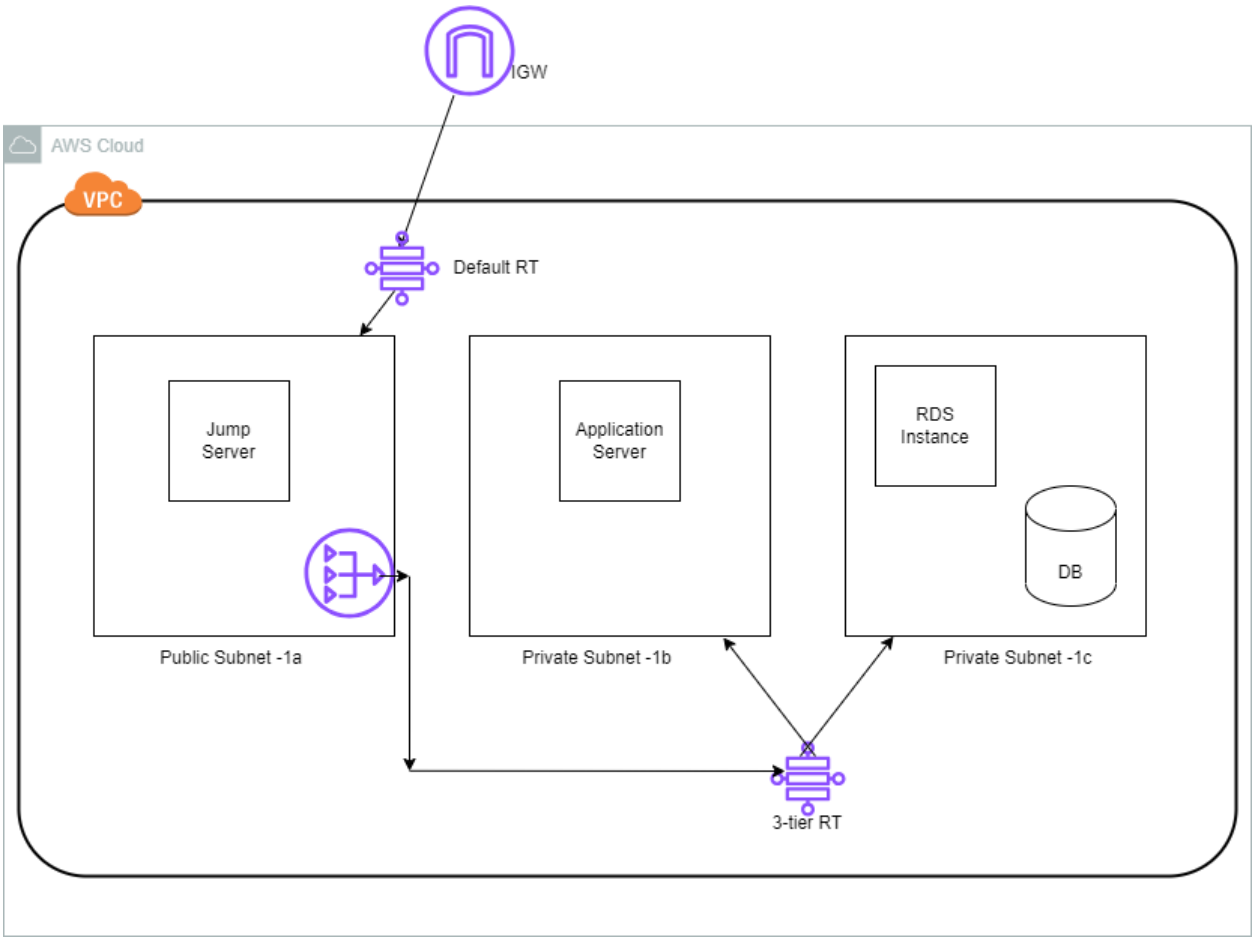
Building a 3-tier web application architecture with AWS

The three-tier architecture is the most popular implementation of a multi-tier architecture and consists of a single Jump server, presentation tier, and data tier. The following illustration shows an example of a simple, generic three-tier application.

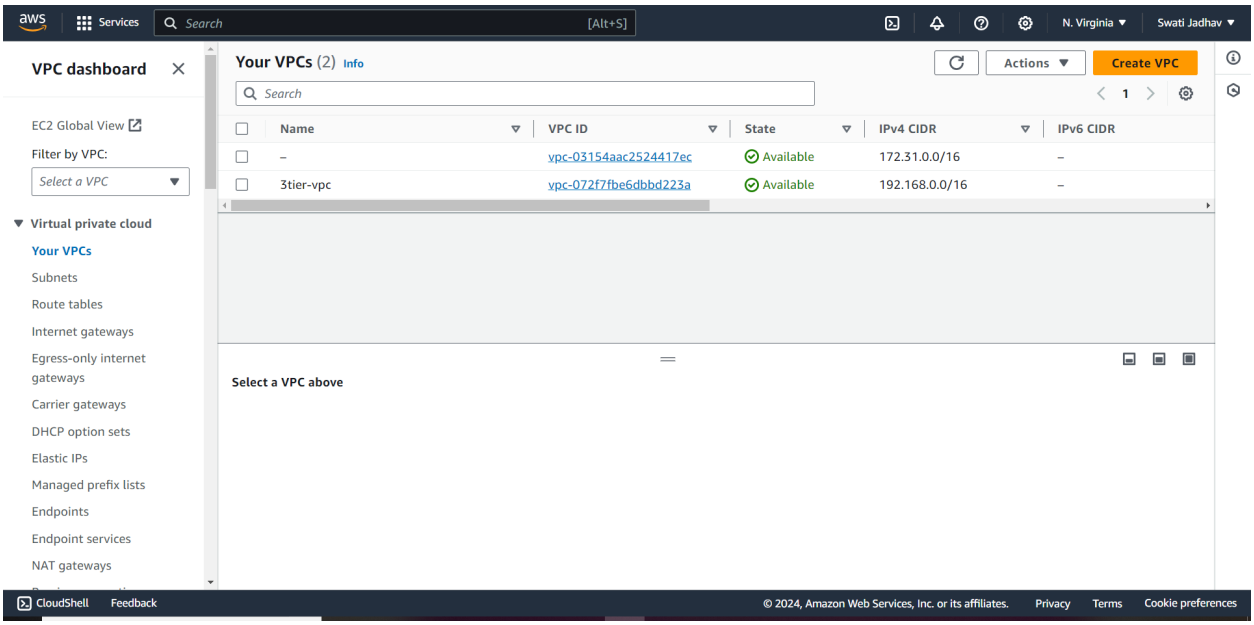
Prerequisites

- Familiarity with the AWS Management Console.
- Familiarity with VPC network structures, EC2 instances, and security groups.
- Familiarity with Linux commands, scripting, and SSH.
- Access to a command line tool.

