Simple Student - Employer Connect Application using AWS

EC2 (Elastic Compute Cloud)

- **Purpose**
  - It provides scalable computing capacity and is designed to make web-scale cloud computing easier for developers.
- **Key-features**
  - Option for virtual computing environments with different OS, known as instances. Multiple configurations for CPU, memory, network and storage
  - Secured logins to instance using key value pair(.pem files or private key file)
  - Scaling option for users to scale up or down the resources as needed.
- **Benefits**
  - Elasticity and scalability, allowing user to scale up or down the resource with option of autoscaling.
  - Pay as you go prices for the resources used.

S3 (Simple Storage Service)

- **Purpose**
  - It is used to store and retrieve any type of data of varying sizes from anywhere on the web.
- **Key-features**
  - Reliable and scalable data storage.
  - Storing in buckets with unique names that are easy to identify.
  - Lifecycle policies and access controls.
- **Benefits**
  - Availability of data anytime and anywhere.
  - secured storage of data with access control mechanisms.
  - Easy to use UI.

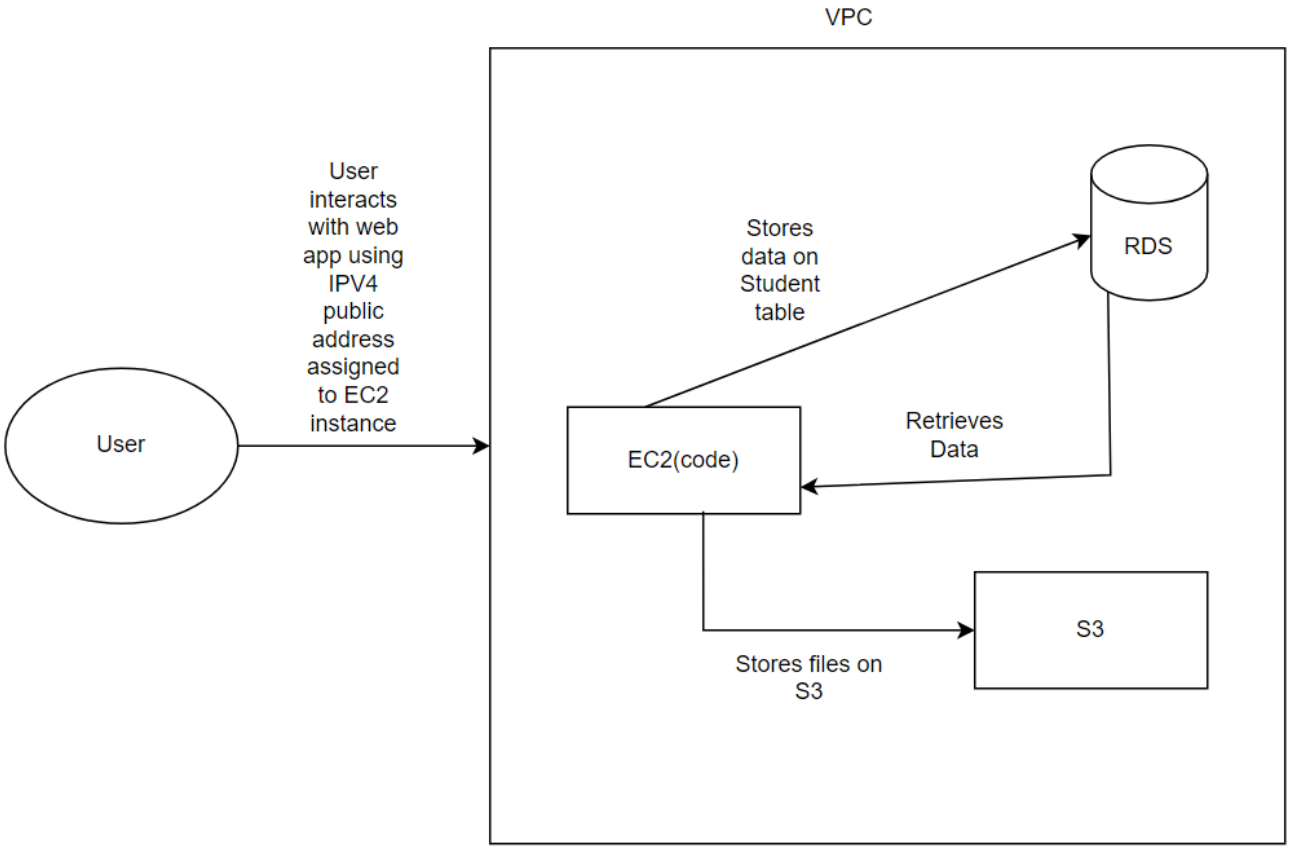
RDS (Relational Database Service)

- **Purpose**
  - It is used to setup, operate, manage various Relational Database management sytems
- **Key-features**
  - Support for multiple database instances
  - High availability of data
  - Automated backups
- **Benefits**
  - Easy to setup a database
  - Autoscaling the database size
  - Scalable performance and storage

CloudFormation

- **Purpose**
  - It provides a way to model and set up AWS resources.
- **Key-features**
  - Using YAML or JSON file to provision resources
  - Managing all AWS resources in one place
  - Version control and replication of AWS infrastructure
- **Benefits**
  - User can spend less amount of time worrying about applications.
  - Streamlines the process of deploying and managing AWS resources.
  - Reduces the risk of manual errors.

Architecture design:



Configuration details:

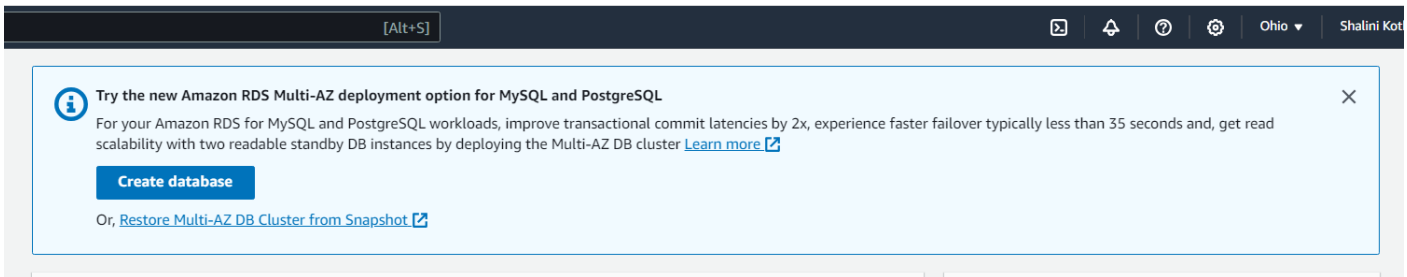
Steps:

Step 1: Creating RDS database instance.
Step 2: launching EC2 instance.
Step 3: Setting up S3 bucket.
Step 4: ssh to EC2 instance and connect to MYSQL client to create table
Step 5: Deploying code to EC2 instance and testing the application functionality
Step 6: Using Cloud Formation to create VPC.

I have explained the details of each step.

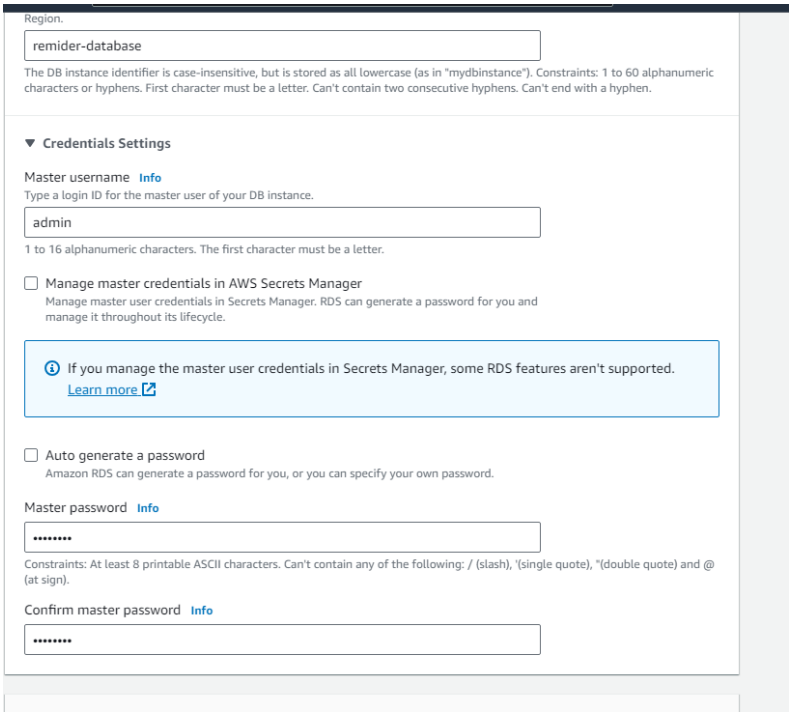
Step 1: Database Setup on AWS RDS

First, login to AWS management console and search for RDS. Once you open RDS services click on create database.

