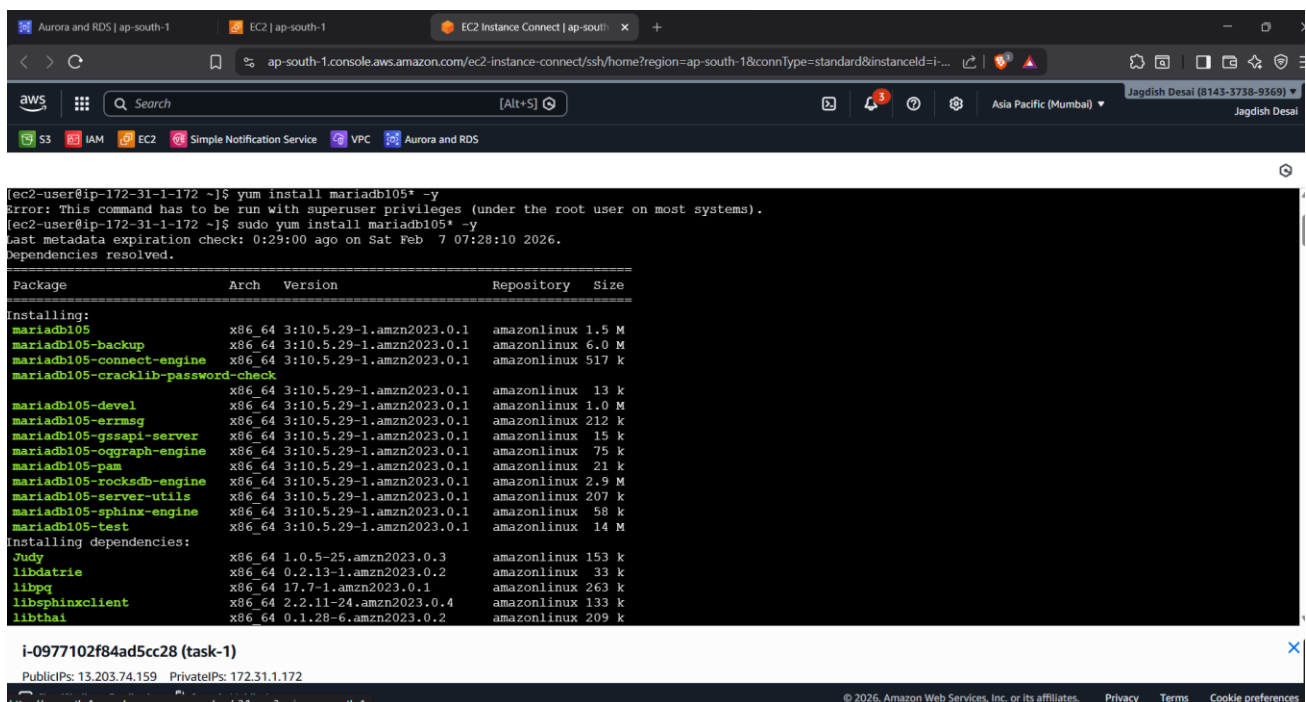


Allow port 3306 mysql port in security groups inbound rules.

After creation of database install mariadb tools on ec2 instance:

It installs MariaDB client tools on the EC2 instance

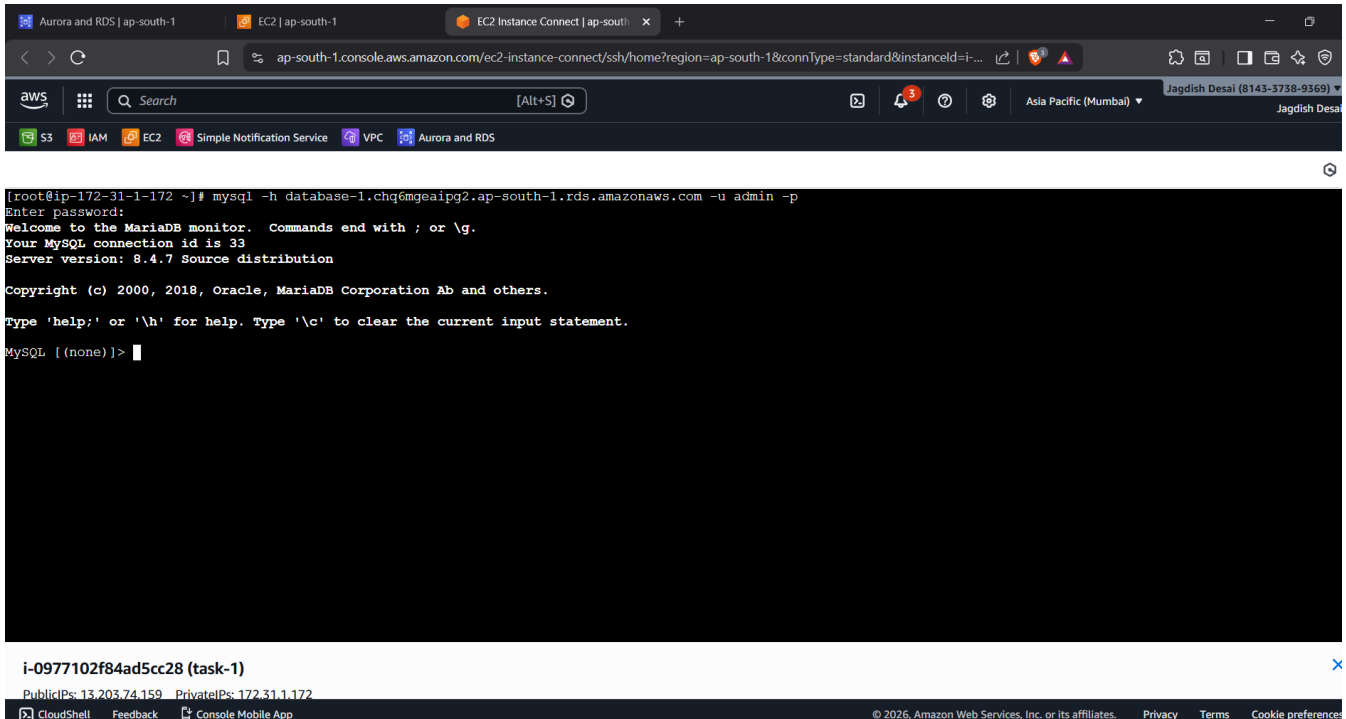
command: `yum install mariadb105* -y`



Then login to our database: We use this command to connect to a MySQL database that is running on AWS RDS from your EC2/server.