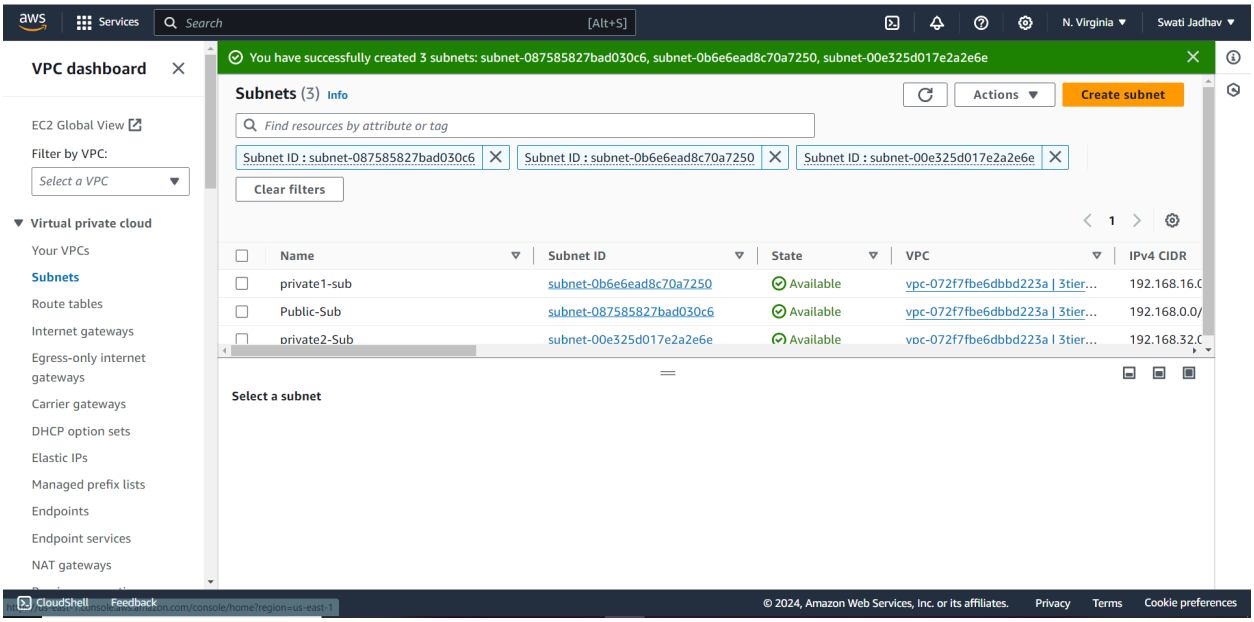


Steps:

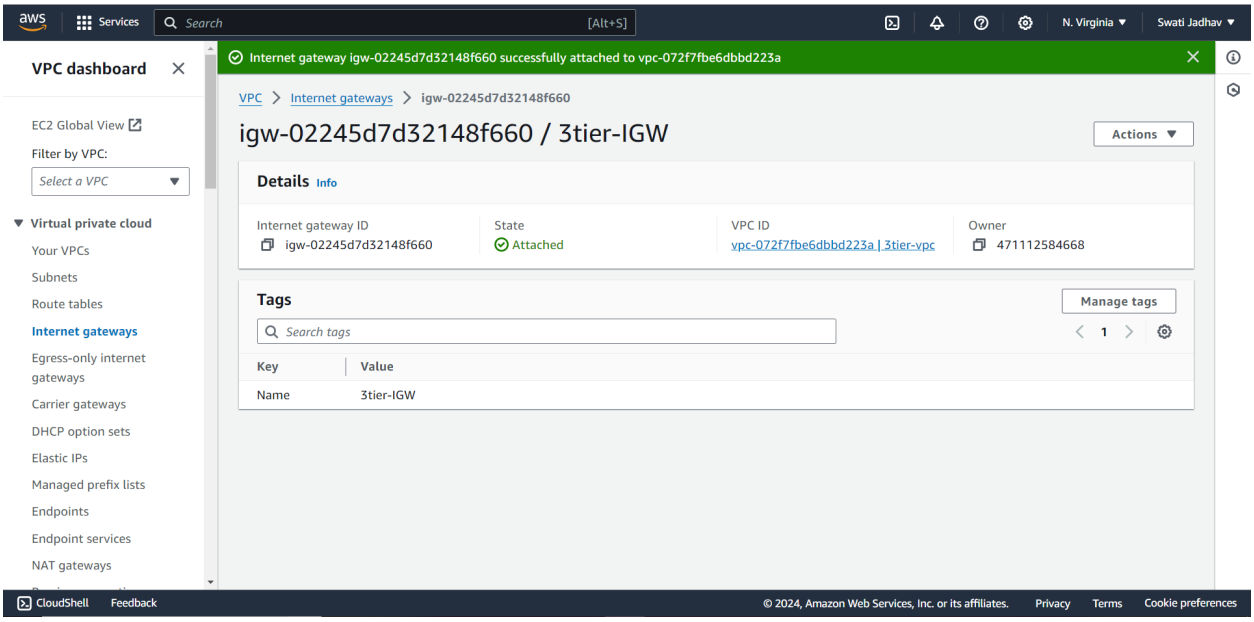
1. Logging to AWS using your login credentials
2. Create a VPC for any reason(N.Virginia) allocate an appropriate CIDR range and give the VPC name as(3 tier-VPC).



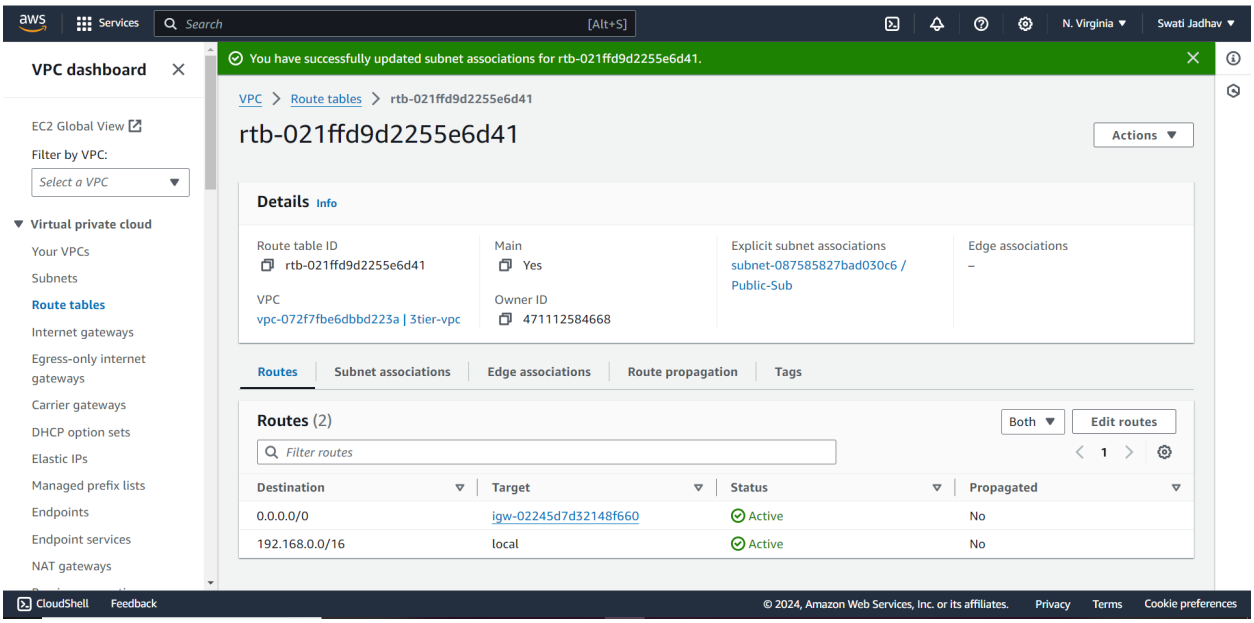
3. Create 3 Subnets into the VPC,1 Public-Subnet as Public-sub & 2 Private Subnet as private1-sub,private2-sub