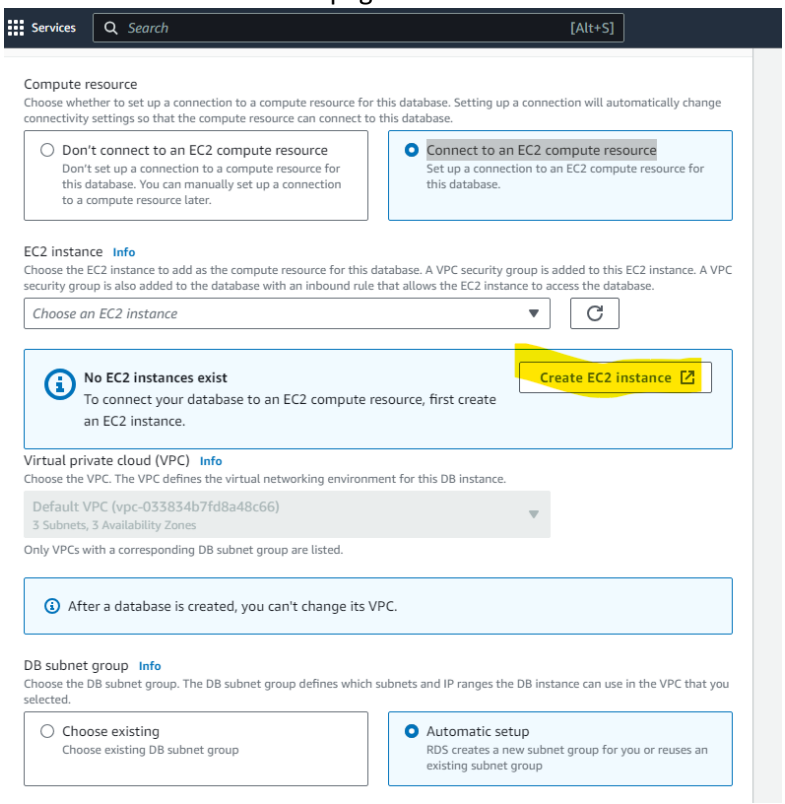


Enter details as below:

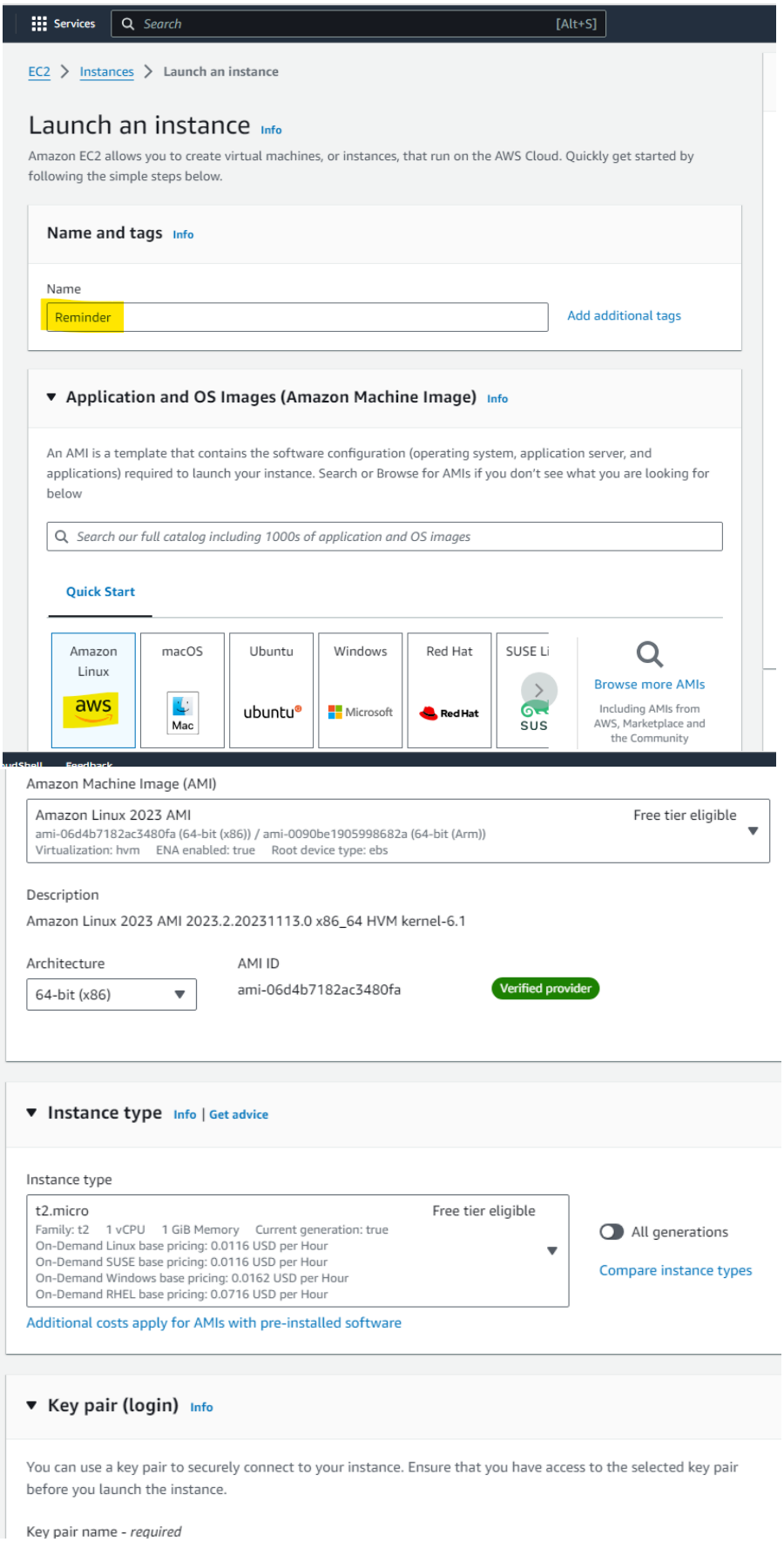
- Choose standard create in Choose a database creation method and engine options as MYSQL.
- Select engine version and template as free tier. In settings, give name for database instance. Enter master username and password. Later I have **changed my database name to student** as I have changed my application to student. The initial screenshots have reminder but later it was changed to student. I have changed the endpoints in web application code as well.



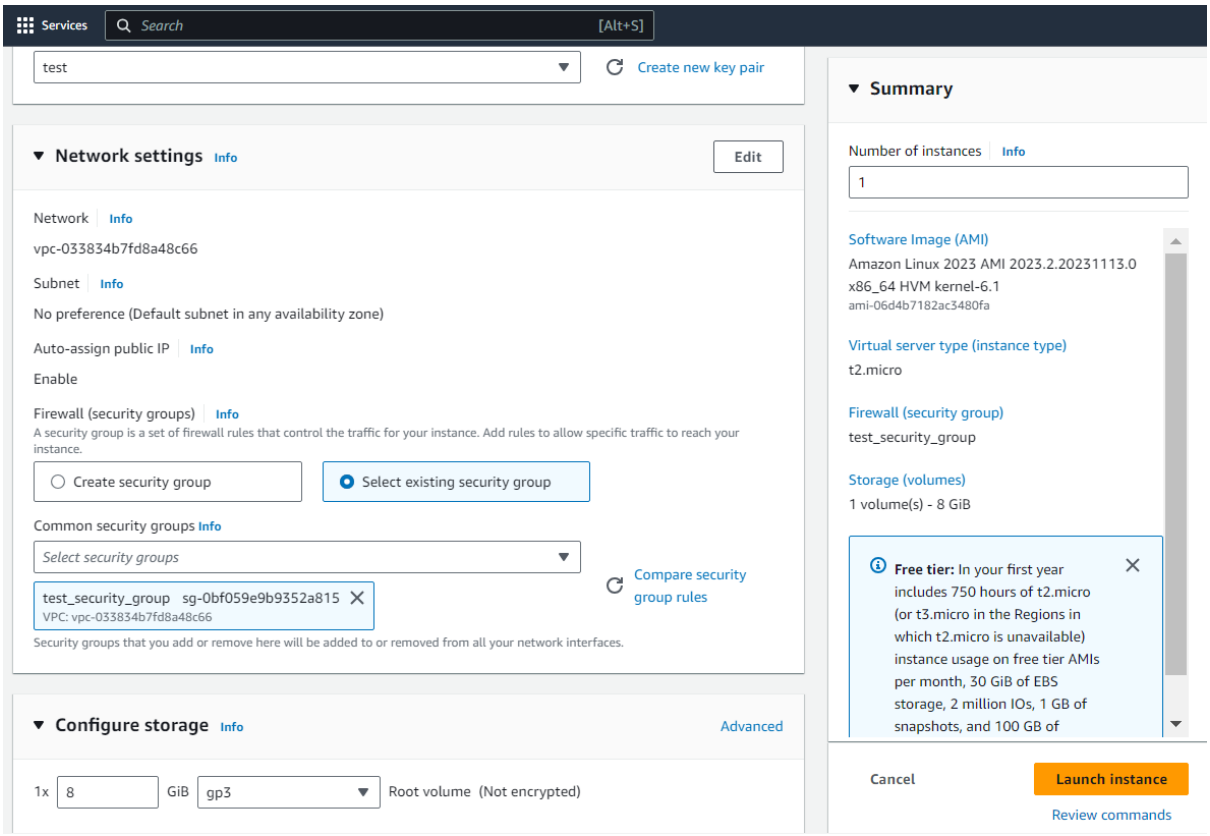
Under connectivity, select Connect to an EC2 compute resource and click on create EC2 instance as there is no EC2 instance at present. It will redirect to launch instance page in EC2 service.