Command

`mysql -h database-1.chq6mgeaipg2.ap-south-1.rds.amazonaws.com -u admin -p`



The screenshot shows the AWS Management Console interface. At the top, there are tabs for 'Aurora and RDS | ap-south-1', 'EC2 | ap-south-1', and 'EC2 Instance Connect | ap-south-1'. The browser address bar shows the URL: `ap-south-1.console.aws.amazon.com/ec2-instance-connect/ssh/home?region=ap-south-1&connType=standard&instanceId=i-0977102f84ad5cc28`. The console header includes the AWS logo, a search bar, and navigation links for S3, IAM, EC2, Simple Notification Service, VPC, and Aurora and RDS. The user's name 'Jagdish Desai' and phone number '(8143-3738-9369)' are visible in the top right. The main content area is a terminal window titled 'i-0977102f84ad5cc28 (task-1)' showing a successful MySQL connection. The terminal output includes the command `mysql -h database-1.chq6mgeaipg2.ap-south-1.rds.amazonaws.com -u admin -p`, the password prompt, and the MySQL welcome message. The terminal prompt is `MySQL [(none)]>`.

```
[root@ip-172-31-1-172 ~]# mysql -h database-1.chq6mgeaipg2.ap-south-1.rds.amazonaws.com -u admin -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 33
Server version: 8.4.7 Source distribution

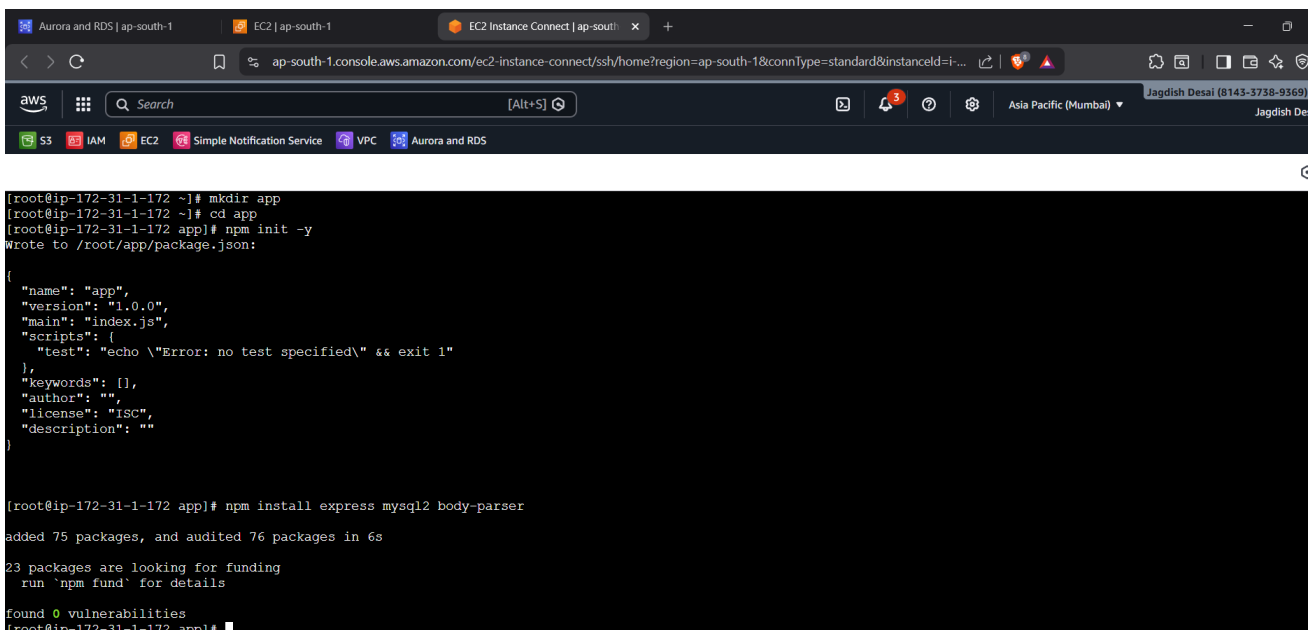
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]>
```

Exit;

Step 4 : Then starting creating our project by creating app directory.



The screenshot shows the AWS Management Console interface, similar to the previous one. The terminal window shows the following commands and output:

```
[root@ip-172-31-1-172 ~]# mkdir app
[root@ip-172-31-1-172 ~]# cd app
[root@ip-172-31-1-172 app]# npm init -y
Wrote to /root/app/package.json:

{
  "name": "app",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": ""
}

[root@ip-172-31-1-172 app]# npm install express mysql2 body-parser
added 75 packages, and audited 76 packages in 6s

23 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
[root@ip-172-31-1-172 app]#
```

Commands used:

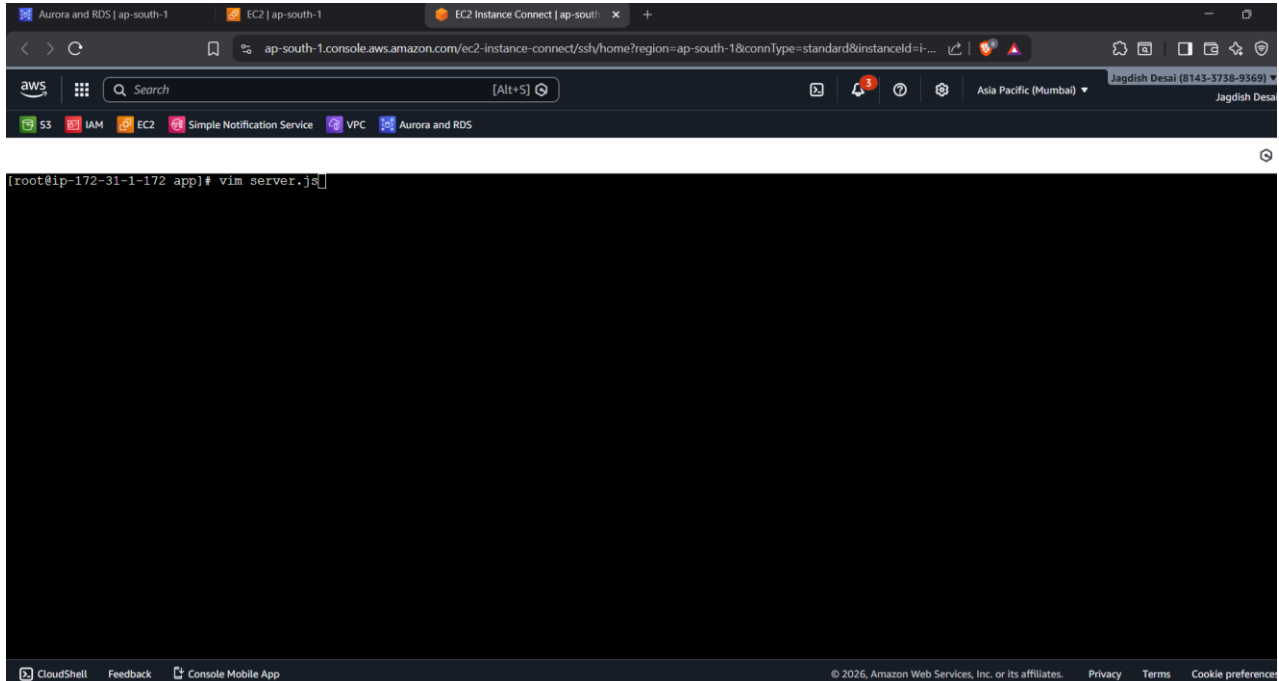
1. `Mkdir app`

2. `cd app`

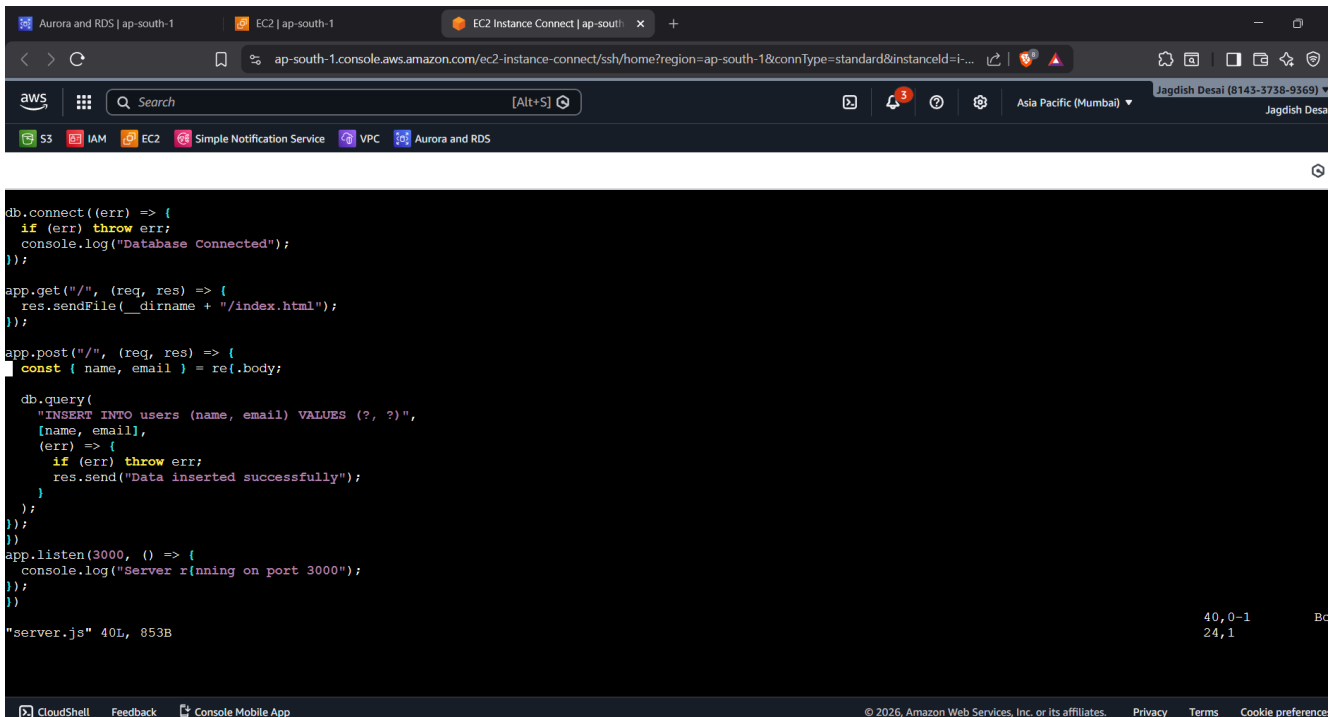
3. npm init -y initializes a Node project instantly.

It creates package.json with default values so you can start installing packages.

Creating backend server.js {nodejs} file : vim server.js



A screenshot of the AWS CloudShell interface. The terminal prompt is `[root@ip-172-31-1-172 app]#`. The user has entered the command `vim server.js`. The terminal is currently empty, showing only the command input.