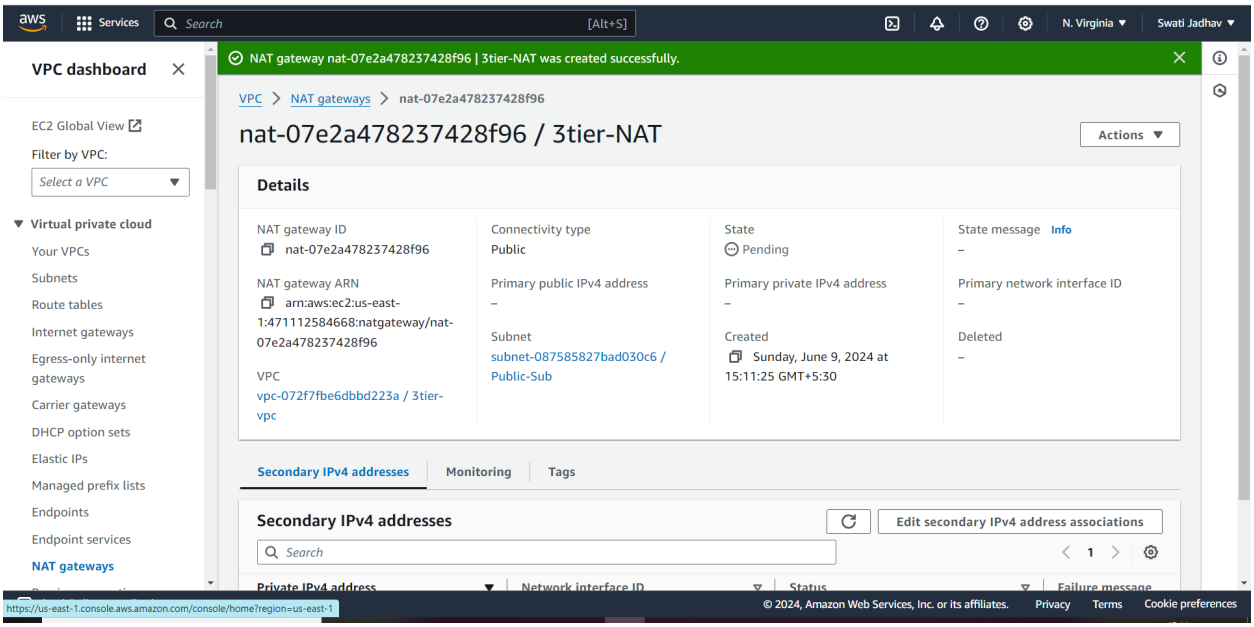
4. Create an Internet Gateway to provide Internet access to the VPC.



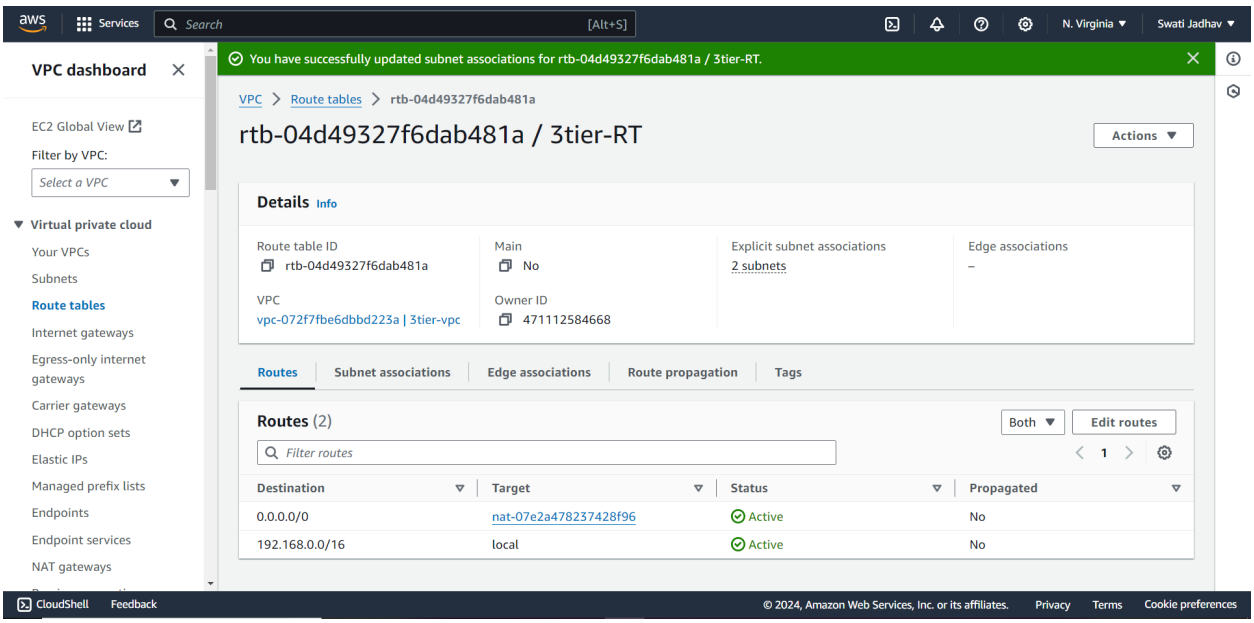
5. Route the Internet gateway to the default (main)route table.