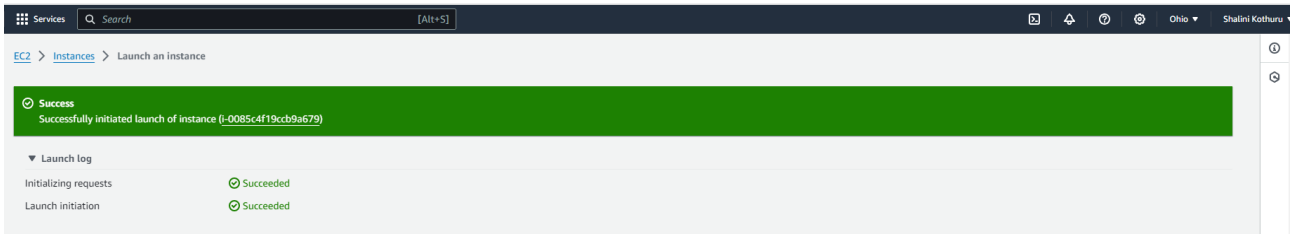
Step 2: Launching EC2 instance (we will be back to step 1 after creating EC2 instance)



I am using the key value pair that was created last time for this EC2 instance as well.

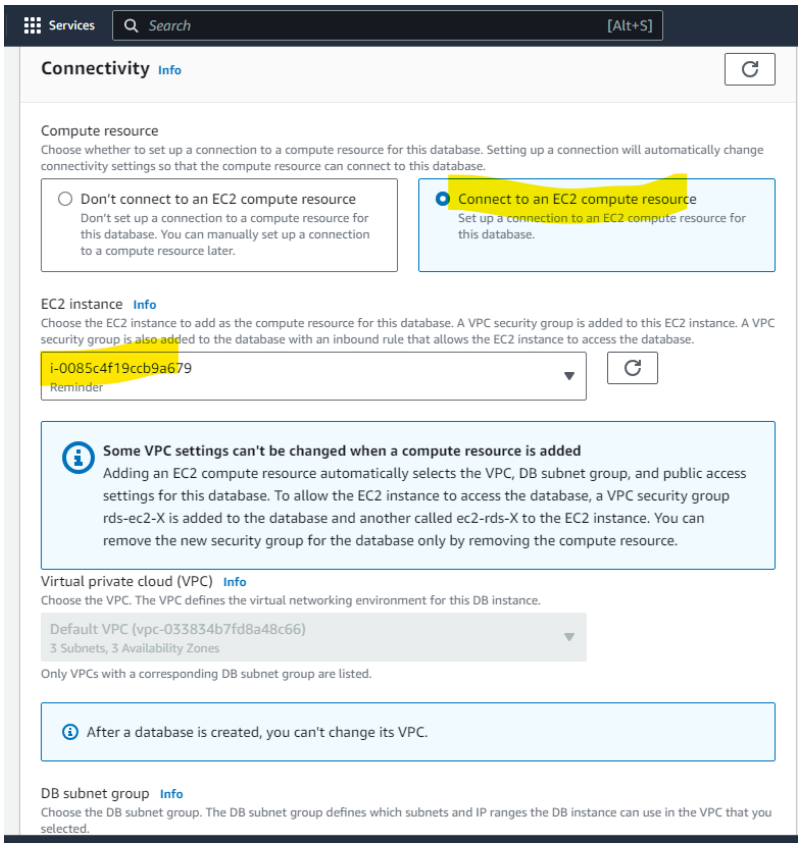


I used existing security group and key value pair. Now click on launch instance. Its launched successfully.

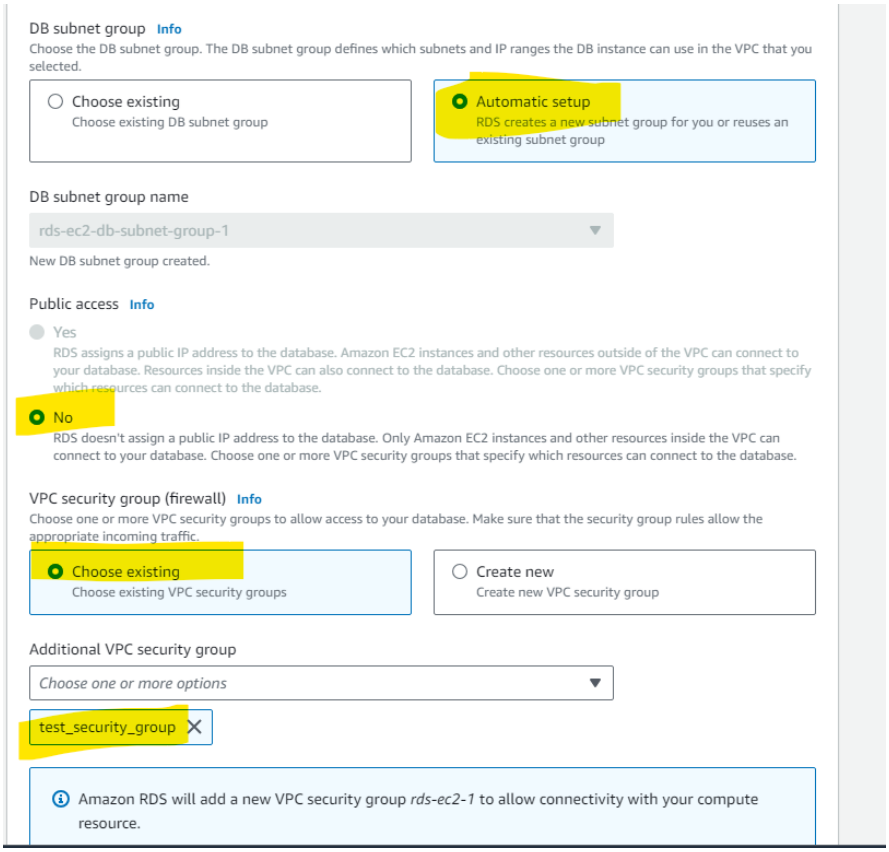


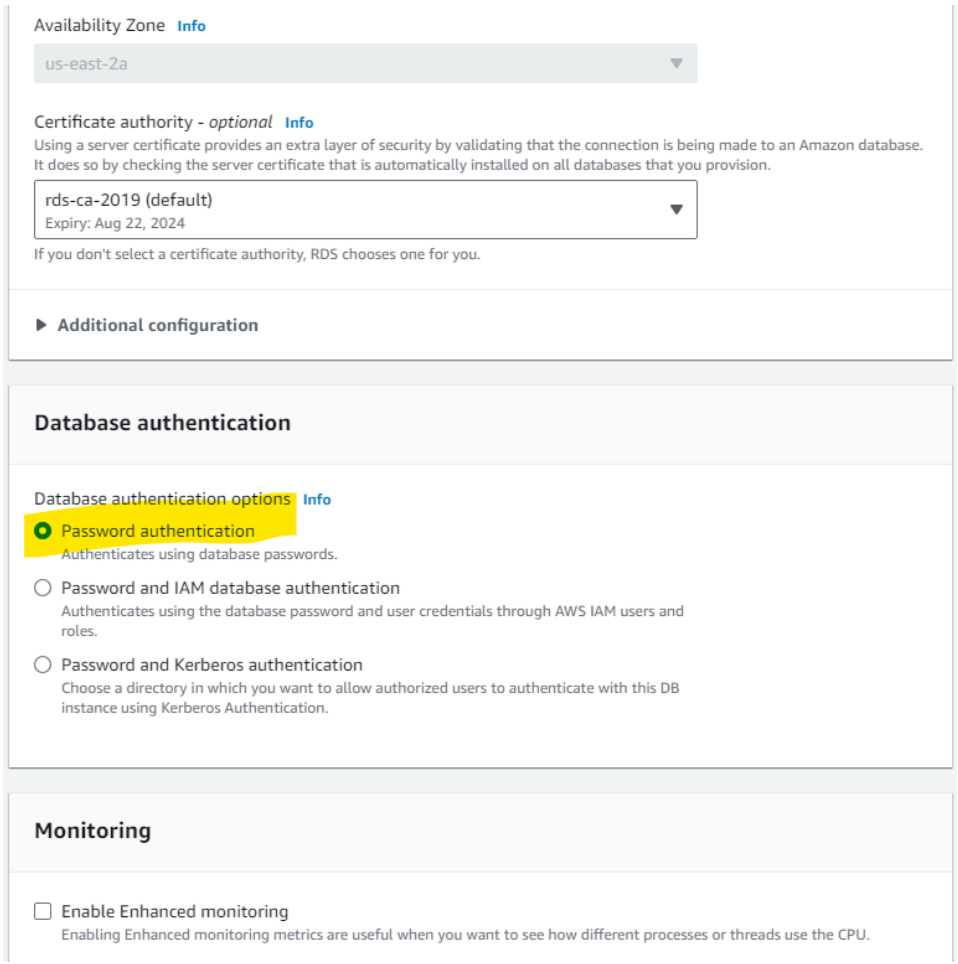
Back to Step 1: Creating RDS database instance.

Select the EC2 instance that was created recently.

