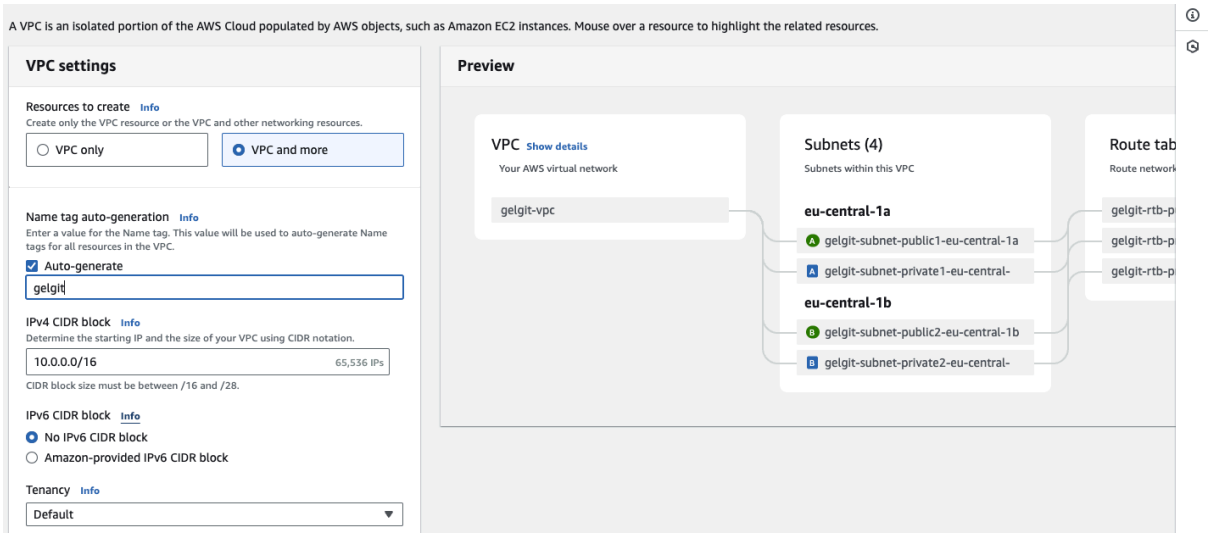
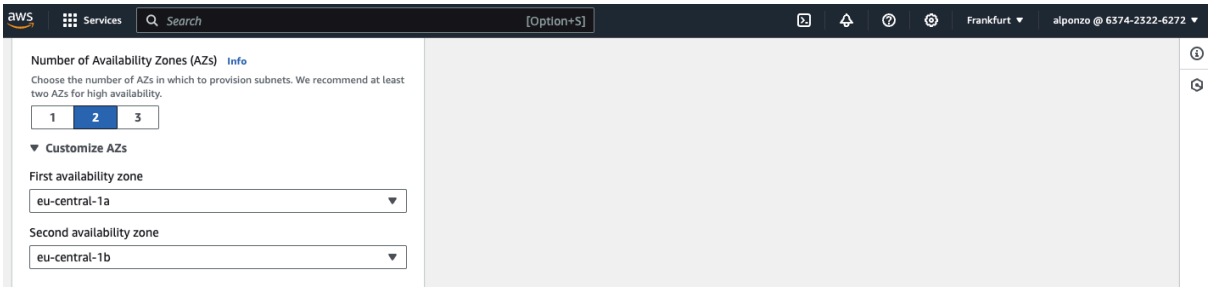
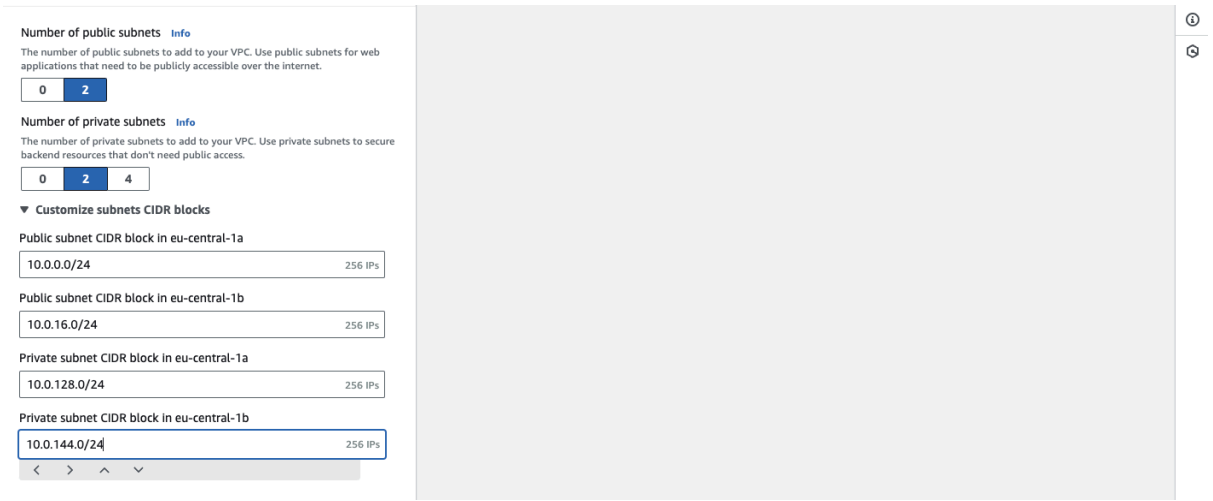
Project Manual