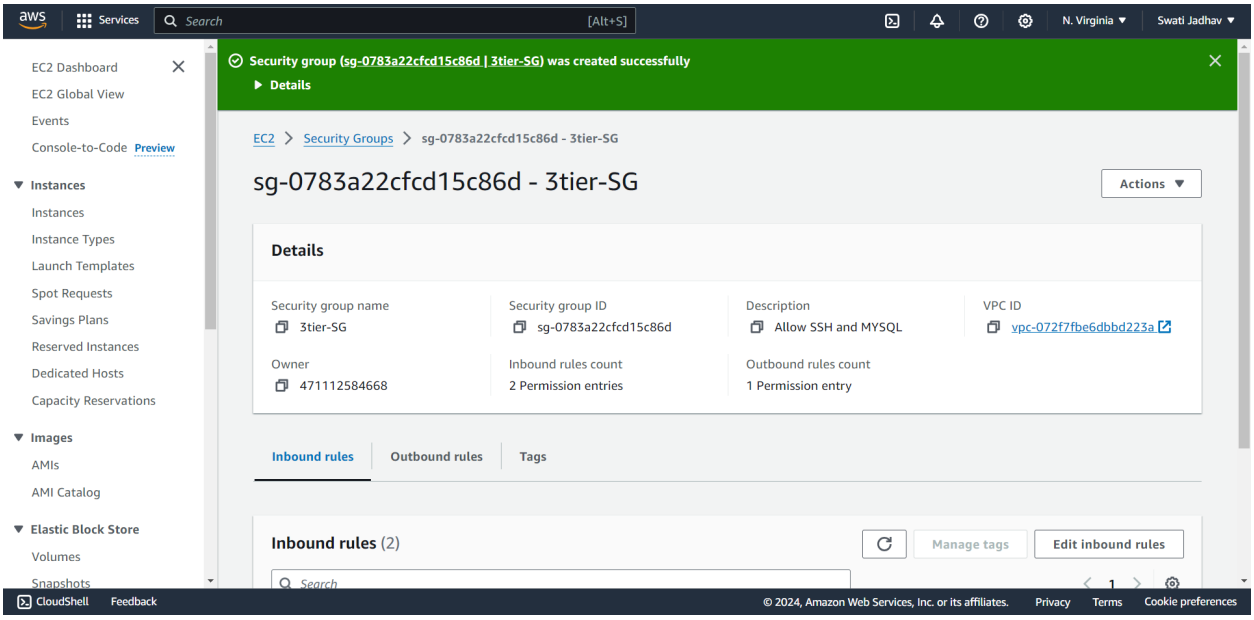
6. Create a NAT gateway(Network Access Translator) and associate it to both the Private subnet(private1-sub, private2-sub)



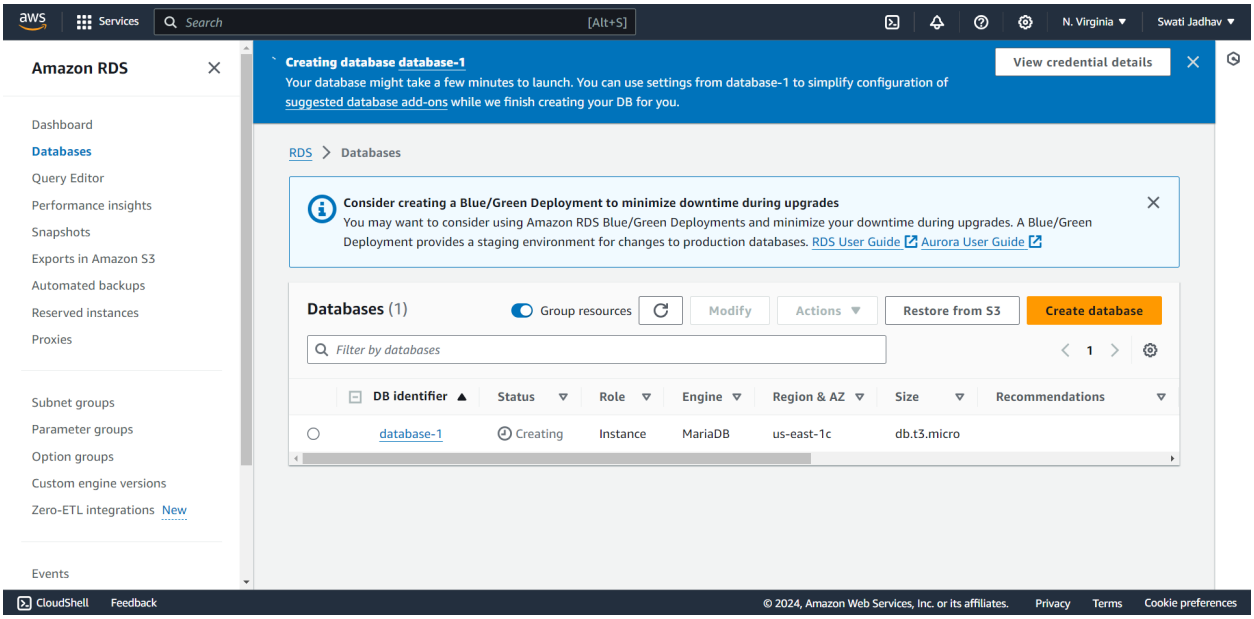
7. Create a Route table (3tier-RT) and route the NAT gateway



8. Create a Security Group with an inbound rule configured in it to allow SSH (22) and MYSQL(3306).



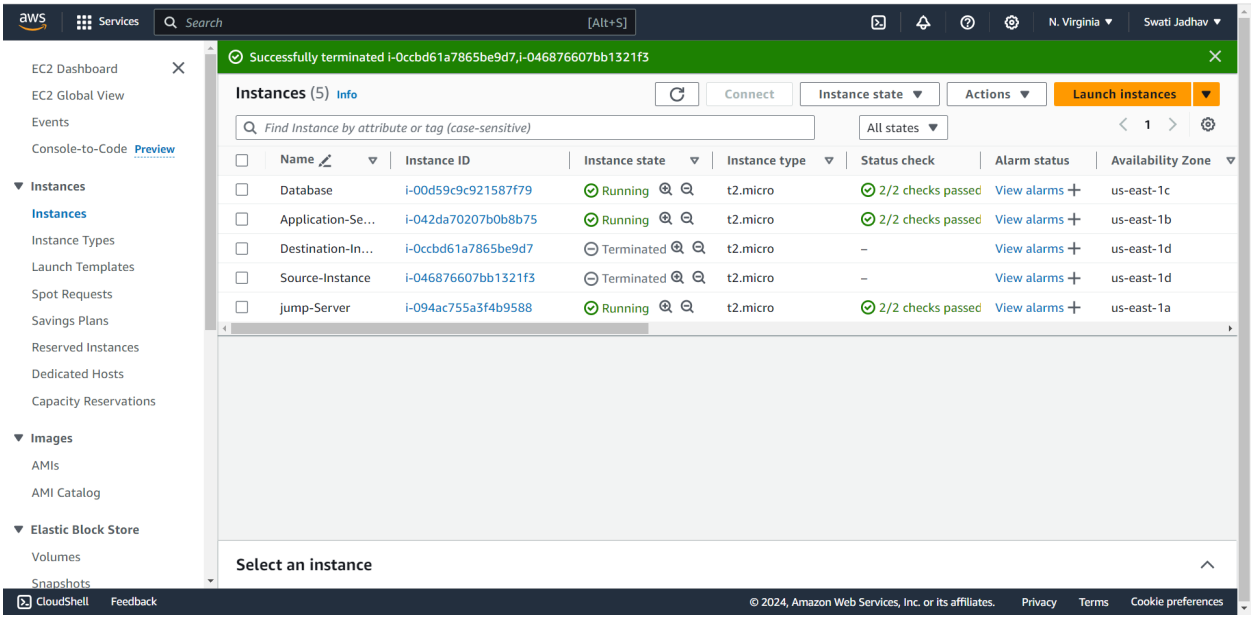
9. Create an RDS by attaching the Security group to it.