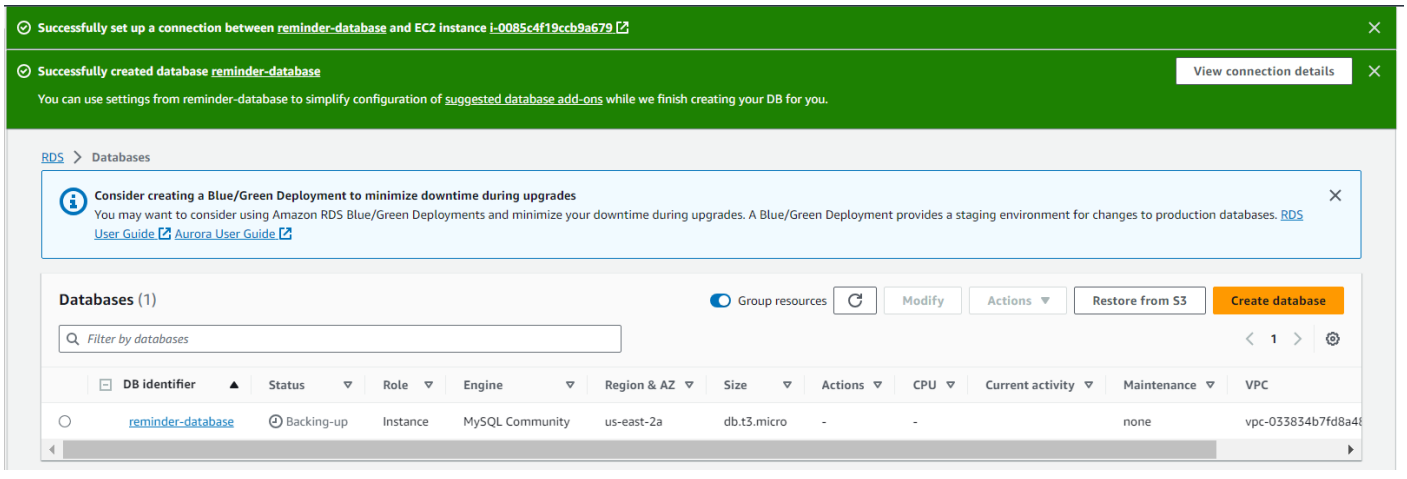


I have change public access to yes later while working on deploying website.

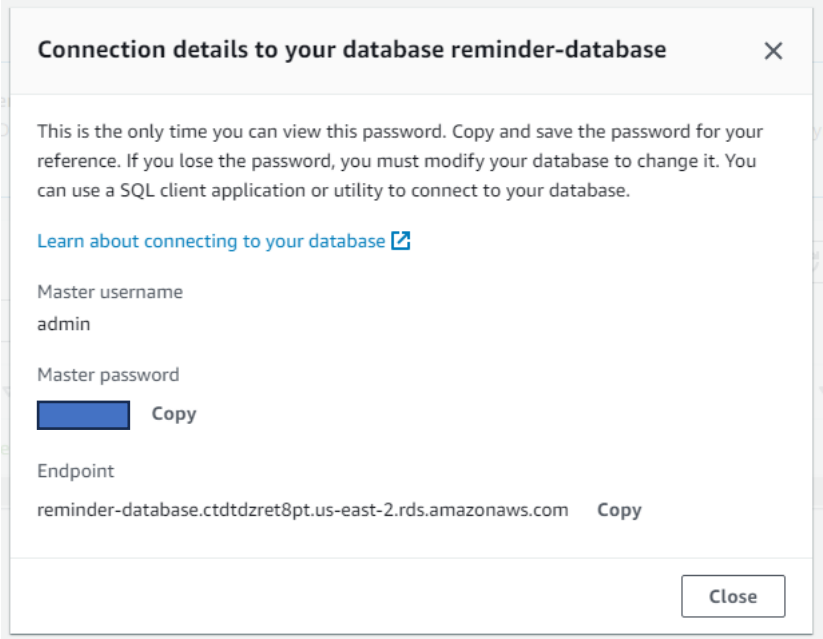




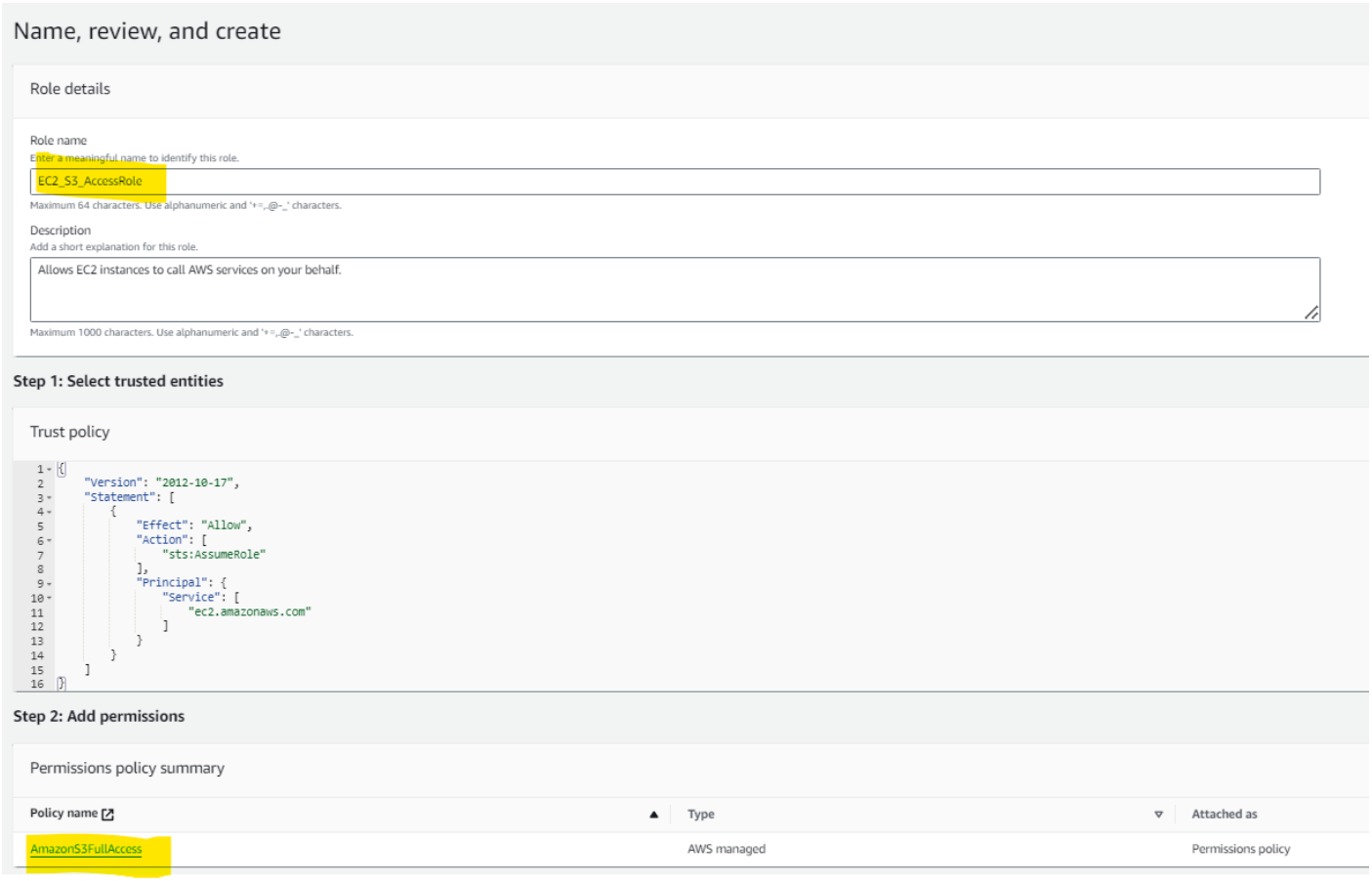
After selecting options as above click on create database at the bottom right corner of current webpage which creates RDS database instance.



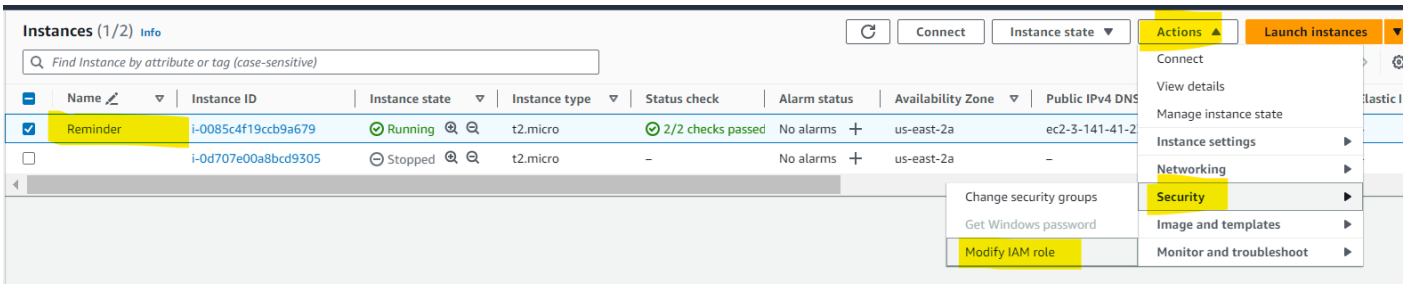
My database connection details are as below.
Endpoint: reminder-database.ctdtdzret8pt.us-east-2.rds.amazonaws.com.
Later It was changed to student.ctdtdzret8pt.us-east-2.rds.amazonaws.com as I changed my database name.