A screenshot of the AWS CloudShell interface showing the contents of the `server.js` file. The code is as follows:

```
db.connect((err) => {
  if (err) throw err;
  console.log("Database Connected");
});

app.get("/", (req, res) => {
  res.sendFile(__dirname + "/index.html");
});

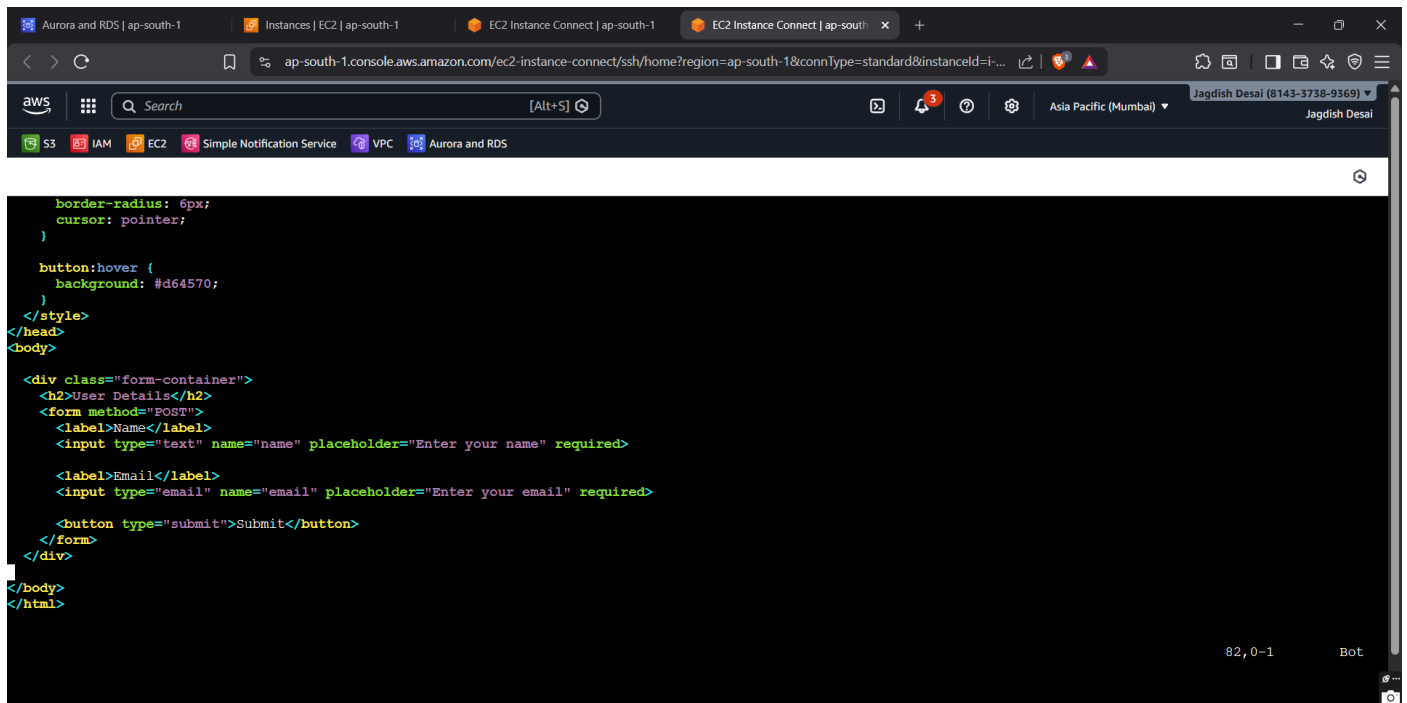
app.post("/", (req, res) => {
  const { name, email } = req.body;

  db.query(
    "INSERT INTO users (name, email) VALUES (?, ?)",
    [name, email],
    (err) => {
      if (err) throw err;
      res.send("Data inserted successfully");
    }
  );
});

app.listen(3000, () => {
  console.log("Server running on port 3000");
});
```

The terminal output at the bottom shows the file size: `"server.js" 40L, 853B`.

Step 5 : Creating index.html file for frontend

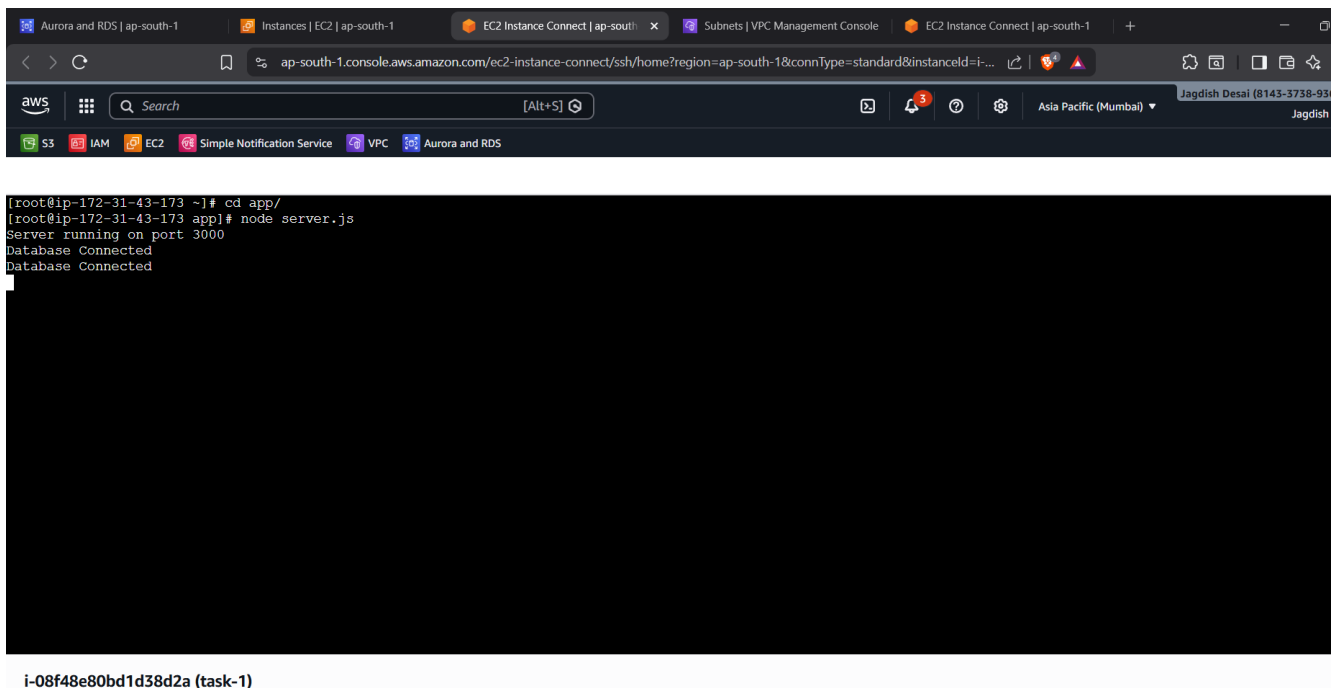


```
border-radius: 6px;
cursor: pointer;
}

button:hover {
  background: #d64570;
}
</style>
</head>
<body>

<div class="form-container">
  <h2>User Details</h2>
  <form method="POST">
    <label>Name</label>
    <input type="text" name="name" placeholder="Enter your name" required>
    <label>Email</label>
    <input type="email" name="email" placeholder="Enter your email" required>
    <button type="submit">Submit</button>
  </form>
</div>
</body>
</html>
```