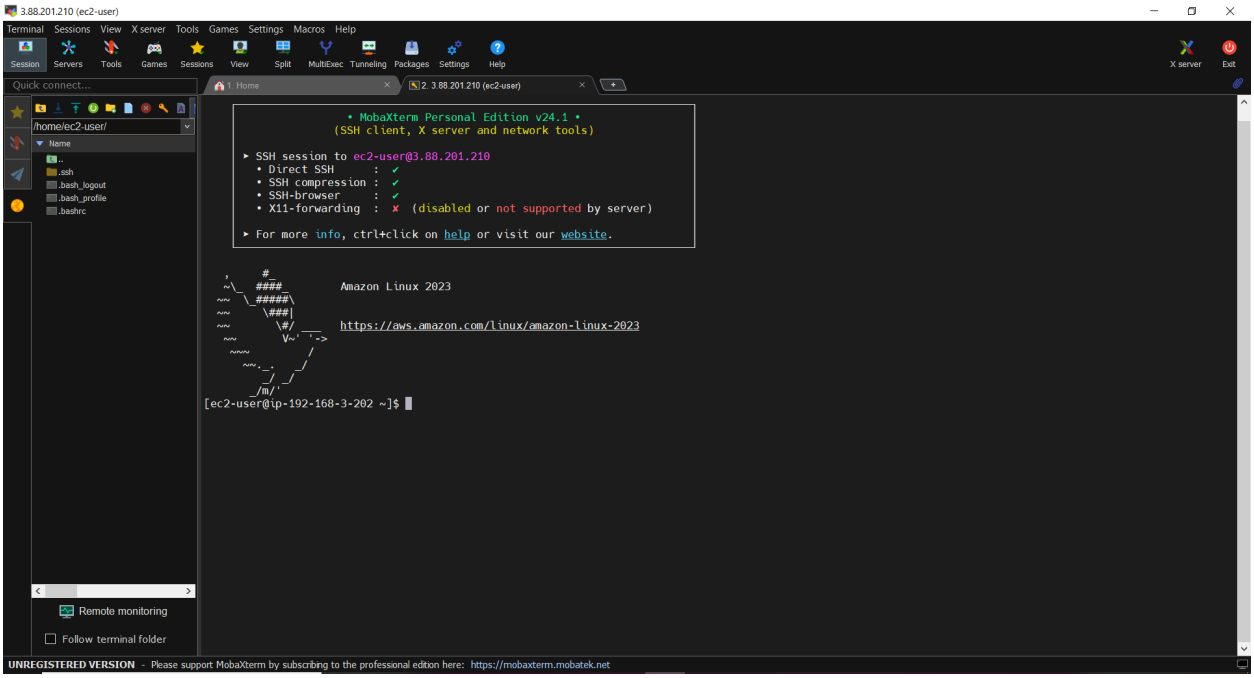
10. Create 3 Instance

- 1. First instance in the public subnet (jump Server)with permission HTTP(80) and SSH(22)

- 2.Second instance is in the private1-subnet (Application-Server)with permission Tomcat(8080) and SSH(22).
- 3.The third instance is in the private2-subnet (Database)with permission MySQL(3306) and SSH(22).