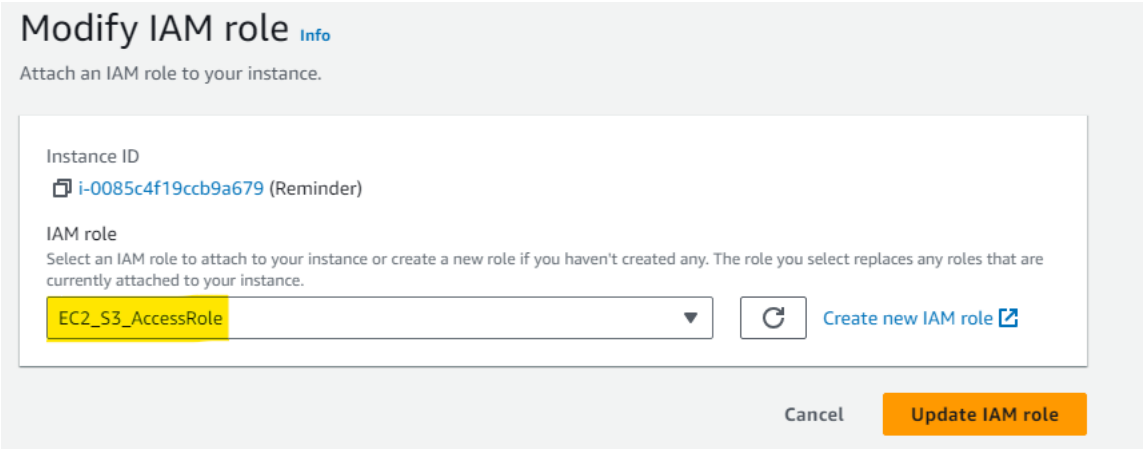
Now I am creating a new IAM role for my EC2 instance. Go to role in AWS management console and create role.
I created new IAM role 'EC2_S3_AccessRole' as below



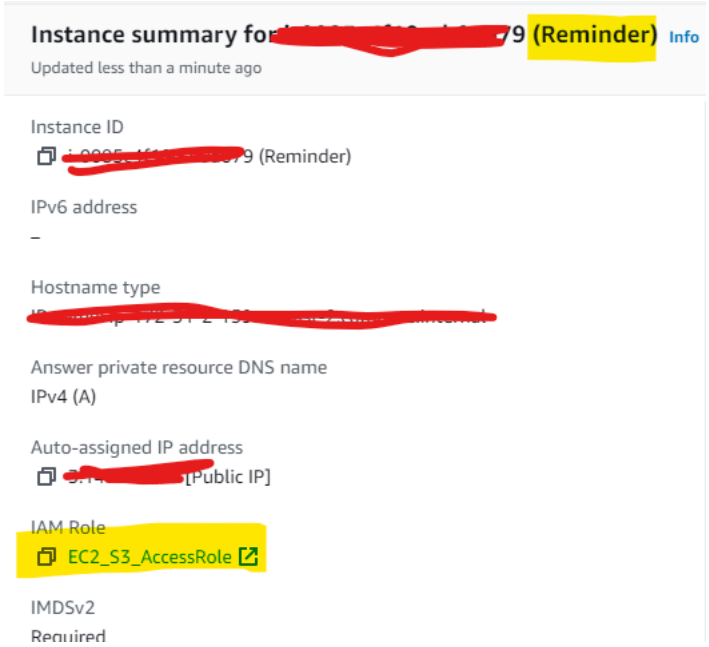
Now, lets attach role to EC2 instance. Go select EC2 instance and click on Action tab> security> Modify IAM Role.



Click on modify IAM role and selecting the role that was created for EC2 and updating the IAM role.



Now, this role is attached to EC2 instance Reminder



Step 3: Setting up S3 bucket

I am using already existing s3 bucket s3bucketshalini which was created as part of last AWS assignment. But I have to update permissions. Select the S3 bucket and under permissions go to bucket policy and update policy as below and save changes. This provides the necessary permissions for my application to read from and write to the S3 bucket.

Bucket ARN

arn:aws:s3:::s3bucketshalini

Policy

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::274082713924:role/EC2_S3_AccessRole"
            },
            "Action": [
                "s3:GetObject",
                "s3:PutObject",
                "s3:DeleteObject"
            ],
            "Resource": "arn:aws:s3:::s3bucketshalini/*"
        }
    ]
}
```

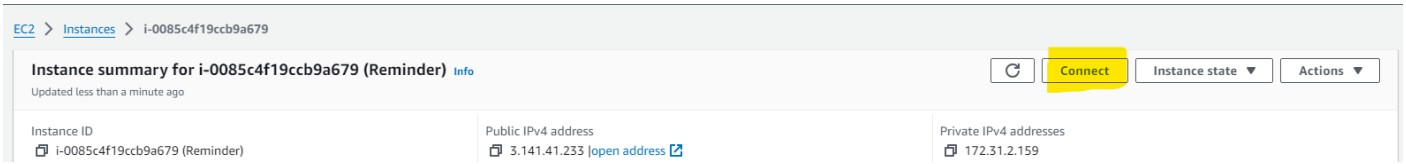
Step 4: SSH to EC2 instance and connect to MYSQL client to create table

Now, connect to EC2 instance using putty and .pem file that was downloaded when creating key-value pair.

I have installed pip3 and flask and git on EC2 instance.

```
[ec2-user@ip-172-31-2-159 ~]$ curl -O https://bootstrap.pypa.io/get-pip.py
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 2570k  100 2570k    0     0  16.8M      0 --:--:-- --:--:-- --:--:-- 16.9M
[ec2-user@ip-172-31-2-159 ~]$ sudo python3 get-pip.py
Collecting pip
  Downloading pip-23.3.1-py3-none-any.whl.metadata (3.5 kB)
Collecting wheel
  Downloading wheel-0.42.0-py3-none-any.whl.metadata (2.2 kB)
Downloading pip-23.3.1-py3-none-any.whl (2.1 MB)
   ---------------------------------------- 2.1/2.1 MB 21.8 MB/s eta 0:00:00
Downloading wheel-0.42.0-py3-none-any.whl (65 kB)
   ---------------------------------------- 65.4/65.4 kB 9.5 MB/s eta 0:00:00
Installing collected packages: wheel, pip
Successfully installed pip-23.3.1 wheel-0.42.0
WARNING: Running pip as the 'root' user can result in broken permissions and con
flicting behaviour with the system package manager. It is recommended to use a v
irtual environment instead: https://pip.pypa.io/warnings/venv
[ec2-user@ip-172-31-2-159 ~]$ pip3 --version
pip 23.3.1 from /usr/local/lib/python3.9/site-packages/pip (python 3.9)
[ec2-user@ip-172-31-2-159 ~]$ python3 --version
Python 3.9.16
[ec2-user@ip-172-31-2-159 ~]$ pip3 install flask
Defaulting to user installation because normal site-packages is not writeable
Collecting flask
  Downloading flask-3.0.0-py3-none-any.whl.metadata (3.6 kB)
Collecting Werkzeug>=3.0.0 (from flask)
  Downloading werkzeug-3.0.1-py3-none-any.whl.metadata (4.1 kB)
Collecting Jinja2>=3.1.2 (from flask)
  Downloading Jinja2-3.1.2-py3-none-any.whl (133 kB)
     ---------------------------------------- 133.1/133.1 kB 3.3 MB/s eta 0:00:00
Collecting itsdangerous>=2.1.2 (from flask)
  Downloading itsdangerous-2.1.2-py3-none-any.whl (15 kB)
Collecting click>=8.1.3 (from flask)
  Downloading click-8.1.7-py3-none-any.whl.metadata (3.0 kB)
Collecting blinker>=1.6.2 (from flask)
  Downloading blinker-1.7.0-py3-none-any.whl.metadata (1.9 kB)
Collecting importlib-metadata>=3.6.0 (from flask)
  Downloading importlib_metadata-7.0.0-py3-none-any.whl.metadata (4.9 kB)
Collecting zipp>=0.5 (from importlib-metadata>=3.6.0->flask)
  Downloading zipp-3.17.0-py3-none-any.whl.metadata (3.7 kB)
Collecting MarkupSafe>=2.0 (from Jinja2>=3.1.2->flask)
  Downloading MarkupSafe-2.1.3-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86
```

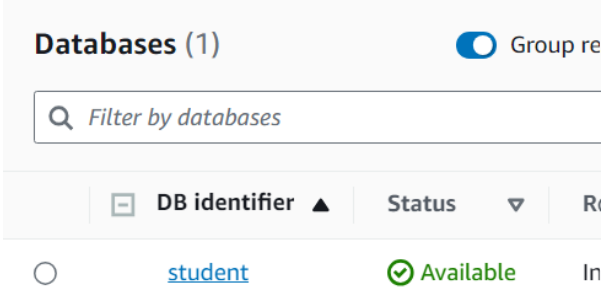
Another way connect to EC2 instance using connect button on the details page of EC2 instance. Both ways works using putty and using connect in ec2 instance.



I have uploaded my web app files to github. Now I am going to use that repository to clone files on EC2 instance.