Step 6: Then Run Application using node server.js command and checking that database connection is established or not:

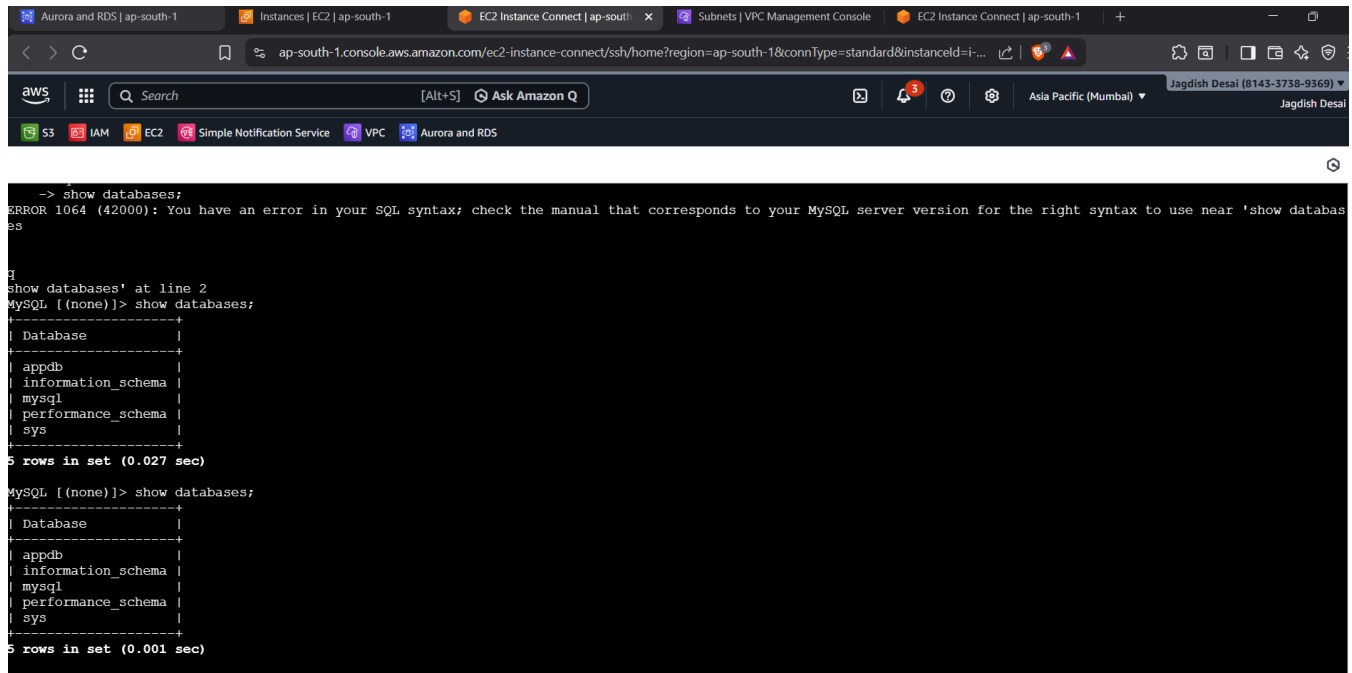


```
[root@ip-172-31-43-173 ~]# cd app/
[root@ip-172-31-43-173 app]# node server.js
Server running on port 3000
Database Connected
Database Connected
```

i-08f48e80bd1d38d2a (task-1)

Step 7: Going to to database on ec2 using endpoint and create appdb named database using command.