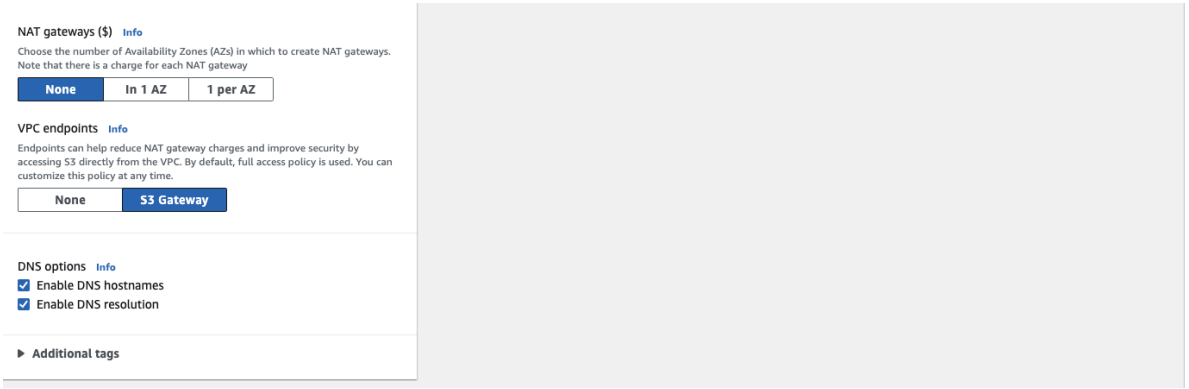
Database Part

VPC

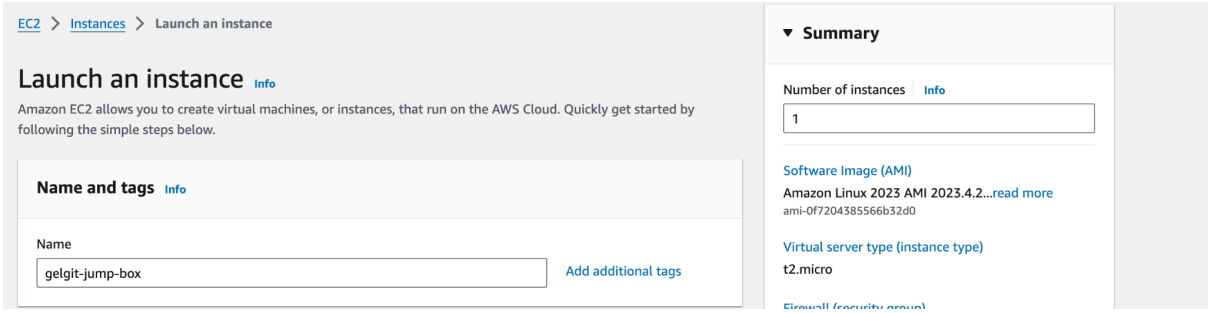




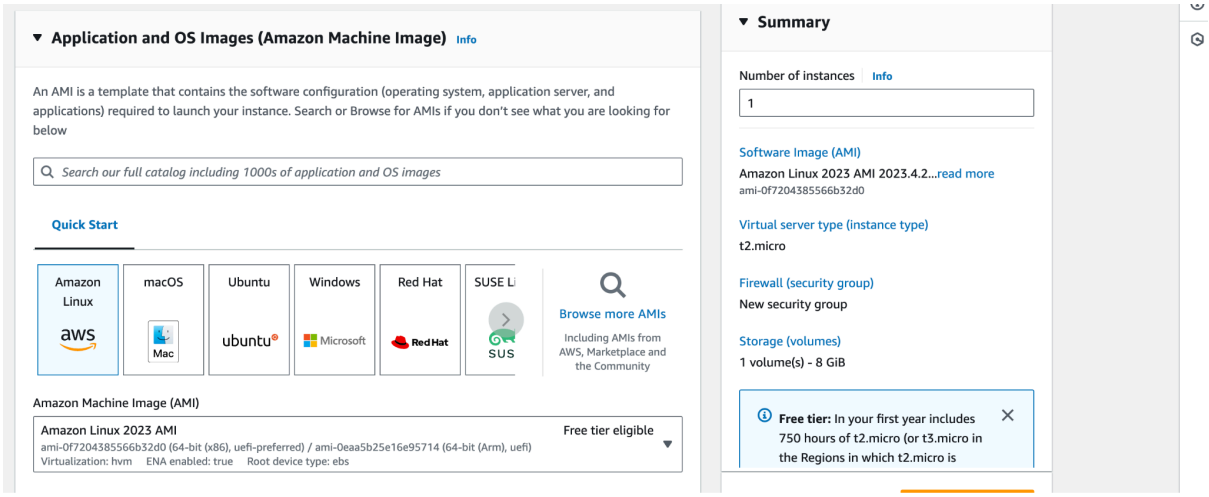


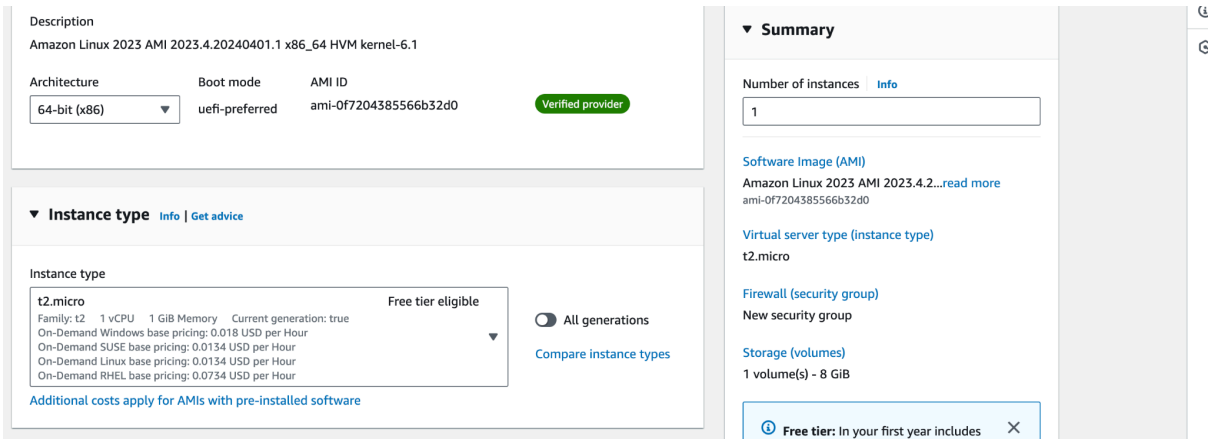


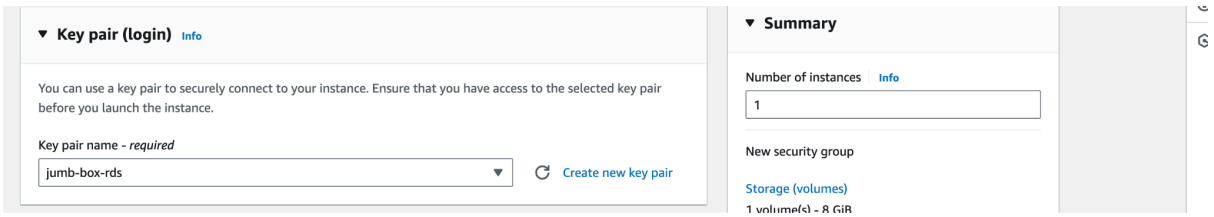
EC2: Bastion Host / Jump Box - To connect DB