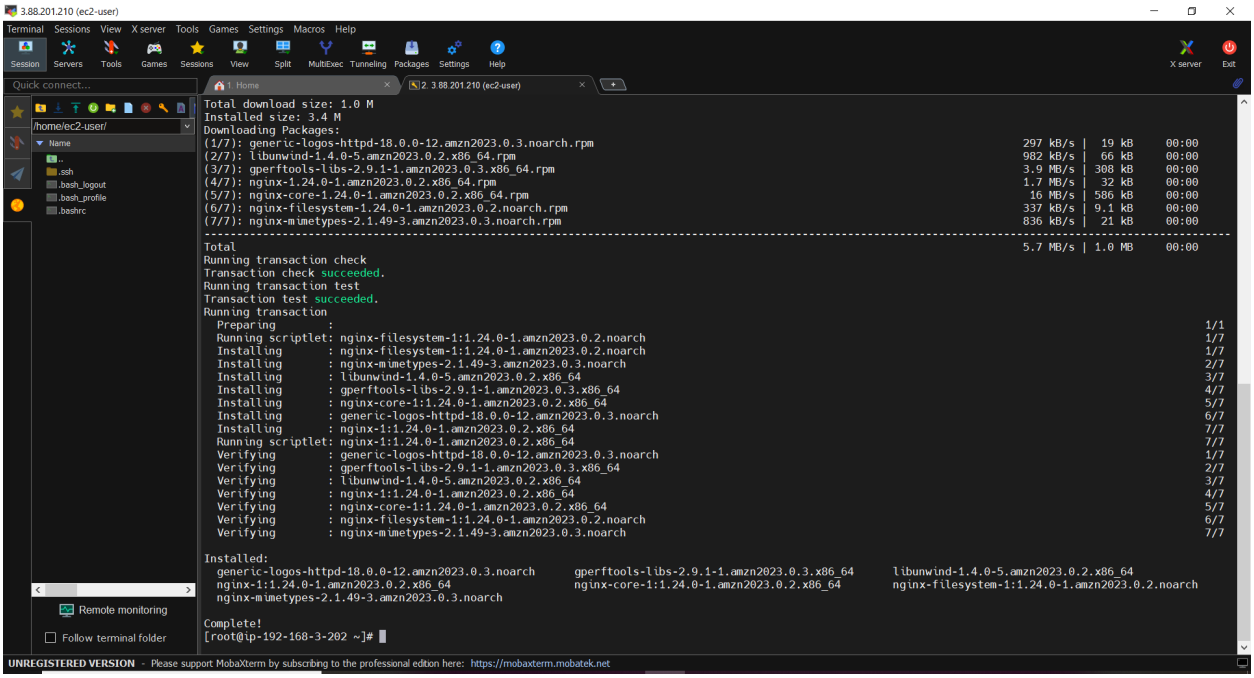


11. Connect to the Jump-server using SSH.



12. Install Nginx on that server

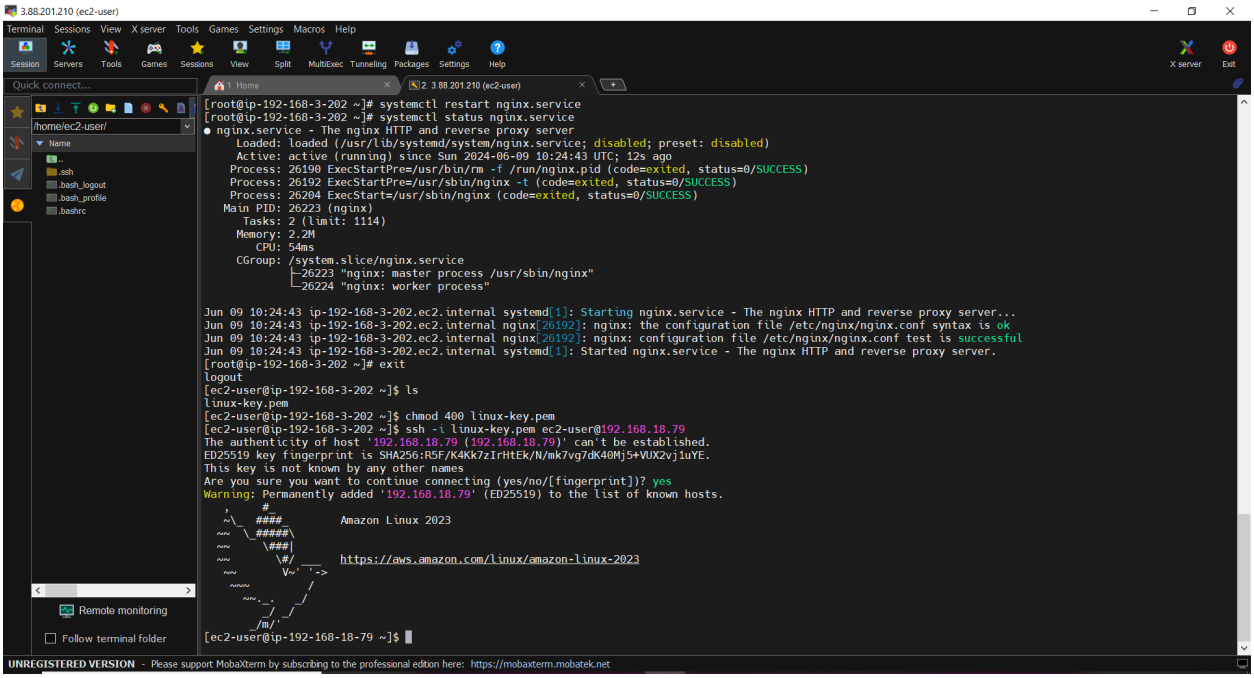
```
sudo -i
yum install nginx -y

vim /etc/nginx/nginx.conf

location /{

proxy_pass http://privateIPofApplicationserver:8080/student/;

}
```



13. Start the Nginx service and copy the key to the jump-server

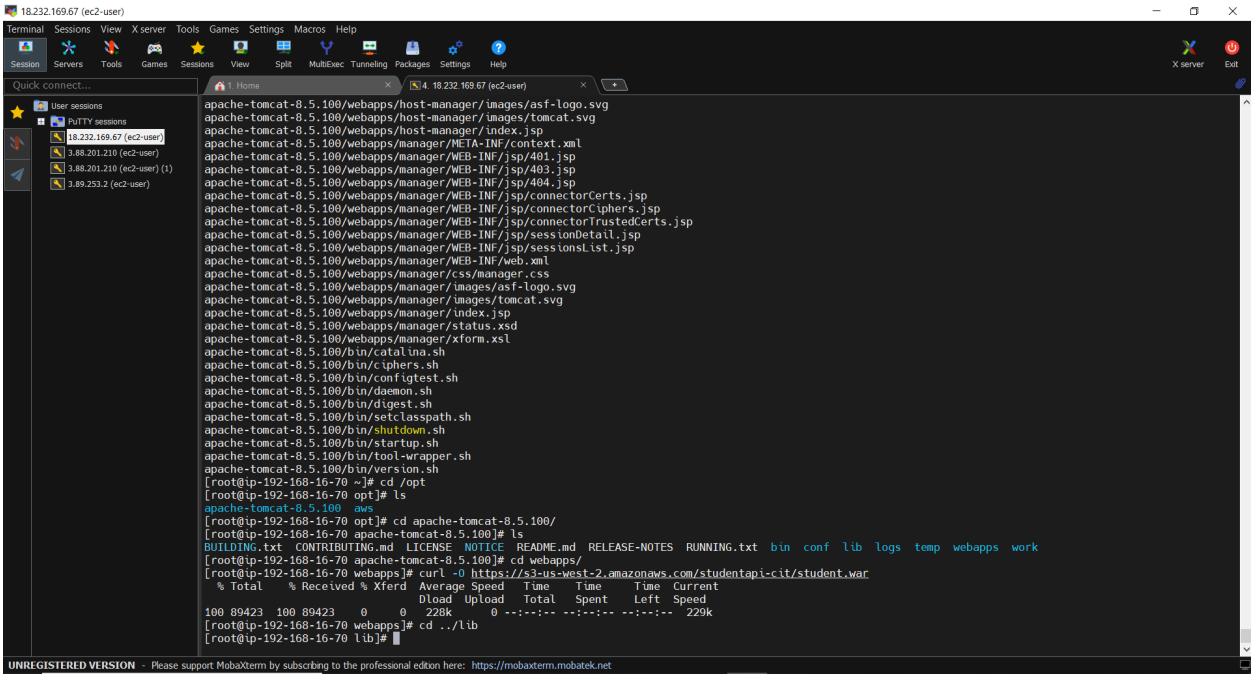
```
systemctl restart nginx.service
To copy the key to the jump server command is
scp -i key-name key-name ec2-user@publicIP:/home/ec2-user
change the key permission to
chmod 400 key-name
```

14. SSH to Application-server



15. Install java

```
sudo –i
• yum install java –y
• curl -O https://dlcdn.apache.org/tomcat/tomcat8/v8.5.100/bin/a
• ls
• tar -xvf apache-tomcat-8.5.100.tar.gz -C /opt
cd /opt
• cd apache-tomcat-8.5.100/
• cd webapps
• curl -O https://s3-us-west-2.amazonaws.com/studentapicit/stude
• cd ../lib
```



16. Install a connector to connect the application and the database.

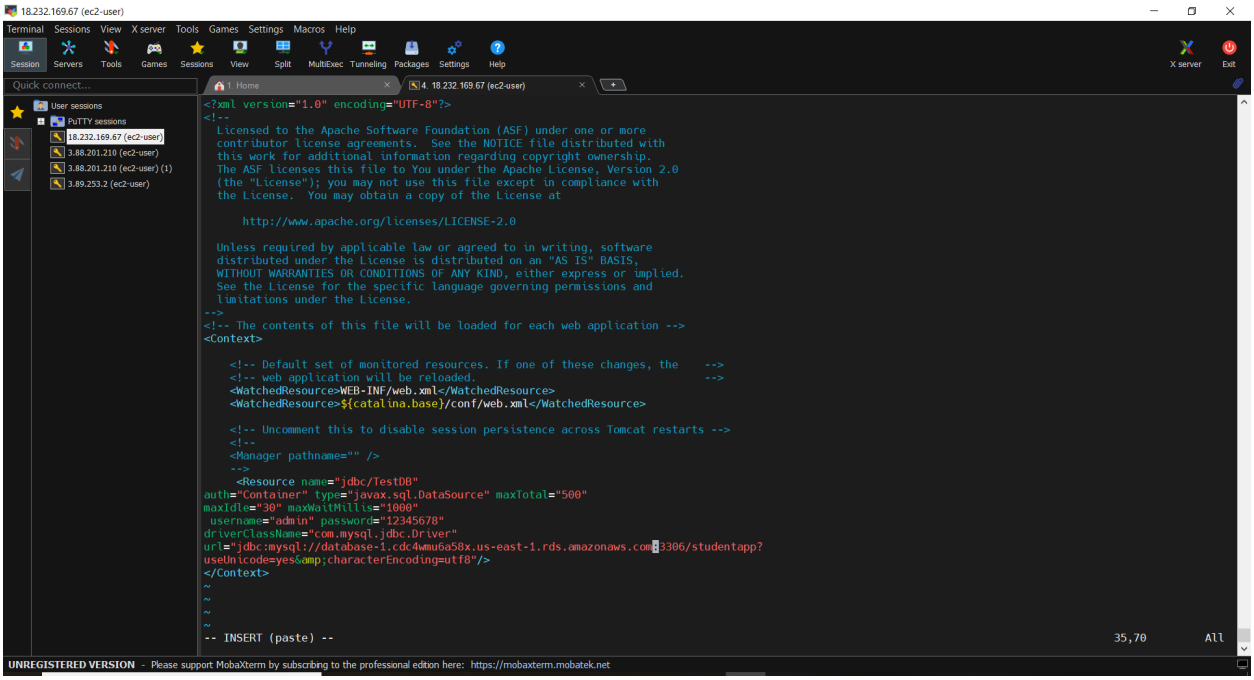
```
curl -O https://s3-us-west-2.amazonaws.com/studentapi-cit/mysql
```