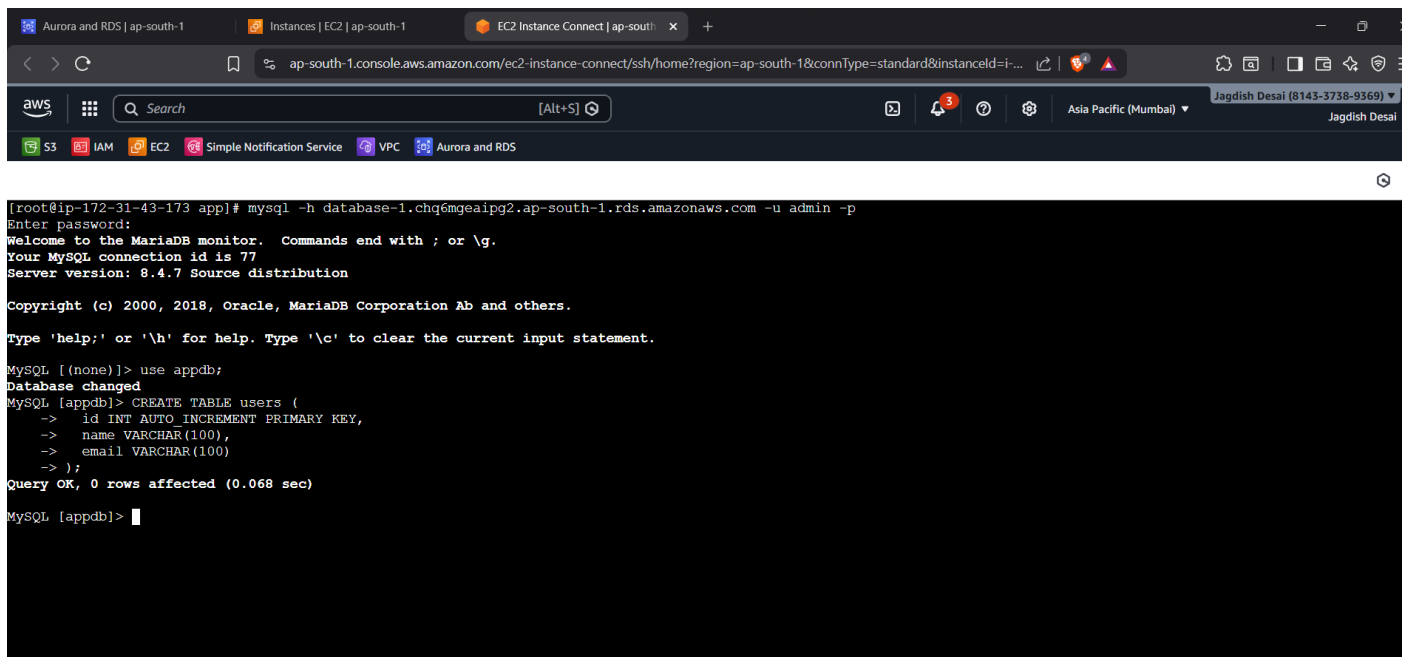
create database appdb;



```
-> show databases;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'show databases' at line 2
MySQL [(none)]> show databases;
+-----+
| Database |
+-----+
| appdb    |
| information_schema |
| mysql    |
| performance_schema |
| sys      |
+-----+
5 rows in set (0.027 sec)

MySQL [(none)]> show databases;
+-----+
| Database |
+-----+
| appdb    |
| information_schema |
| mysql    |
| performance_schema |
| sys      |
+-----+
5 rows in set (0.001 sec)
```

Step 8: After check That Using Command use appdb; for using database after that creating table users for storing data into that table.



```
[root@ip-172-31-43-173 app]# mysql -h database-1.chq6mgeaigp2.ap-south-1.rds.amazonaws.com -u admin -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 77
Server version: 8.4.7 Source distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

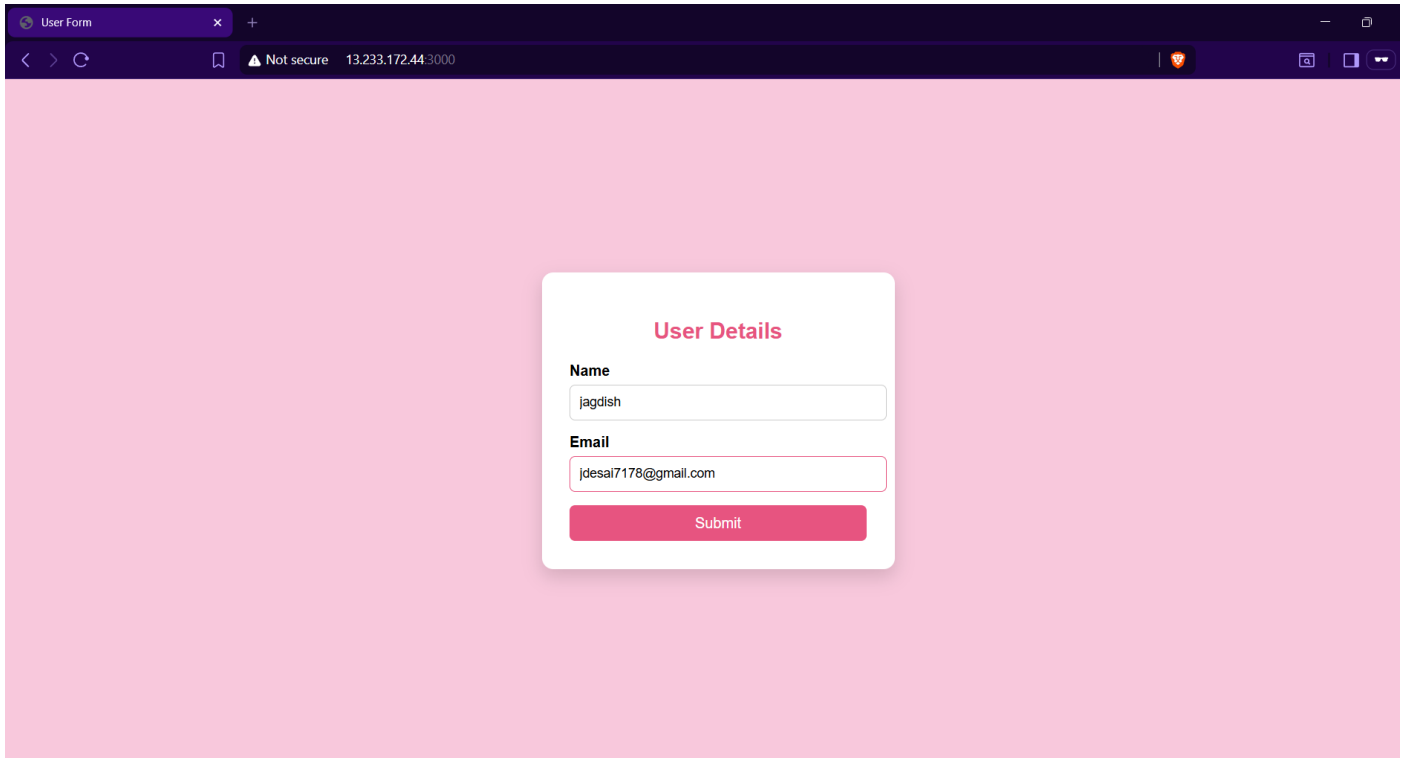
MySQL [(none)]> use appdb;
Database changed
MySQL [appdb]> CREATE TABLE users (
->   id INT AUTO_INCREMENT PRIMARY KEY,
->   name VARCHAR(100),
->   email VARCHAR(100)
-> );
Query OK, 0 rows affected (0.068 sec)

MySQL [appdb]>
```