```
[ec2-user@ip-172-31-2-159 ~]$ git clone https://github.iu.edu/skothuru/bda_simple_web_app.git
Cloning into 'bda_simple_web_app'...
Username for 'https://github.iu.edu': skothuru
Password for 'https://skothuru@github.iu.edu':
remote: Enumerating objects: 11, done.
remote: Counting objects: 100% (11/11), done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 11 (delta 0), reused 8 (delta 0), pack-reused 0
Receiving objects: 100% (11/11), done.
[ec2-user@ip-172-31-2-159 ~]$ cd bda_simple_web_app/
```

I renamed by database instance from reminder-database to student since I changed my application.



I have installed MySQL command line client using following command ‘sudo dnf update -y’ and I have connected to database instance using command. Next step is to create a database student

```
[ec2-user@ip-172-31-2-159 ~]$ mysql -h student.ctdtdzret8pt.us-east-2.rds.amazonaws.com -P 3306 -u admin -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 97
Server version: 8.0.33 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)]> create database student;
Query OK, 1 row affected (0.017 sec)

MySQL [(none)]> use student
Database changed
```

I have created table student in database student as show in below picture.

```
mysql> create table student(
    -> email varchar(255) primary key,
    -> first_name varchar(255),last_name varchar(255),mobile varchar(10),college varchar(255),major varchar(255),graduation_year YEAR,skills TEXT,
    -> area_of_interest TEXT);
Query OK, 0 rows affected (0.03 sec)

mysql> show tables
    -> ;
+------------------+
| Tables_in_student |
+------------------+
| student          |
+------------------+
1 row in set (0.00 sec)
```

Step 5: Deploying code to EC2 instance and testing the application functionality

Next step is deploying the code in EC2 instance. I have developed code locally and pushed that code to GitHub and later cloned it on EC2 instance.

Testing functionality of my web application:

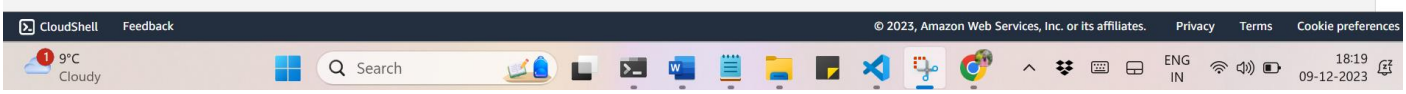
I have provided screenshots with timestamps for reference

Initially I don’t have any records in my table student and no files in S3 bucket s3bucketshalini. Please refer screenshot with timestamp

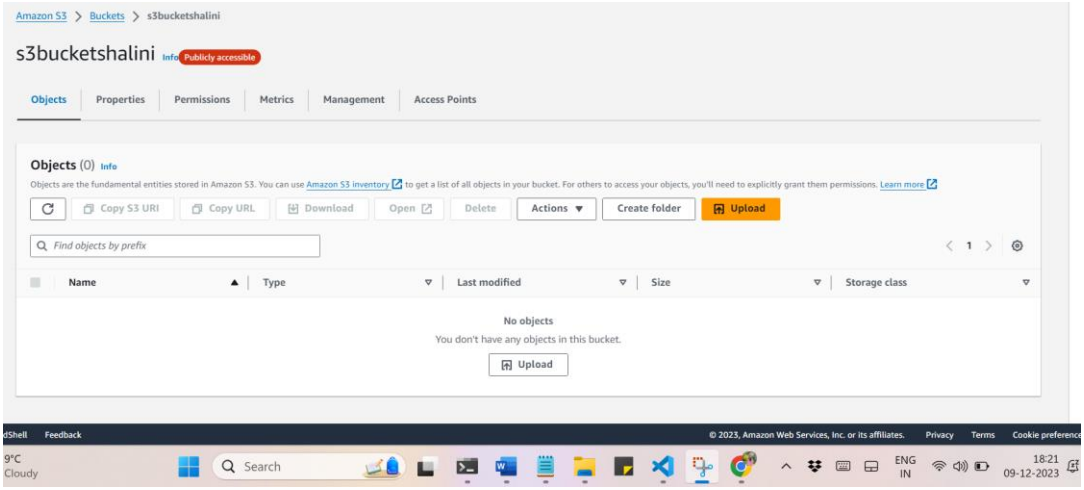
Tables with 0 rows

```
mysql> select * from student;
Empty set (0.00 sec)