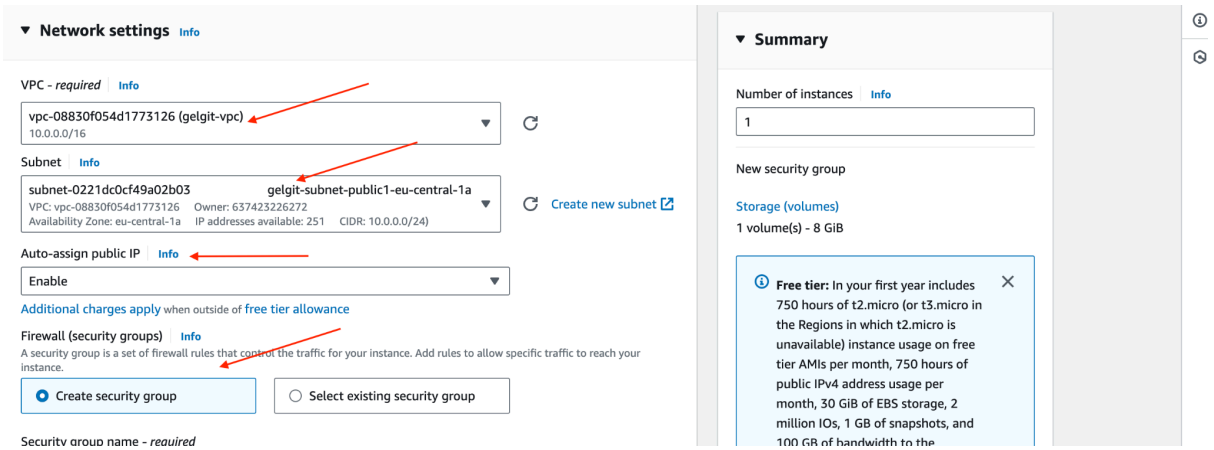
Currently used: gelgit-jumpbox-rds-ec2

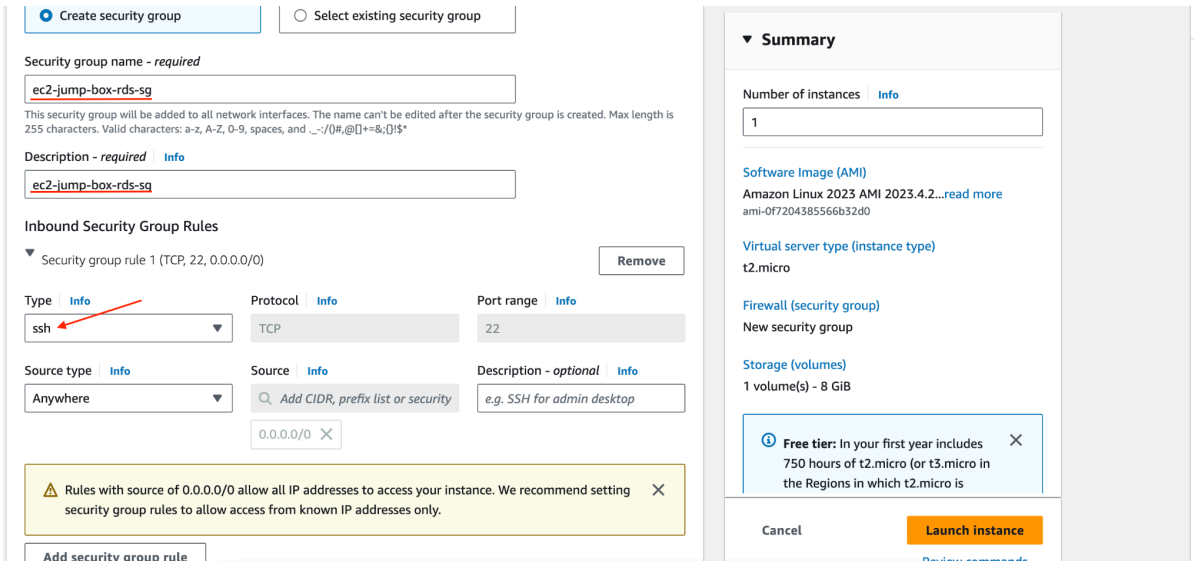




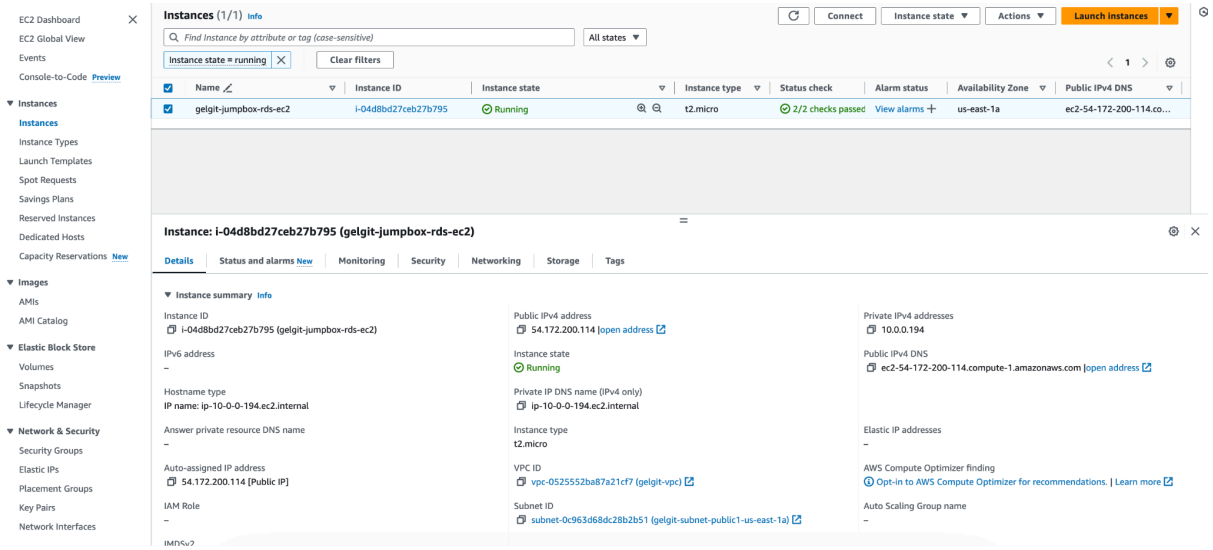


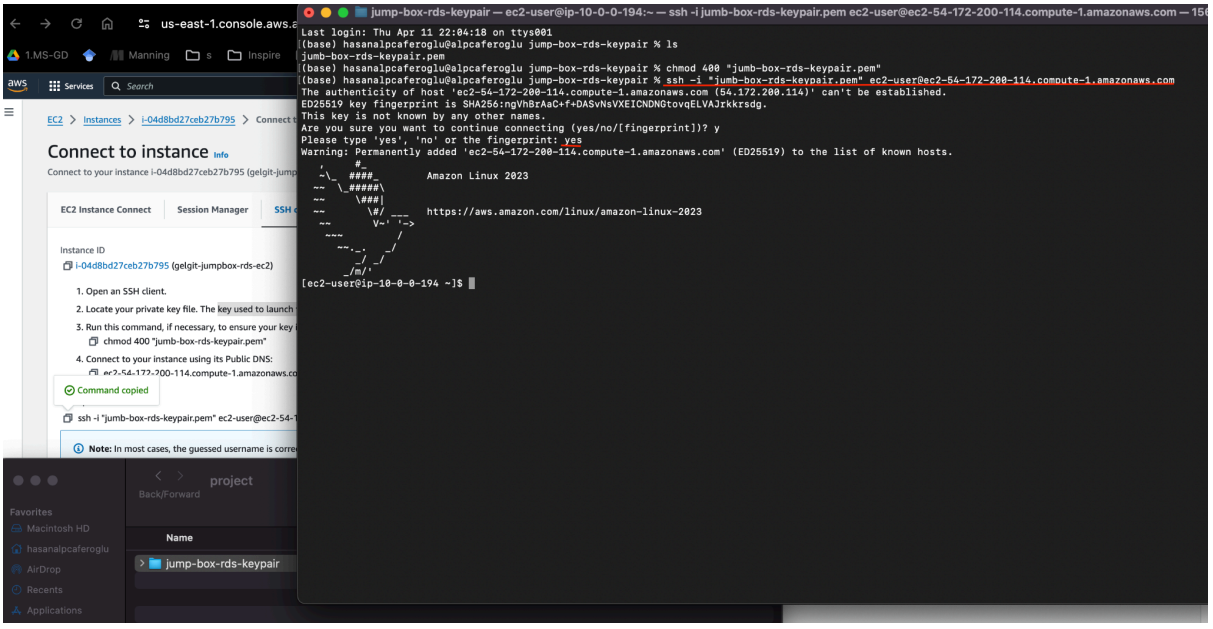
Currently used key pair: jumb-box-rds-keypair