- Ls(mysql-connector.jar)
- Cd ../conf/

```
                                 Dload  Upload   Total   Spent    Left  Speed
100   303    0   303    0     0   1440      0 --:--:-- --:--:-- --:--:--  1442
[ec2-user@ip-192-168-16-61 webapps]$ cd ../lib
[ec2-user@ip-192-168-16-61 lib]$ curl -O https://s3-us-west-2.amazonaws.com/studentapi-cit/mysqlconnector.jar
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   289    0   289    0     0   1292      0 --:--:-- --:--:-- --:--:--  1295
[ec2-user@ip-192-168-16-61 lib]$ ls
annotations-api.jar       catalina.jar   jaspic-api.jar      tomcat-coyote.jar   tomcat-i18n-ja.jar     tomcat-jni.jar
catalina-ant.jar          ecj-4.6.3.jar  jsp-api.jar         tomcat-dbcp.jar     tomcat-i18n-ko.jar     tomcat-util-scan.jar
catalina-ha.jar           el-api.jar     mysqlconnector.jar  tomcat-i18n-de.jar  tomcat-i18n-ru.jar     tomcat-util.jar
catalina-storeconfig.jar  jasper-el.jar  servlet-api.jar     tomcat-i18n-es.jar  tomcat-i18n-zh-CN.jar  tomcat-websocket.jar
catalina-tribes.jar       jasper.jar     tomcat-api.jar      tomcat-i18n-fr.jar  tomcat-jdbc.jar        websocket-api.jar
[ec2-user@ip-192-168-16-61 lib]$ cd ../conf/
[ec2-user@ip-192-168-16-61 conf]$ vim context.xml
[ec2-user@ip-192-168-16-61 conf]$ vim context.xml
[ec2-user@ip-192-168-16-61 conf]$ cd ../bin
[ec2-user@ip-192-168-16-61 bin]$ ./catalina.sh start
Using CATALINA_BASE:   /home/ec2-user/apache-tomcat-8.5.100
Using CATALINA_HOME:   /home/ec2-user/apache-tomcat-8.5.100
Using CATALINA_TMPDIR: /home/ec2-user/apache-tomcat-8.5.100/temp
Using JRE_HOME:        /usr
Using CLASSPATH:       /home/ec2-user/apache-tomcat-8.5.100/bin/bootstrap.jar:/home/ec2-user/apache-tomcat-8.5.100/bin/tomcat-juli.jar
Using CATALINA_OPTS:
Tomcat started.
[ec2-user@ip-192-168-16-61 bin]$
```

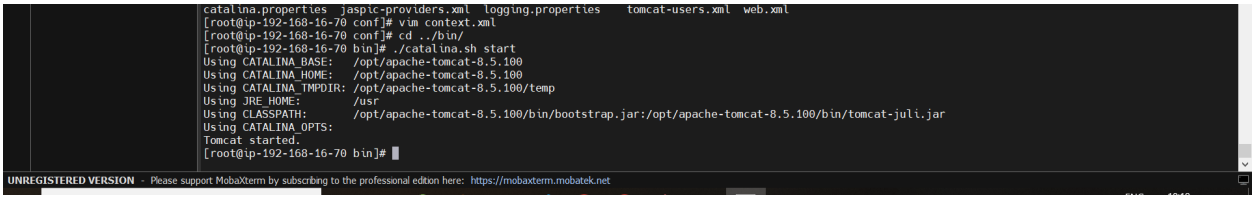
17.

```
Vim context.xml
• (in context tab [last of page]) <Resource name="jdbc/TestDB"
auth="Container" type="javax.sql.DataSource" maxTotal="500"
maxIdle="30" maxWaitMillis="1000"
username="admin" password="12345678"
driverClassName="com.mysql.jdbc.Driver"
url="jdbc:mysql://endpoint:3306/studentapp?
useUnicode=yes&characterEncoding=utf8"/>
```



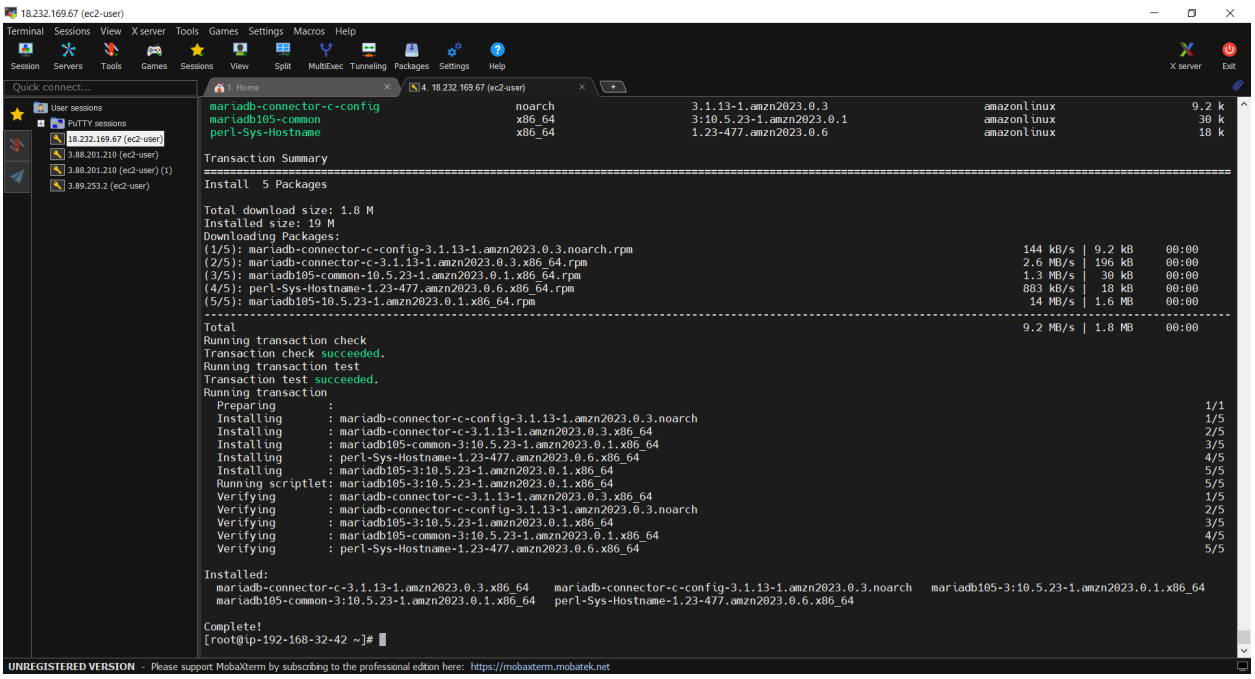
18. Start the Tomcat server.

```
cd ../bin
• ./catalina.sh start
• exit
```