mysql>
```

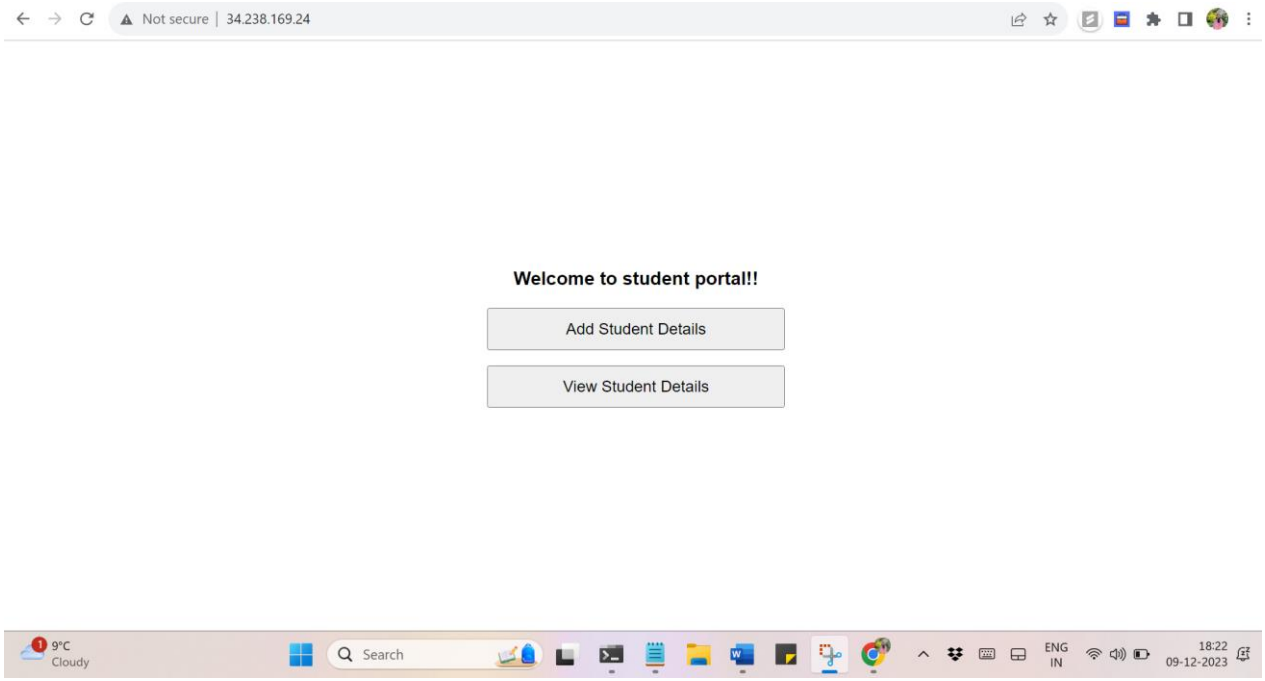



S3 bucket with no files

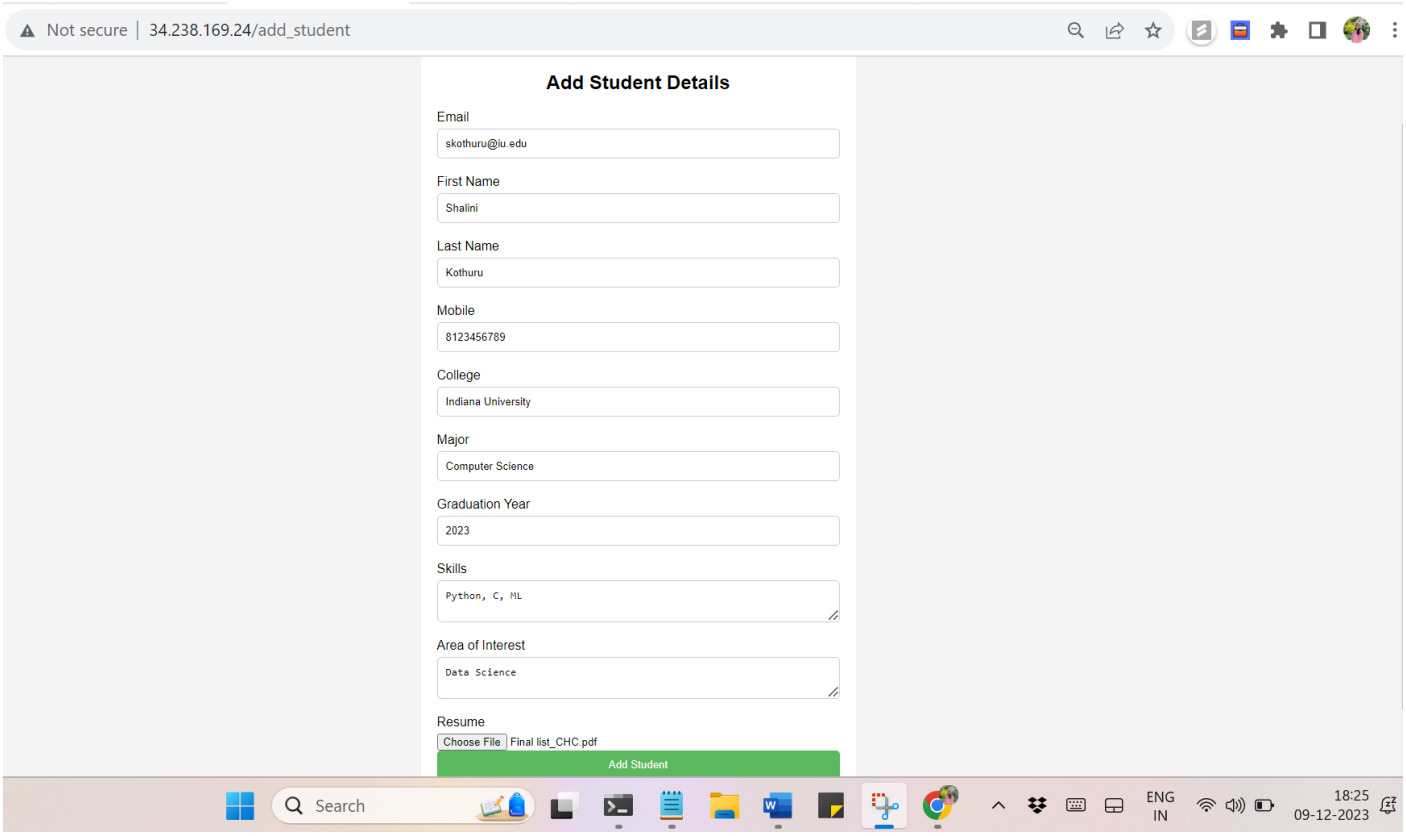


This is my public IP address of EC2 instance. I am going to connect to using http://34.238.169.24/ when I run my app.py file using python3 app.py

Below screen is my main portal

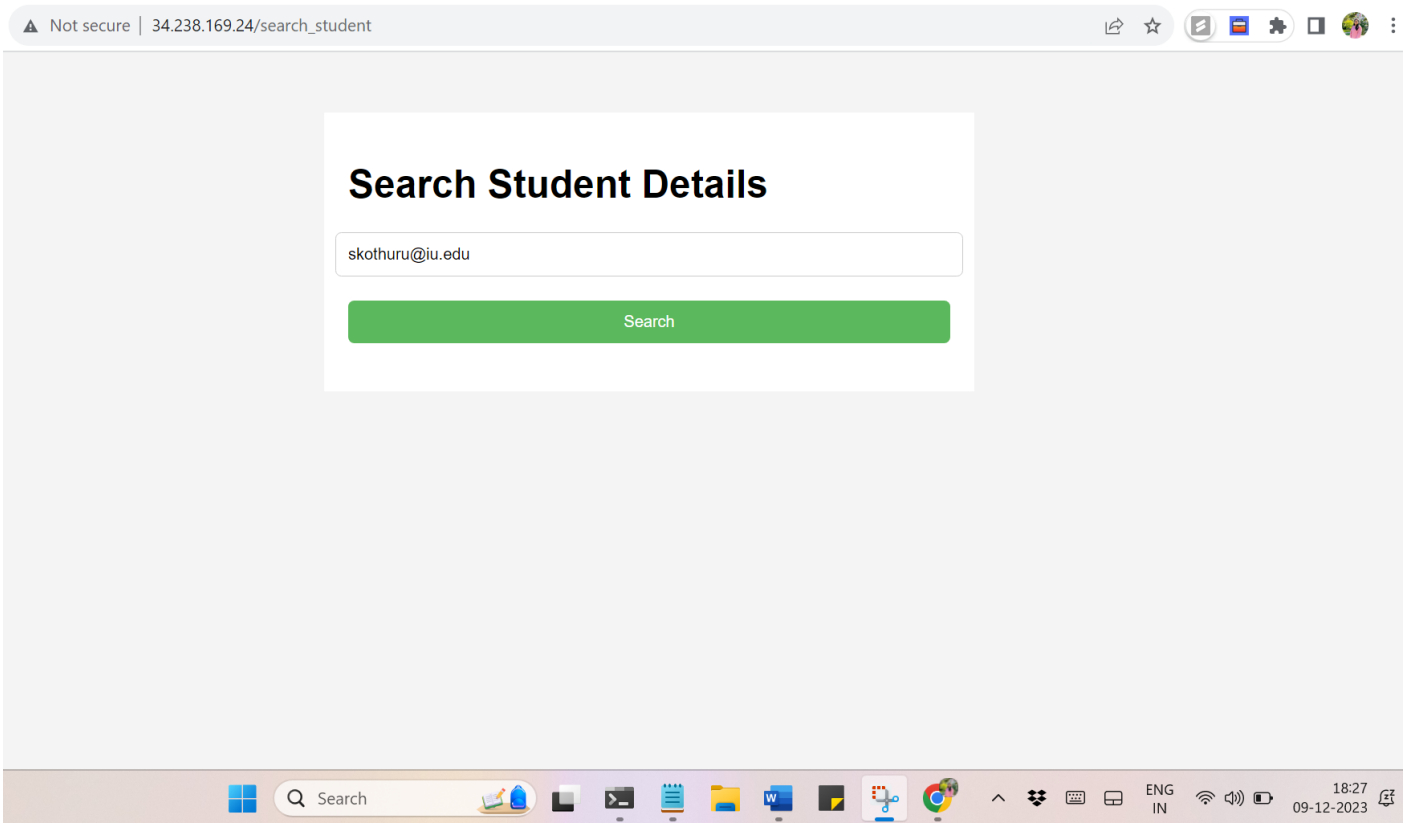


On clicking add student details and user get a chance to enter student details

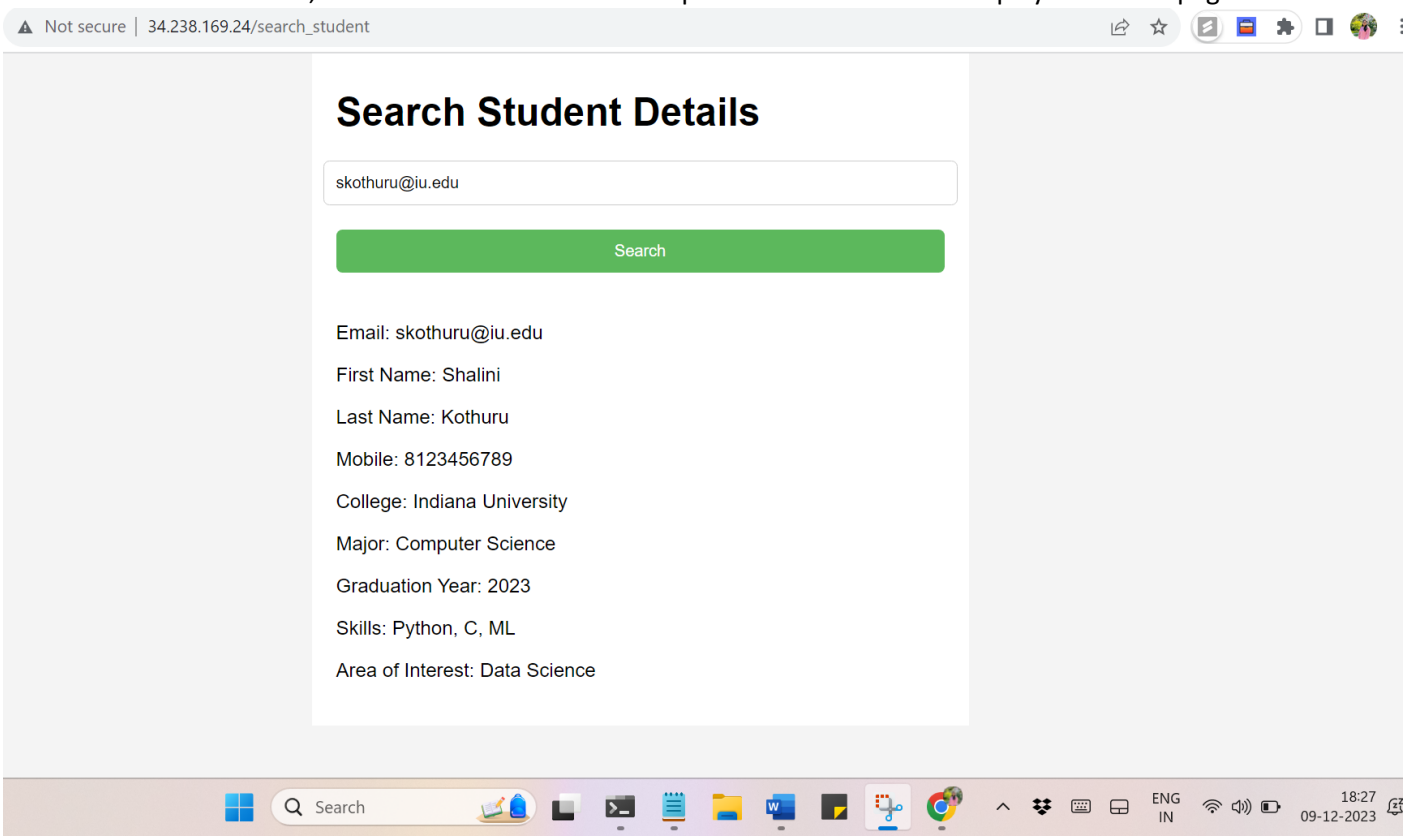


Once user completes filling details, on successful entry to database, it gives firstname's details entered in our database successfully. Since firstname was Shalini, it mentioned Shalini

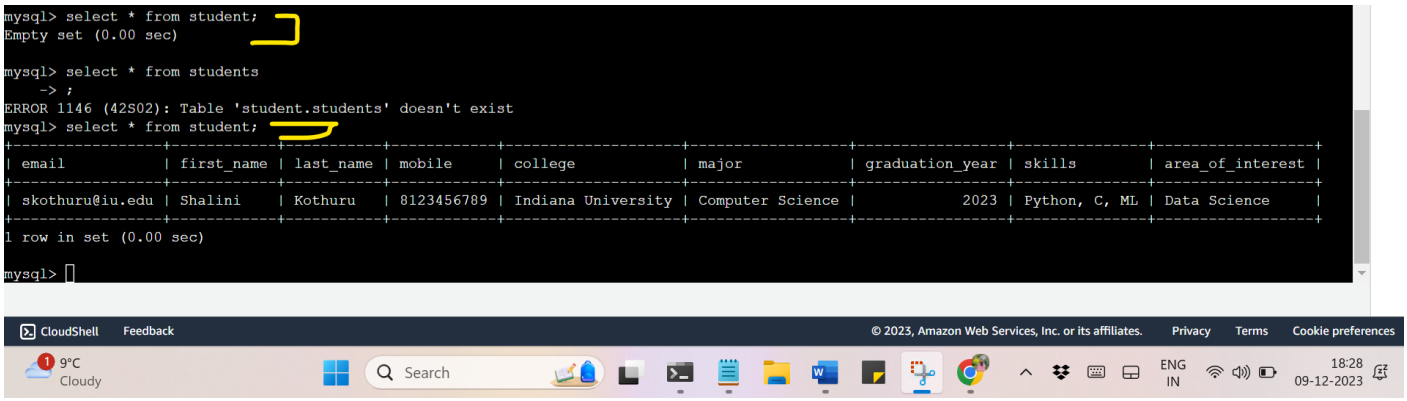
On clicking view student details in main page, it will redirect to below page



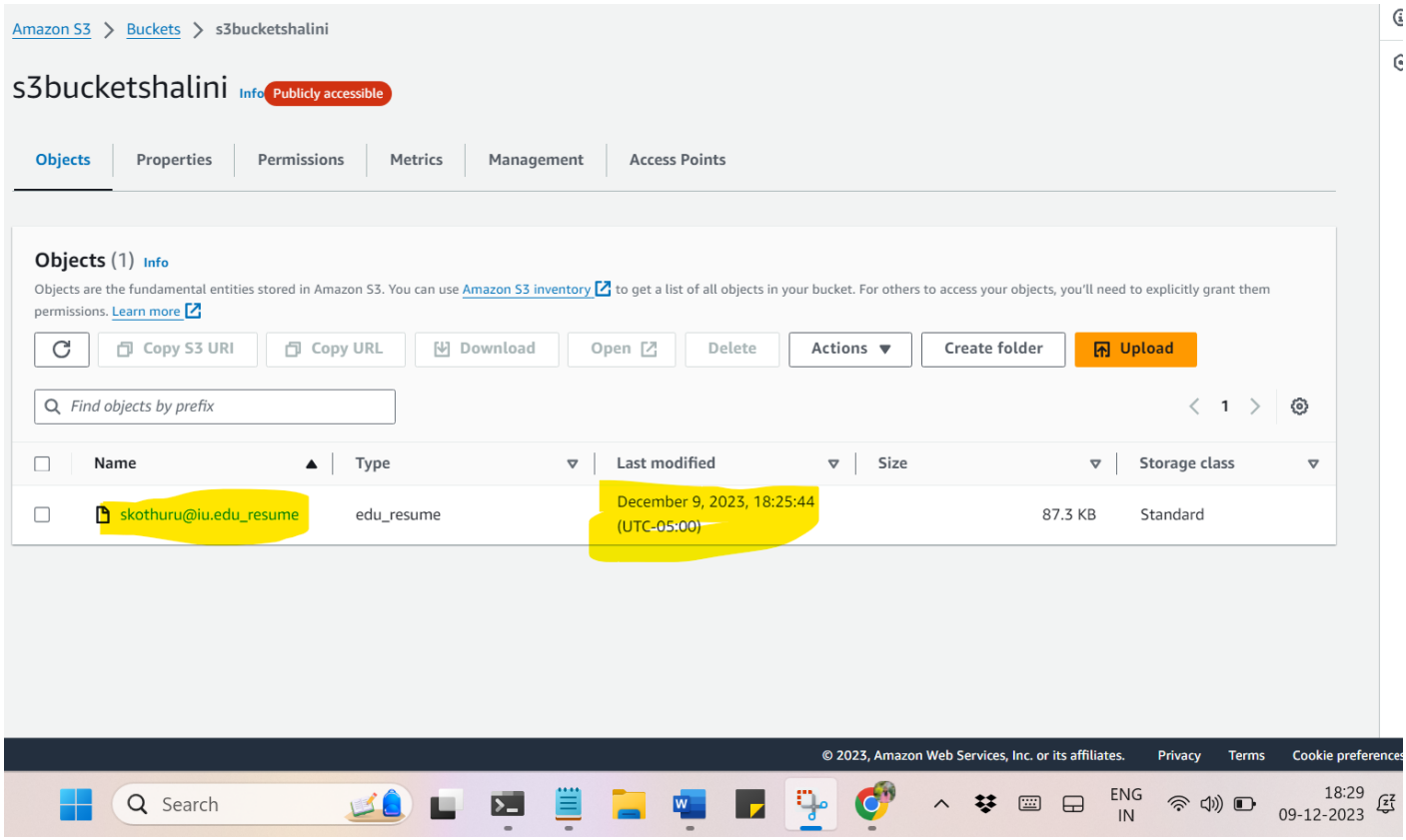
Once user enters email ID, the details associated with respective email ID will be displayed on web page.



Let’s check database records and S3 bucket. Initially there was no record in database but after add student through web application, we can see a record is inserted with same details.

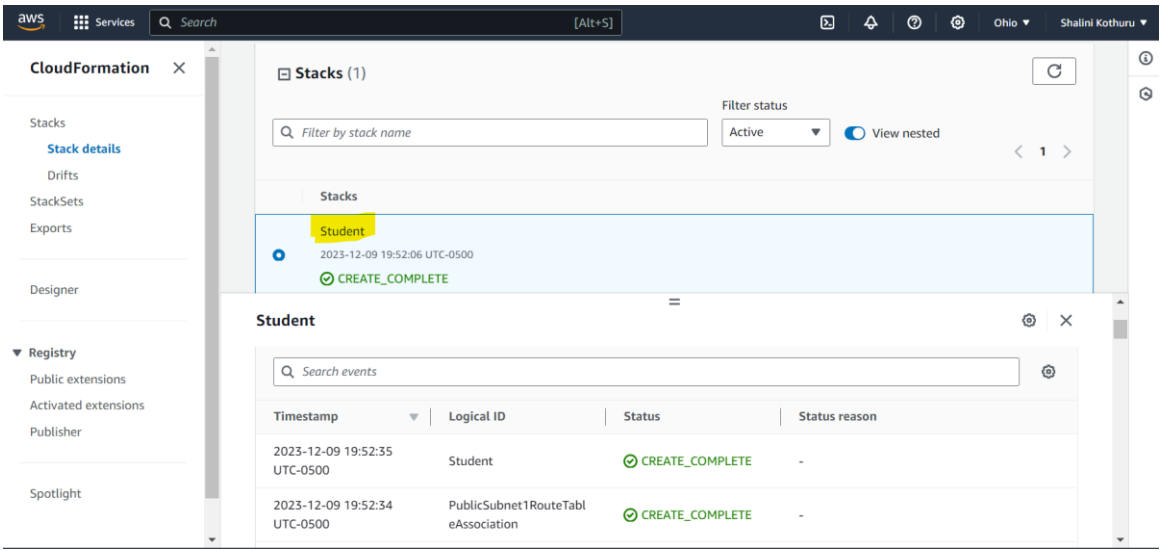


In S3 bucket, you can see file was added. In web application, I added logic to rename the file with emailID_resume so the file is stored in S3 as shown in picture. You can see timestamp, it matches with time when query was executed.



Step 6: Using Cloud Formation to create VPC.

Go to Cloud Formation in AWS and click on create stack. I have uploaded yaml file and named it student. As you can see, it was created successfully.



Explain the benefits and advantages of using AWS Compute, Storage, Database, and Infrastructure Management Services for your scenario.

- RDS: AWS RDS helped me in simplifying the setup, operation, and scaling of the relational database for the student web application. Unlike traditional databases which requires extensive planning for each phase, AWS RDS is easy to use. Using AWS RDS, it easy to backup and create. I can simply rename the database which changes its name everyehere where it was used.
- EC2: I can choose the operating system in EC2. For hosting different applications, I can create different EC2 instance with different configurations. I tried my application on Linux and Ubuntu when I had issue using one OS. There is option to autoscale applications. Although the application doesn't need autoscaling now, but its very useful.
- S3: I used it for storing the resume files that was uploaded by user. Storing it on local machines may lead to access problems but storing it on S3 buckets, it accessible on any bucket.
- CloudFormation: It allowed me to model and setup up AWS resources. I have the option of automating the deployment and updating the infrastructure which makes it easy to use the infrastructure.

Reflect on the challenges and considerations encountered during the implementation process and discuss how AWS services addressed those challenges.

- I had issue with using one OS, then I just deleted that instance and created another one in just a minute.
- Since I change my thought of doing different application after setting up some part of AWS configuration, it was easy for me to rename the objects to match according to my application.