Current security group name = jumpbox-rds / jumpbox-rds-sg

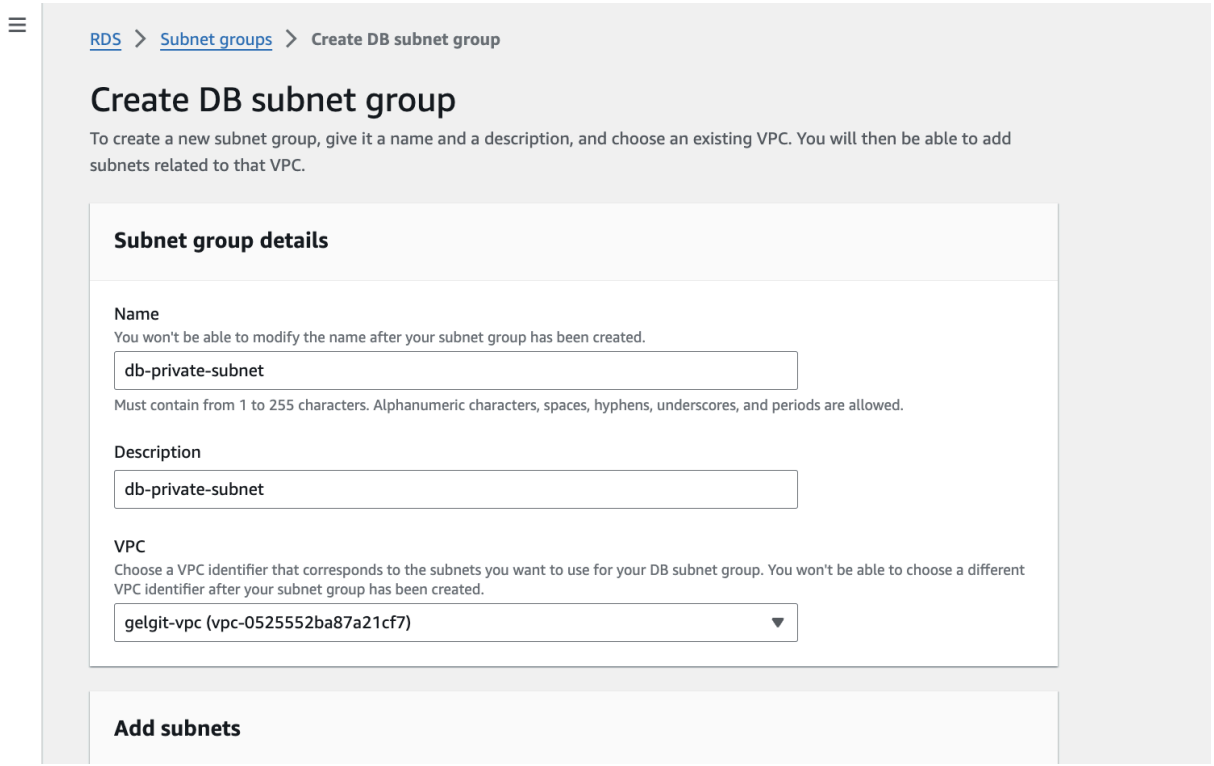




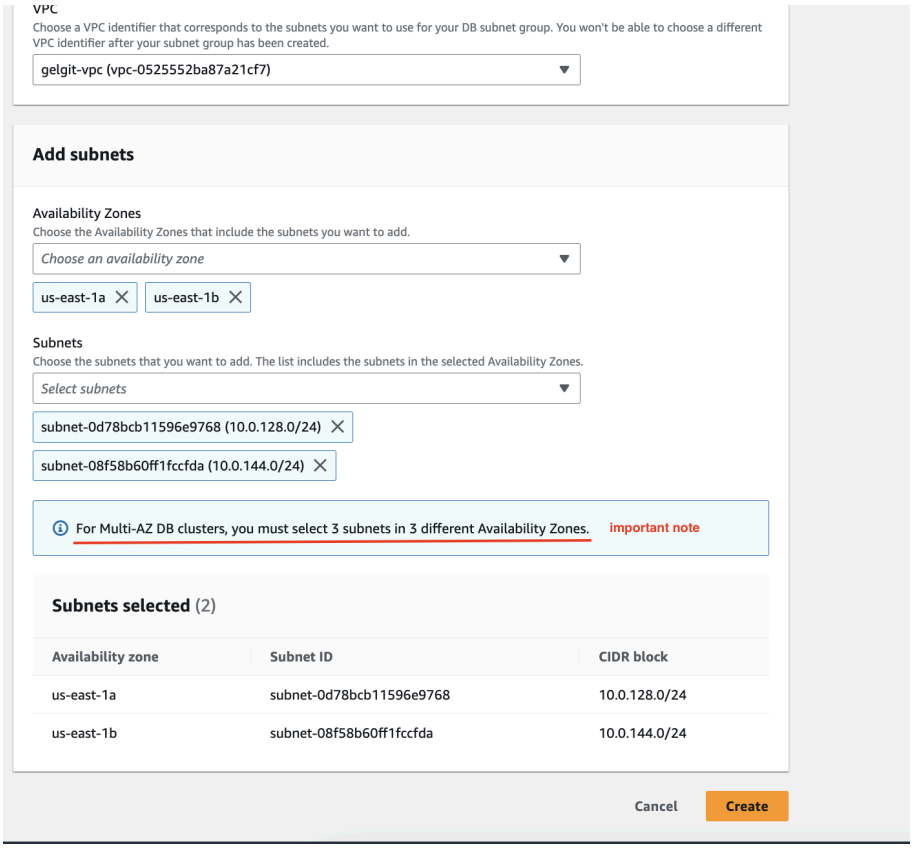
AWS RDS MySQL - N. Virginia

- https://beabetterdev.com/2022/12/13/how-to-connect-to-an-rds-or-aurora-database-in-a-private-subnet/

Step 7 – Create RDS Subnet Group

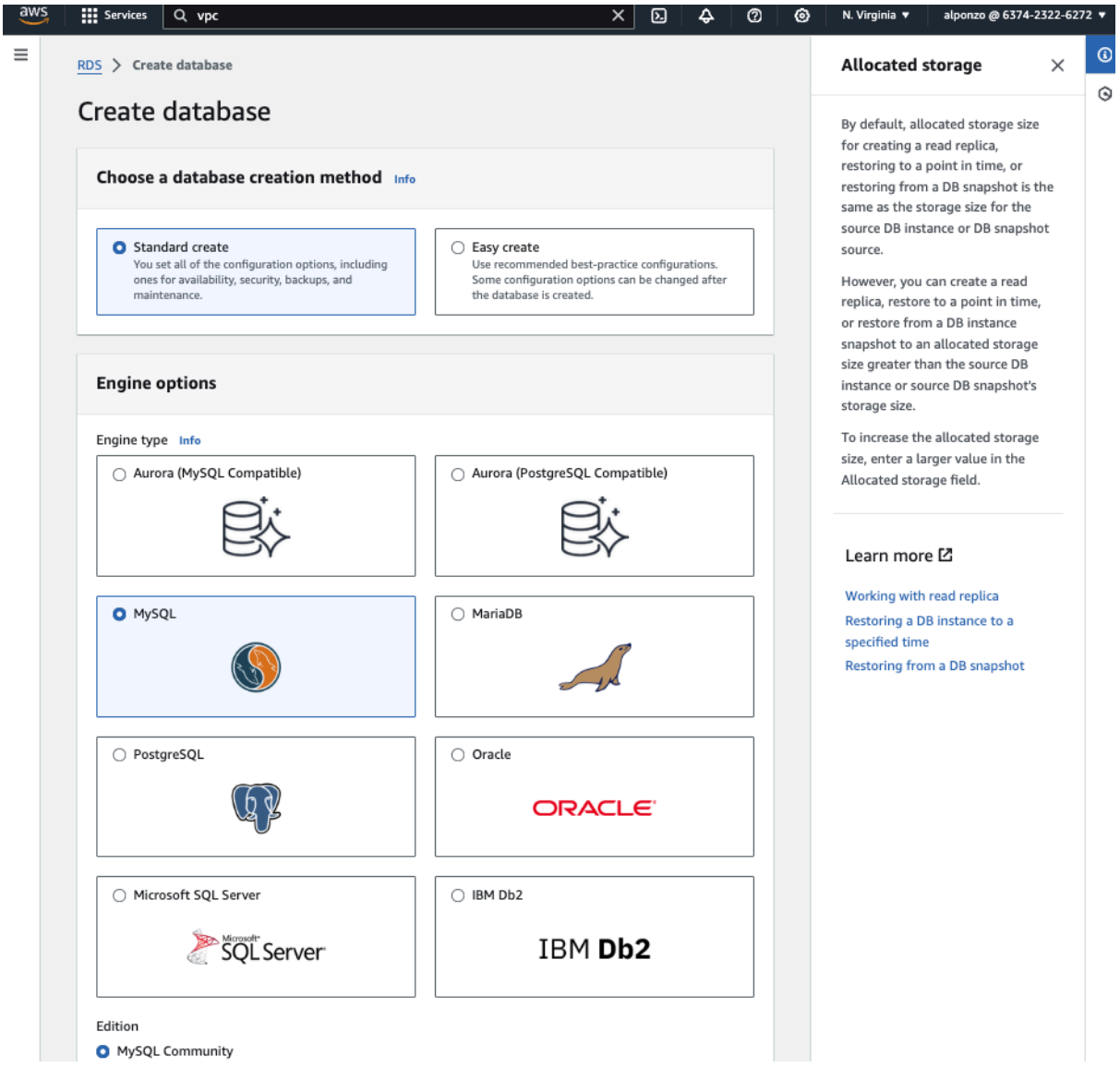


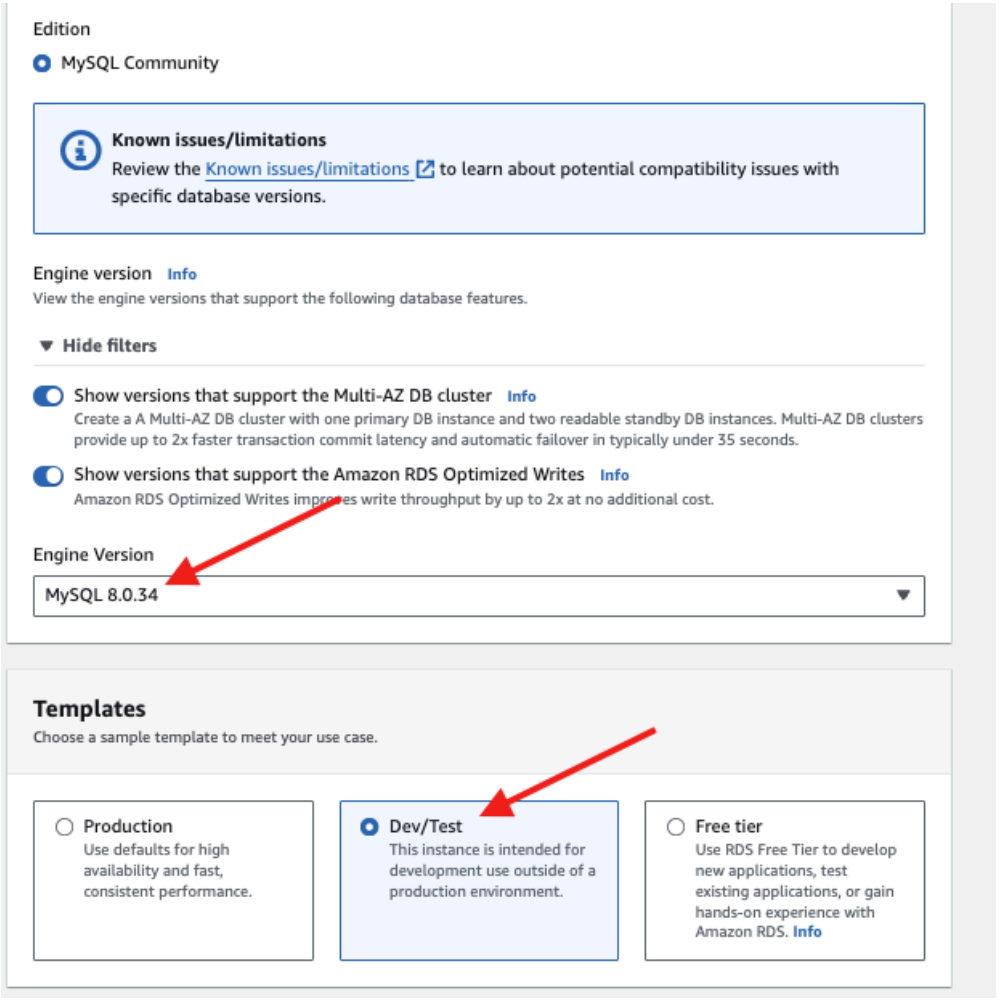
select the two private subnets from the two availability zones