19. SSH to the Database instance

- install mariadb

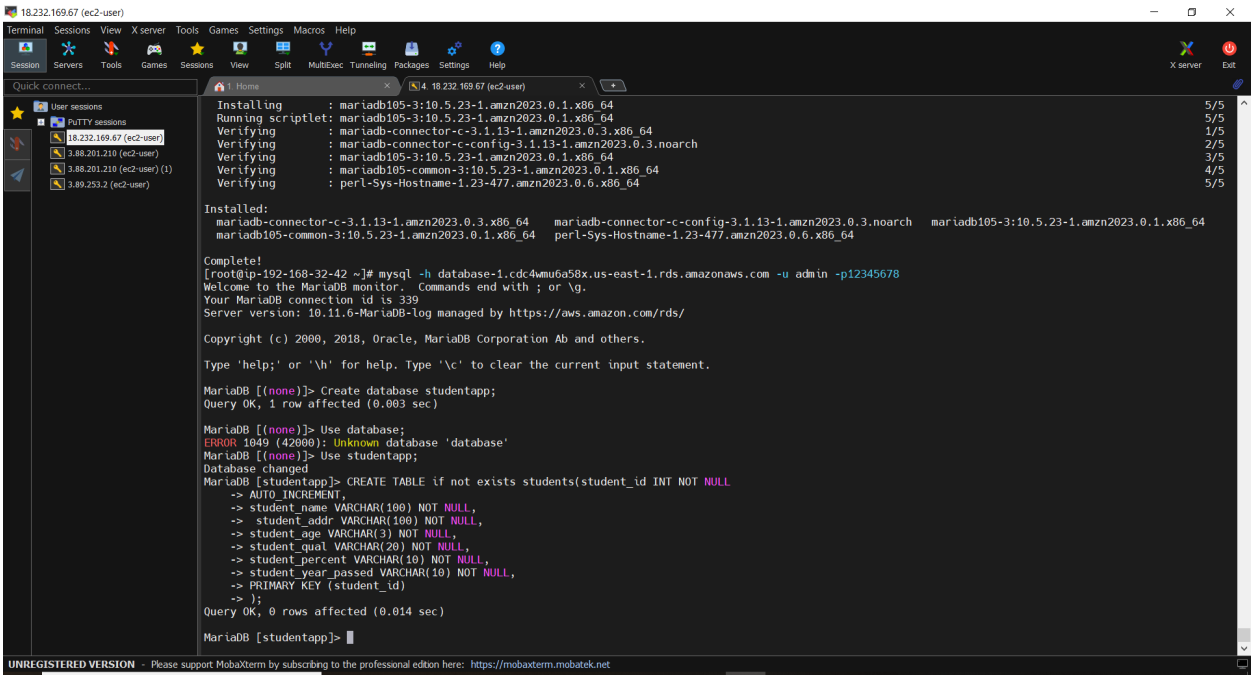


20. Connect to RDS

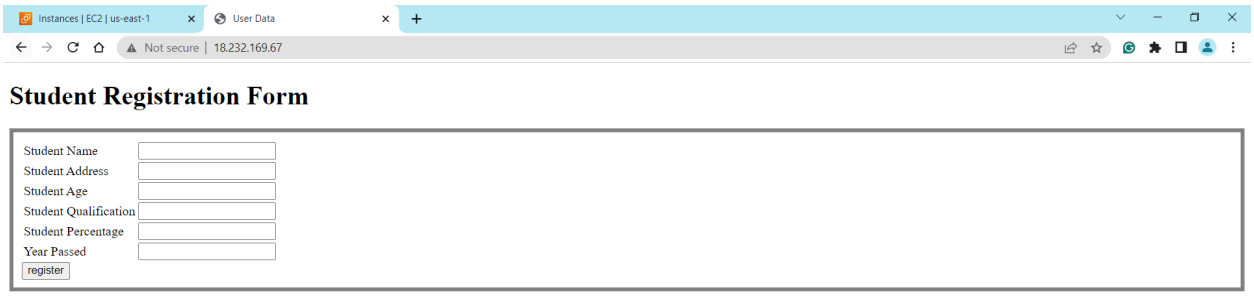
- create a table.

```
Ssh –i key-name ec2-user@dbinstanceip (connected to db. instanc
• Yum install mariadb105 –y
• Mysql –h rdsendpoint –u admin –p(connected to RDS)
• Create database;
• Use database;
• CREATE TABLE if not exists students(student_id INT NOT NULL
```

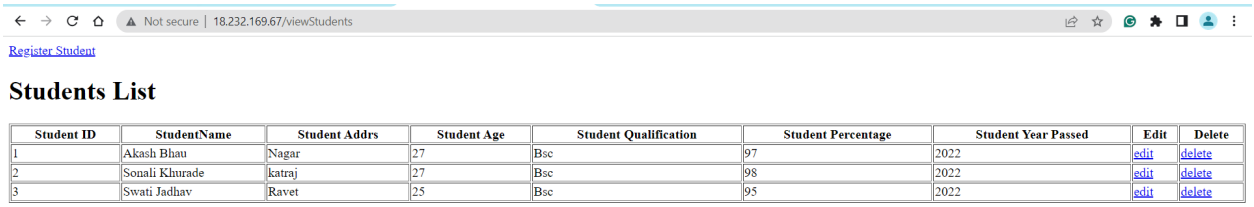
```
AUTO_INCREMENT,
student_name VARCHAR(100) NOT NULL,
student_addr VARCHAR(100) NOT NULL,
student_age VARCHAR(3) NOT NULL,
student_qual VARCHAR(20) NOT NULL,
student_percent VARCHAR(10) NOT NULL,
student_year_passed VARCHAR(10) NOT NULL,
PRIMARY KEY (student_id)
);
```



21. Hit the Public IP of the Jump-server which is present in the Public-subnet.
22. The webpage is successfully hosted.



23. The data is stored in the RDS that we had created.



Summary:

Hosted a website and stored the data in the database (RDS).Created a **VPC** and **three subnets** in different **Availability Zones**. which **one public subnet** and **two private subnets**. In the public subnet, we created a **jump server** and in one private subnet where we were going to host the application in the **application server** and in another private subnet the **database server** for storing data. Configured **Nginx in the jump server**, **java, and application into the application server**. **Install MariaDB in the database and create a table**. This is how we hosted an application in one server and stored data into another server.

Errors in 3-tier application