Step 9: Open browser and use public ip of task-1 instance

`http://13.233.172.44:3000/`

Result:



The screenshot shows a web browser window with a single tab titled 'User Form'. The address bar displays 'Not secure' and the URL '13.233.172.44:3000'. The main content area has a light pink background. In the center, there is a white rounded rectangle containing the form. The form has a title 'User Details' in red. Below the title, there are two labels: 'Name' and 'Email'. The 'Name' field contains the text 'jagdish' and the 'Email' field contains 'jdesai7178@gmail.com'. At the bottom of the form is a red button labeled 'Submit'.

After Submitting Details form:



After Submitting form details it will shows data inserted successfully.

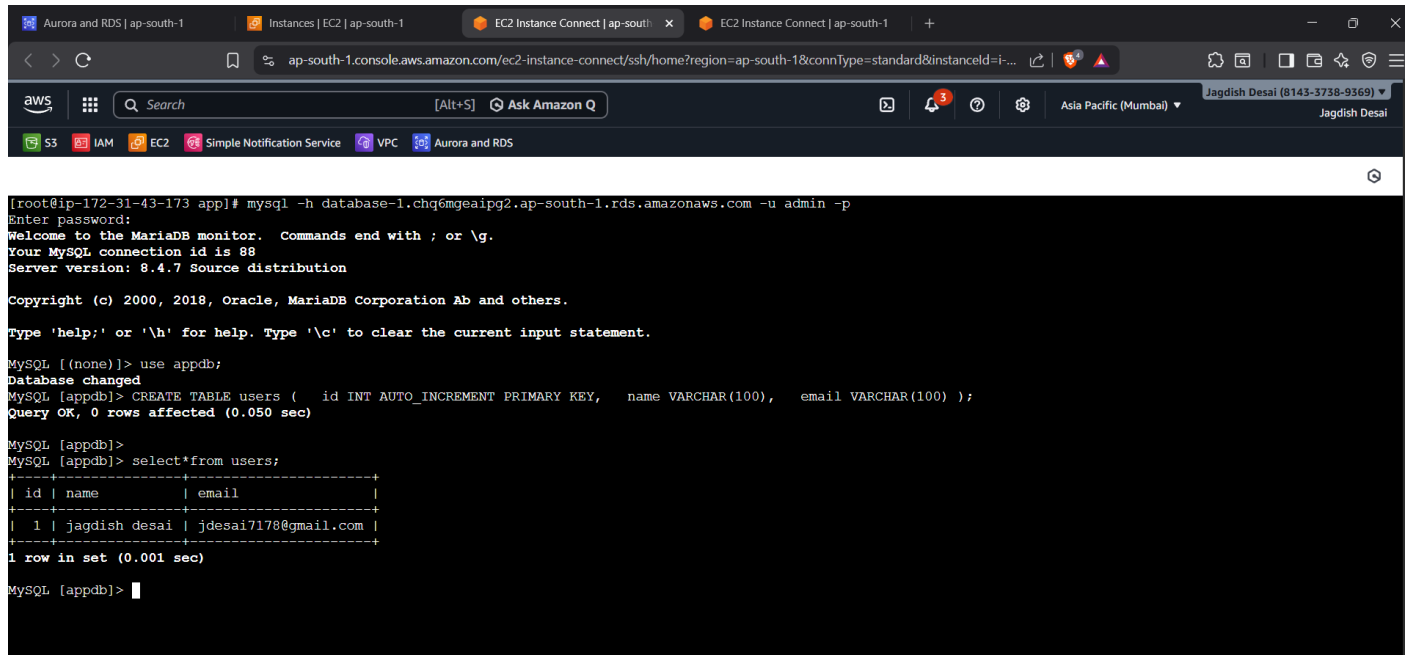
Means our data successfully stored in appdb database users table.

Step 10: After Inserting data go to your instance and again check mysql database which is named appdb. By using commands

1.use appdb;

2.select*from users; //this will show you table data which can be inserted by user

Step 10: appdb Database users data



```
[root@ip-172-31-43-173 app]# mysql -h database-1.chg6mgeaigp2.ap-south-1.rds.amazonaws.com -u admin -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 88
Server version: 8.4.7 Source distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

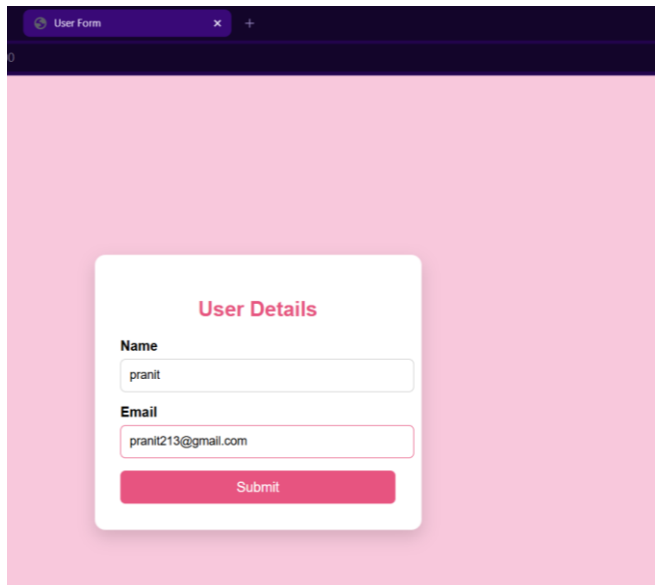
MySQL [(none)]> use appdb;
Database changed
MySQL [appdb]> CREATE TABLE users ( id INT AUTO_INCREMENT PRIMARY KEY, name VARCHAR(100), email VARCHAR(100) );
Query OK, 0 rows affected (0.050 sec)

MySQL [appdb]>
MySQL [appdb]> select*from users;
+----+-----+-----+
| id | name  | email |
+----+-----+-----+
| 1  | jagdish desai | jdesai7178@gmail.com |
+----+-----+-----+
1 row in set (0.001 sec)

MySQL [appdb]>
```