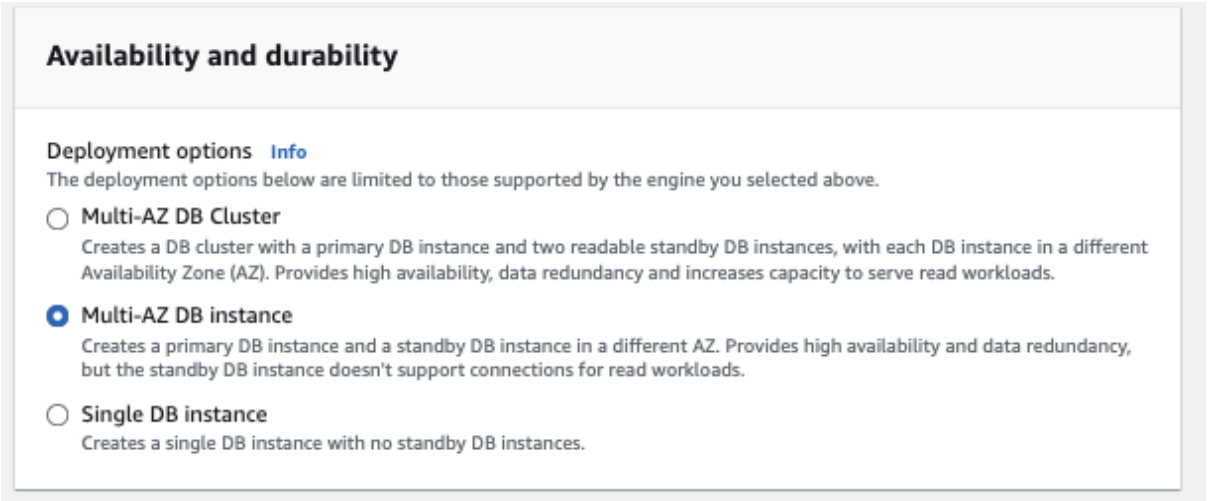


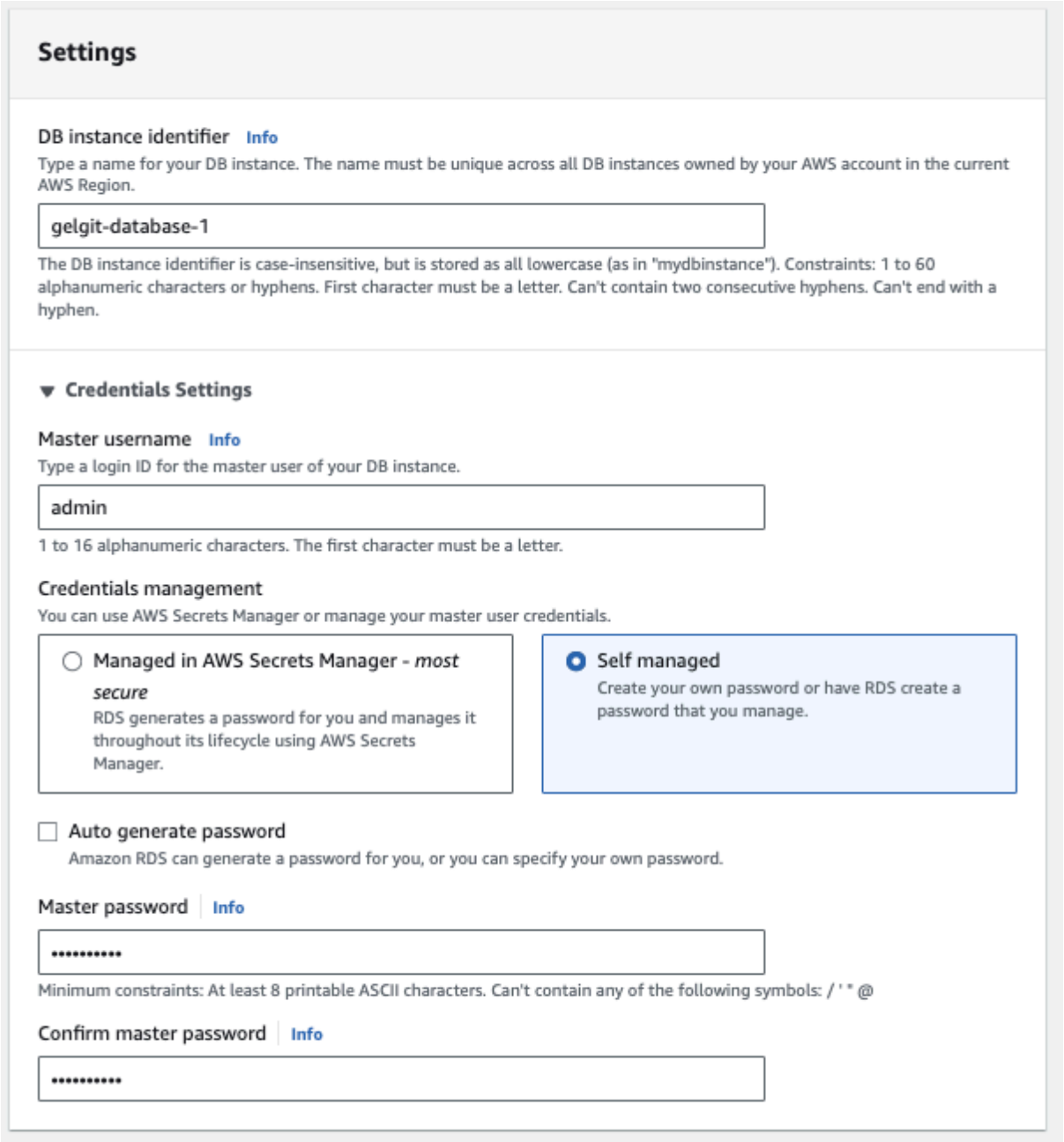
Step 8 – Create your Aurora or RDS

To create your database, click on Databases on the left hand navigation bar and click the Create Database button.
We're going to use Standard Create.
For Engine Type, select Amazon Aurora and tick the radio box that says Amazon Aurora PostgreSQL-Compatible Edition. Note that you can use another database type here if you'd like, or even the Serverless variation of Aurora as well – all will work.
For templates, select Dev/Test.
For Master Password, input a secure password.
Change your Instance Type to something small like a db.t3.medium or better yet, use serverless.
Under the Connectivity section, make the following changes.
Select your VPC.
Change the DB Subnet Group to be private-subnet-group which is the one we created in the previous step.
For Public Access, click No.
For Security Group, click Create New and name it rds-aurora-security-group. Finally, change the availability zone to 1a (the same AZ as our jump box from earlier steps).
Below is an image of my Connectivity settings.




Edition
MySQL Community
Known issues/limitations
Review the Known issues/limitations to learn about potential compatibility issues with specific database versions.
Engine version Info
View the engine versions that support the following database features.
Hide filters
Show versions that support the Multi-AZ DB cluster Info
Create a A Multi-AZ DB cluster with one primary DB instance and two readable standby DB instances. Multi-AZ DB clusters provide up to 2x faster transaction commit latency and automatic failover in typically under 35 seconds.
Show versions that support the Amazon RDS Optimized Writes Info
Amazon RDS Optimized Writes improves write throughput by up to 2x at no additional cost.
Engine Version
MySQL 8.0.34
Templates
Choose a sample template to meet your use case.
Production
Use defaults for high availability and fast, consistent performance.
Dev/Test
This instance is intended for development use outside of a production environment.
Free tier
Use RDS Free Tier to develop new applications, test existing applications, or gain hands-on experience with Amazon RDS. Info


Availability and durability
Deployment options Info
The deployment options below are limited to those supported by the engine you selected above.
Multi-AZ DB Cluster
Creates a DB cluster with a primary DB instance and two readable standby DB instances, with each DB instance in a different Availability Zone (AZ). Provides high availability, data redundancy and increases capacity to serve read workloads.
Multi-AZ DB instance
Creates a primary DB instance and a standby DB instance in a different AZ. Provides high availability and data redundancy, but the standby DB instance doesn't support connections for read workloads.
Single DB instance
Creates a single DB instance with no standby DB instances.



Master Username: admin
Master Password: Master2024

Instance configuration

The DB instance configuration options below are limited to those supported by the engine that you selected above.

DB instance class Info

▼ Hide filters

Show instance classes that support Amazon RDS Optimized Writes Info

Amazon RDS Optimized Writes improves write throughput by up to 2x at no additional cost.

Include previous generation classes

- ○ Standard classes (includes m classes)
- ○ Memory optimized classes (includes r and x classes)
- ◉ Burstable classes (includes t classes)

db.t3.micro
2 vCPUs 1 GiB RAM Network: 2,085 Mbps ▼

Storage

Storage type Info

Provisioned IOPS SSD (io2) storage volumes are now available.

General Purpose SSD (gp3)
Performance scales independently from storage

Allocated storage Info

20 GiB

Minimum: 20 GiB. Maximum: 6,144 GiB

After you modify the storage for a DB instance, the status of the DB instance will be in storage-optimization. Your instance will remain available as the storage-optimization operation completes.
Learn more

▶ **Advanced settings**

Baseline IOPS of 3,000 IOPS and storage throughput of 125 MiBps are included for allocated storage less than 400 GiB.

▼ **Storage autoscaling**

Storage autoscaling Info

Provides dynamic scaling support for your database's storage based on your application's needs.

☑ Enable storage autoscaling

Enabling this feature will allow the storage to increase after the specified threshold is exceeded.

Maximum storage threshold Info

Charges will apply when your database autoscales to the specified threshold

100 GiB

The minimum value is 22 GiB and the maximum value is 6,144 GiB

Connectivity Info

Compute resource

Choose whether to set up a connection to a compute resource for this database. Setting up a connection will automatically change connectivity settings so that the compute resource can connect to this database.

- (●) **Don't connect to an EC2 compute resource**
  Don't set up a connection to a compute resource for this database. You can manually set up a connection to a compute resource later.
- ( ) **Connect to an EC2 compute resource**
  Set up a connection to an EC2 compute resource for this database.

Network type Info

To use dual-stack mode, make sure that you associate an IPv6 CIDR block with a subnet in the VPC you specify.

- (●) **IPv4**
  Your resources can communicate only over the IPv4 addressing protocol.
- ( ) **Dual-stack mode**
  Your resources can communicate over IPv4, IPv6, or both.

Virtual private cloud (VPC) Info

Choose the VPC. The VPC defines the virtual networking environment for this DB instance.

gelgit-vpc (vpc-0525552ba87a21cf7)
4 Subnets, 2 Availability Zones

Only VPCs with a corresponding DB subnet group are listed.

ⓘ After a database is created, you can't change its VPC.

DB subnet group Info

Choose the DB subnet group. The DB subnet group defines which subnets and IP ranges the DB instance can use in the VPC that you selected.

db-private-subnet
2 Subnets, 2 Availability Zones

Public access Info

- ( ) **Yes**
  RDS assigns a public IP address to the database. Amazon EC2 instances and other resources outside of the VPC can connect to your database. Resources inside the VPC can also connect to the database. Choose one or more VPC security groups that specify which resources can connect to the database.
- (●) **No**
  RDS doesn't assign a public IP address to the database. Only Amazon EC2 instances and other resources inside the VPC can connect to your database. Choose one or more VPC security groups that specify which resources can connect to the database.

VPC security group (firewall) Info

Choose one or more VPC security groups to allow access to your database. Make sure that the security group rules allow the appropriate incoming traffic.

- ○ Choose existing
  Choose existing VPC security groups
- ◉ Create new
  Create new VPC security group

New VPC security group name

rds-mysql-sg

RDS Proxy

RDS Proxy is a fully managed, highly available database proxy that improves application scalability, resiliency, and security.

☐ Create an RDS Proxy Info

RDS automatically creates an IAM role and a Secrets Manager secret for the proxy. RDS Proxy has additional costs. For more information, see Amazon RDS Proxy pricing.

Certificate authority - *optional* Info

Using a server certificate provides an extra layer of security by validating that the connection is being made to an Amazon database. It does so by checking the server certificate that is automatically installed on all databases that you provision.

rds-ca-rsa2048-g1 (default)
Expiry: May 26, 2061

If you don't select a certificate authority, RDS chooses one for you.

▼ Additional configuration

Database port Info

TCP/IP port that the database will use for application connections.

3306

Tags

A tag consists of a case-sensitive key-value pair.

No tags associated with the resource.

Add new tag

You can add up to 50 more tags.

Database authentication

Database authentication options Info

- (•) Password authentication
  Authenticates using database passwords.
- ( ) Password and IAM database authentication
  Authenticates using the database password and user credentials through AWS IAM users and roles.
- ( ) Password and Kerberos authentication
  Choose a directory in which you want to allow authorized users to authenticate with this DB instance using Kerberos Authentication.

Monitoring

- [x] Enable Enhanced Monitoring
  Enabling Enhanced Monitoring metrics are useful when you want to see how different processes or threads use the CPU.

Granularity

10 seconds

Monitoring Role

default

Clicking "Create database" will authorize RDS to create the IAM role rds-monitoring-role

Additional configuration

Database options, encryption turned on, backup turned on, backtrack turned off, maintenance, CloudWatch Logs, delete protection turned off.

Database options

Initial database name Info

gelgit_database

If you do not specify a database name, Amazon RDS does not create a database.

DB parameter group Info

default.mysql8.0

Option group Info

default:mysql-8-0

Backup

Backup

Enable automated backups

Creates a point-in-time snapshot of your database

⚠ Please note that automated backups are currently supported for InnoDB storage engine only. If you are using MyISAM, refer to details here.

Backup retention period Info

The number of days (1-35) for which automatic backups are kept.

10 days

Backup window Info

The daily time range (in UTC) during which RDS takes automated backups.

- (•) Choose a window
- ( ) No preference

Start time

00 : 00 UTC

Duration

0.5 hours

Copy tags to snapshots

Backup replication Info

Enable replication in another AWS Region

Enabling replication automatically creates backups of your DB instance in the selected Region, for disaster recovery, in addition to the current Region.

Encryption

Encryption

☑ Enable encryption

Choose to encrypt the given instance. Master key IDs and aliases appear in the list after they have been created using the AWS Key Management Service console. Info

AWS KMS key Info

(default) aws/rds

Account

637423226272

KMS key ID

alias/aws/rds

Log exports

Select the log types to publish to Amazon CloudWatch Logs

- ☑ Audit log
- ☑ Error log
- ☑ General log
- ☑ Slow query log

IAM role

The following service-linked role is used for publishing logs to CloudWatch Logs.

RDS service-linked role

Maintenance

Auto minor version upgrade Info

☑ Enable auto minor version upgrade

Enabling auto minor version upgrade will automatically upgrade to new minor versions as they are released. The automatic upgrades occur during the maintenance window for the database.

Maintenance window Info

Select the period you want pending modifications or maintenance applied to the database by Amazon RDS.

- ○ Choose a window
- ◉ No preference

Deletion protection

☐ Enable deletion protection

Protects the database from being deleted accidentally. While this option is enabled, you can't delete the database.