You will see the data successfully stored in appdb database users table.

After entering some another users details:



The screenshot shows a web browser window with a tab titled 'User Form'. The page has a light pink background. In the center, there is a white rounded rectangle containing the form. The form has a title 'User Details' in red. Below the title, there are two input fields: 'Name' with the value 'pranit' and 'Email' with the value 'pranit213@gmail.com'. At the bottom of the form is a red 'Submit' button.

Database Automatically Updated :

```
MySQL [appdb]> select*from users;
+-----+
| id | name          | email                      |
+-----+
| 1  | jagdish desai | jdesai7178@gmail.com     |
+-----+
1 row in set (0.001 sec)

MySQL [appdb]> select*from users;
+-----+
| id | name          | email                      |
+-----+
| 1  | jagdish desai | jdesai7178@gmail.com     |
| 2  | pranit        | pranit213@gmail.com      |
+-----+
2 rows in set (0.001 sec)
```

Project Workflow

Node.js + Express + MySQL (AWS RDS)

1. User Access

The user opens the web application using the public IP address of the EC2 instance in a browser.

2. Frontend Interaction

A simple HTML form is displayed on the browser.
The user enters their **name** and **email** and clicks the submit button.

3. Request Sent to Server

When the form is submitted, the browser sends the user data to the backend server using an HTTP POST request.

4. Backend Processing

The Node.js application running on the EC2 instance receives the request.
Express.js processes the request and extracts the user details from the form.

5. Database Connection

The backend application connects to the MySQL database hosted on AWS RDS using the RDS endpoint and credentials.

6. Data Storage

The received name and email are inserted into the **users** table in the MySQL RDS database.

7. Response to User

After successful insertion, the server sends a confirmation message back to the browser.

8. Continuous Service

The server keeps running and can handle multiple user requests without stopping.

Overall Data Flow

User Browser → EC2 (Node.js + Express) → MySQL RDS → Response to User

Code Used In Task:

Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>User Form</title>
  <style>
    body {
      margin: 0;
      padding: 0;
      height: 100vh;
      display: flex;
      justify-content: center;
      align-items: center;
      background: #f8c8dc; /* ice pink background */
      font-family: Arial, sans-serif;
    }

    .form-container {
      background: white;
      padding: 30px;
```

```
border-radius: 12px;  
width: 320px;  
box-shadow: 0 8px 20px rgba(0, 0, 0, 0.15);  
}
```

```
.form-container h2 {  
  text-align: center;  
  margin-bottom: 20px;  
  color: #e75480;  
}
```

```
label {  
  font-weight: bold;  
  display: block;  
  margin-bottom: 5px;  
}
```

```
input {  
  width: 100%;  
  padding: 10px;  
  margin-bottom: 15px;  
  border-radius: 6px;  
  border: 1px solid #ccc;  
  font-size: 14px;  
}
```

```
input:focus {  
  outline: none;  
  border-color: #e75480;  
}
```

```
button {
  width: 100%;
  padding: 10px;
  background: #e75480;
  border: none;
  color: white;
  font-size: 16px;
  border-radius: 6px;
  cursor: pointer;
}
button:hover {
  background: #d64570;
}
</style>
</head>
<body>
<div class="form-container">
  <h2>User Details</h2>
  <form method="POST">
    <label>Name</label>
    <input type="text" name="name" placeholder="Enter your name" required>

    <label>Email</label>
    <input type="email" name="email" placeholder="Enter your email" required>
    <button type="submit">Submit</button>
  </form>
</div>

</body>
</html>
```

Server.js

```
const express = require("express");
const mysql = require("mysql2");
const bodyParser = require("body-parser");
const app = express();
app.use(bodyParser.urlencoded({ extended: true }));
const db = mysql.createConnection({
  host: "database-1.chq6mgeaipg2.ap-south-1.rds.amazonaws.com",
  user: "admin",
  password: "shankarrao71",
  database: "appdb"
});
db.connect(err => {
  if (err) throw err;
  console.log("Database Connected");
});
app.post("/", (req, res) => {
  const { name, email } = req.body;
  db.query(
    "INSERT INTO users (name, email) VALUES (?, ?)",
    [name, email],
    () => res.send("Data inserted successfully")
  );
});
app.get("/", (req, res) => {
  res.sendFile(__dirname + "/index.html");
});
```

```

app.listen(3000, () => {
  console.log("Server running on port 3000");
});

db.connect((err) => {
  if (err) {
    console.error("DB connection failed:", err.message);
    process.exit(1);
  }
  console.log("Database Connected");
});

```

mysql commands I have used:

SQL Command	Description
CREATE DATABASE appdb;	Creates a new database for the project
USE appdb;	Selects the database to work in
CREATE TABLE users (id INT AUTO_INCREMENT PRIMARY KEY, name VARCHAR(100), email VARCHAR(100));	Creates a table to store user details
SELECT * FROM users;	Displays all records from the users table
SHOW TABLES;	Shows all tables in the database
EXIT;	Exits the MySQL database session