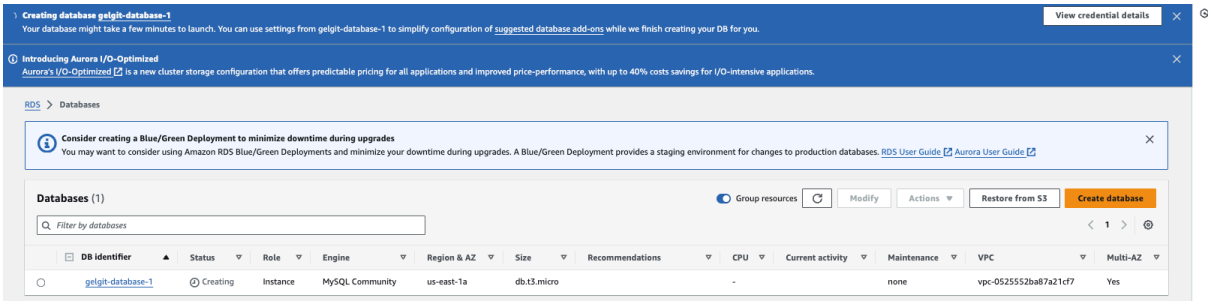
Estimated Monthly costs

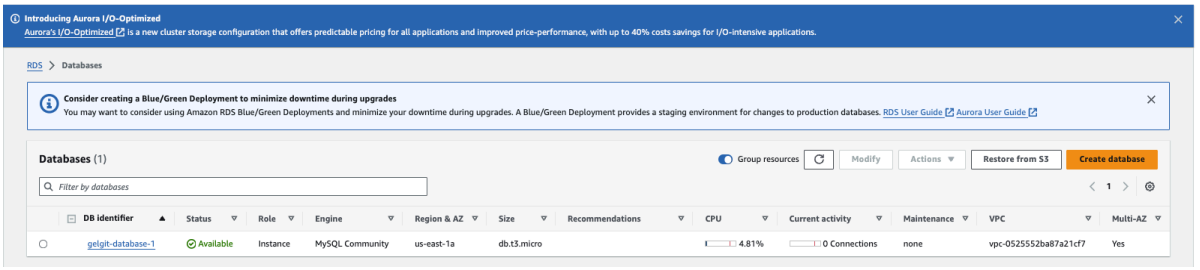
DB instance	24.82 USD
Storage	4.60 USD
Total	**29.42 USD**

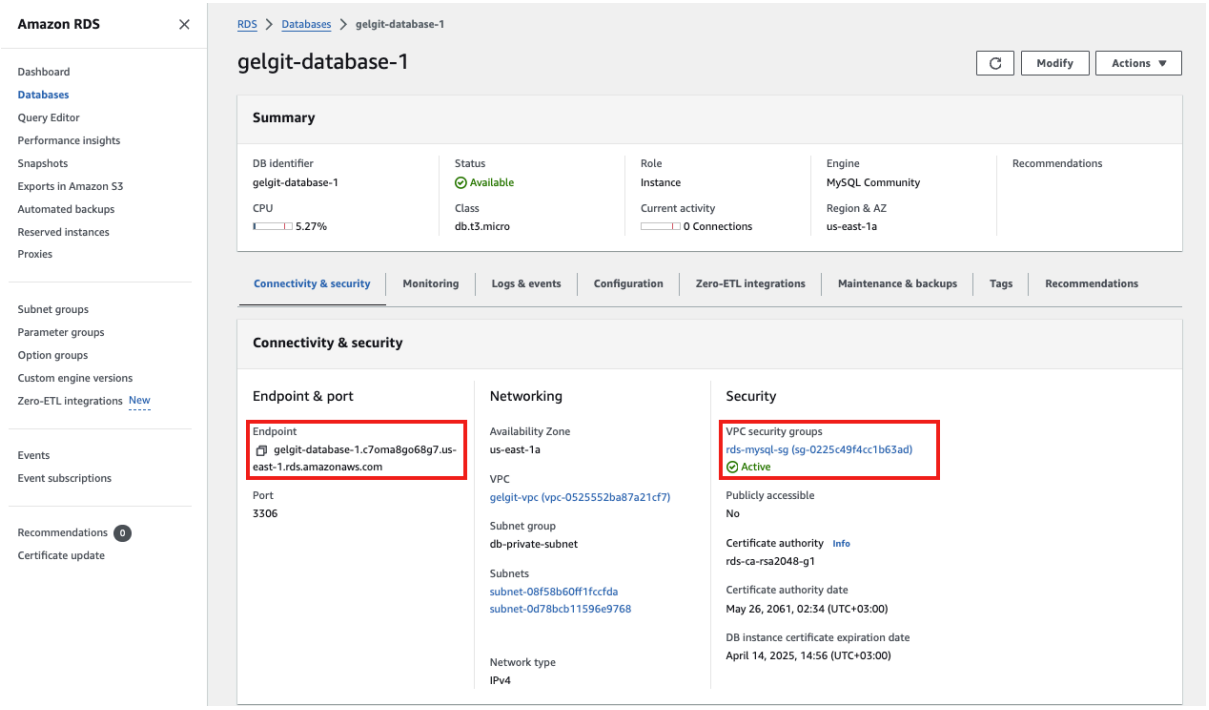
This billing estimate is based on on-demand usage as described in Amazon RDS Pricing. Estimate does not include costs for backup storage, IOs (if applicable), or data transfer.

Estimate your monthly costs for the DB Instance using the AWS Simple Monthly Calculator.

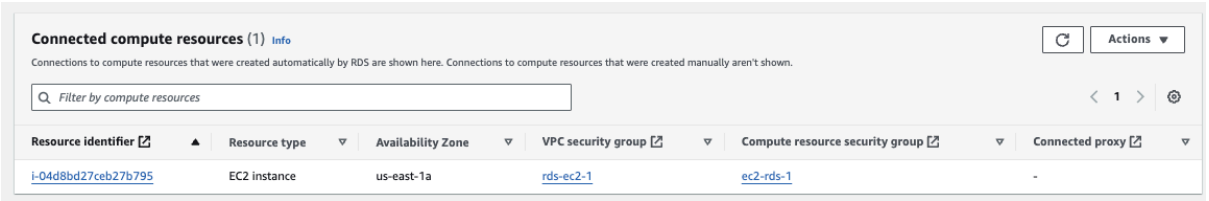
You are responsible for ensuring that you have all of the necessary rights for any third-party products or services that you use with AWS services.





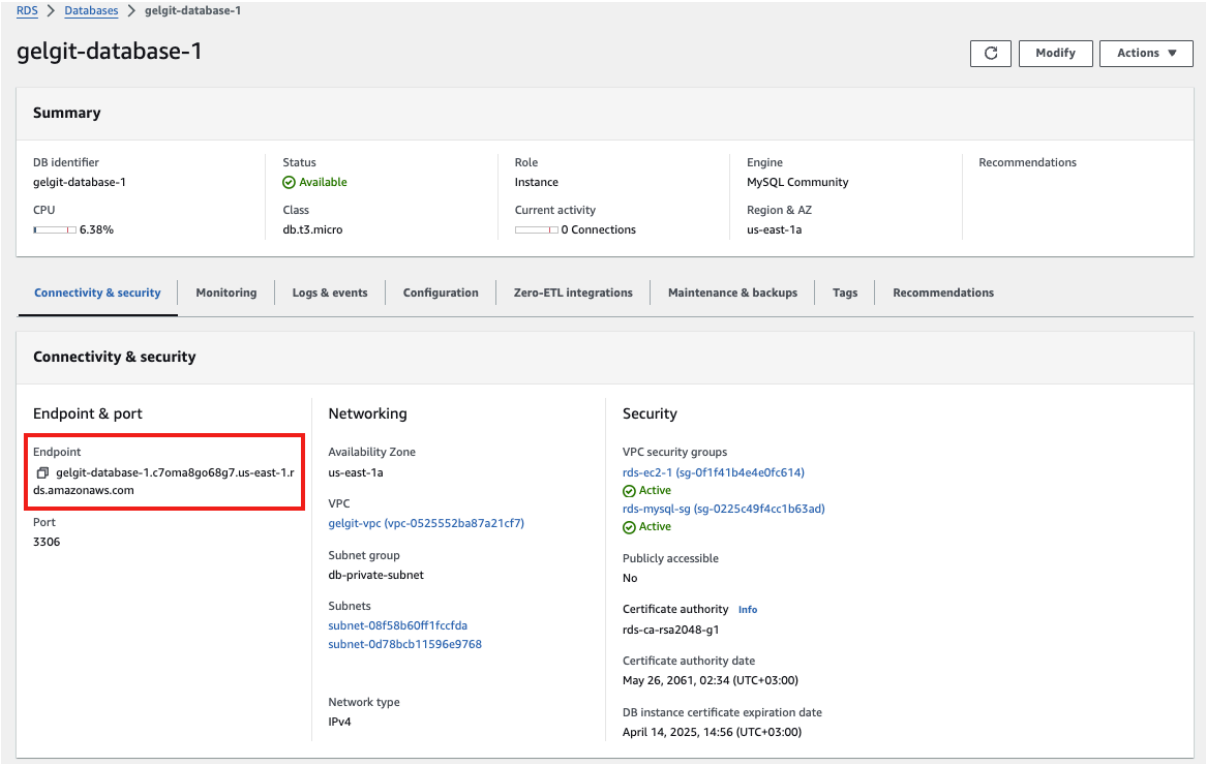


Connect DB to Compute Resources



Connect to a MySQL DB instance

1) Find the endpoint (DNS name) and port number for your DB instance.



DB Endpoint:
gelgit-database-1.c7oma8go68g7.us-east-1.rds.amazonaws.com

DB Port:
3306

2) Connect to the EC2 instance that you created earlier by following the steps in Connect to your Linux instance in the Amazon EC2 User Guide for Linux Instances.

We recommend that you connect to your EC2 instance using SSH. If the SSH client utility is installed on Windows, Linux, or Mac, you can connect to the instance using the following command format:

```
ssh -i location_of_pem_file ec2-user@ec2-instance-public-dns-name
```

For example, assume that ec2-database-connect-key-pair.pem is stored in /dir1 on Linux, and the public IPv4 DNS for your EC2 instance is ec2-12-345-678-90.compute-1.amazonaws.com. Your SSH command would look as follows:

```
ssh -i /dir1/ec2-database-connect-key-pair.pem
ec2-user@ec2-12-345-678-90.compute-1.amazonaws.com
```

For our specific case, you can go to the EC2 instance and select it. After that click on the "connect" button. New page shows you the way of connections. Select SSH client connection and run the given command after locating your private key file in your computer.

In our case:

```
ssh -i "jumb-box-rds-keypair.pem" ec2-user@ec2-54-172-200-114.compute-1.amazonaws.com
```

3) Get the latest bug fixes and security updates by updating the software on your EC2 instance. To do this, use the following command. (The -y option installs the updates without asking for confirmation. To examine updates before installing, omit this option.)

```
sudo dnf update -y
```

4) To install the mysql command-line client from MariaDB on Amazon Linux 2023, run the following command:

```
sudo dnf install mariadb105
```

5) Substitute the DB instance endpoint (DNS name) for endpoint, and substitute the master username that you used for admin. Provide the master password that you used when prompted for a password.

```
mysql -h <endpoint> -P 3306 -u <admin> -p
```

In our project

```
mysql -h gelgit-database-1.c7oma8go68g7.us-east-1.rds.amazonaws.com -P 3306 -u admin -p
```

After you enter the password for the user, you should see output similar to the following.

```
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 3082
Server version: 8.0.28 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)]>
```

For more information about connecting to a MySQL DB instance, see Connecting to a DB instance running the MySQL database engine. If you can't connect to your DB instance, see Can't connect to Amazon RDS DB instance.

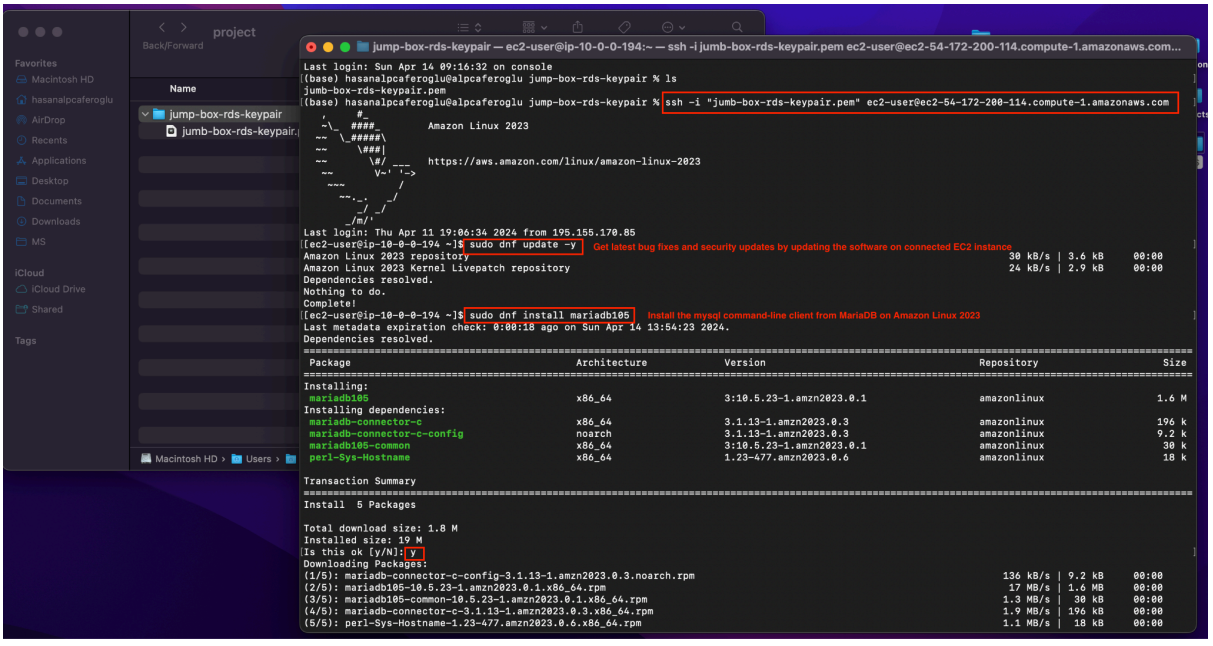
For security, it is a best practice to use encrypted connections. Only use an unencrypted MySQL connection when the client and server are in the same VPC and the network is trusted. For information about using encrypted connections, see Connecting from the MySQL command-line client with SSL/TLS (encrypted).

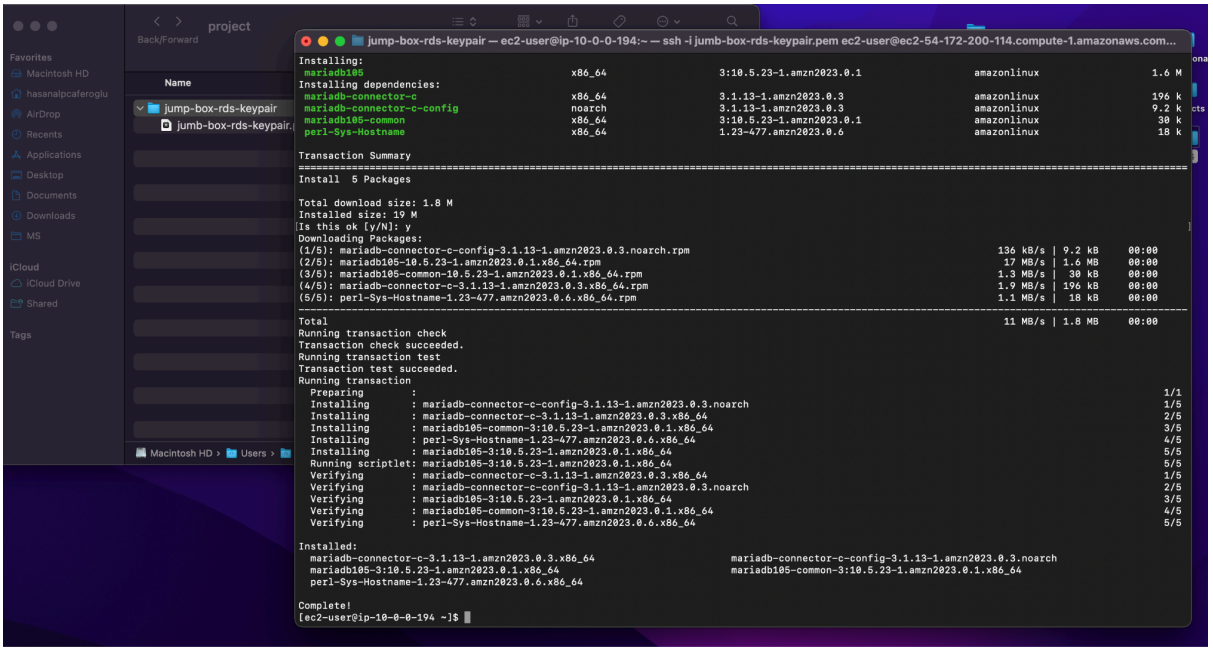
6) Run SQL commands.

For example, the following SQL command shows the current date and time:

```
SELECT CURRENT_TIMESTAMP;
```

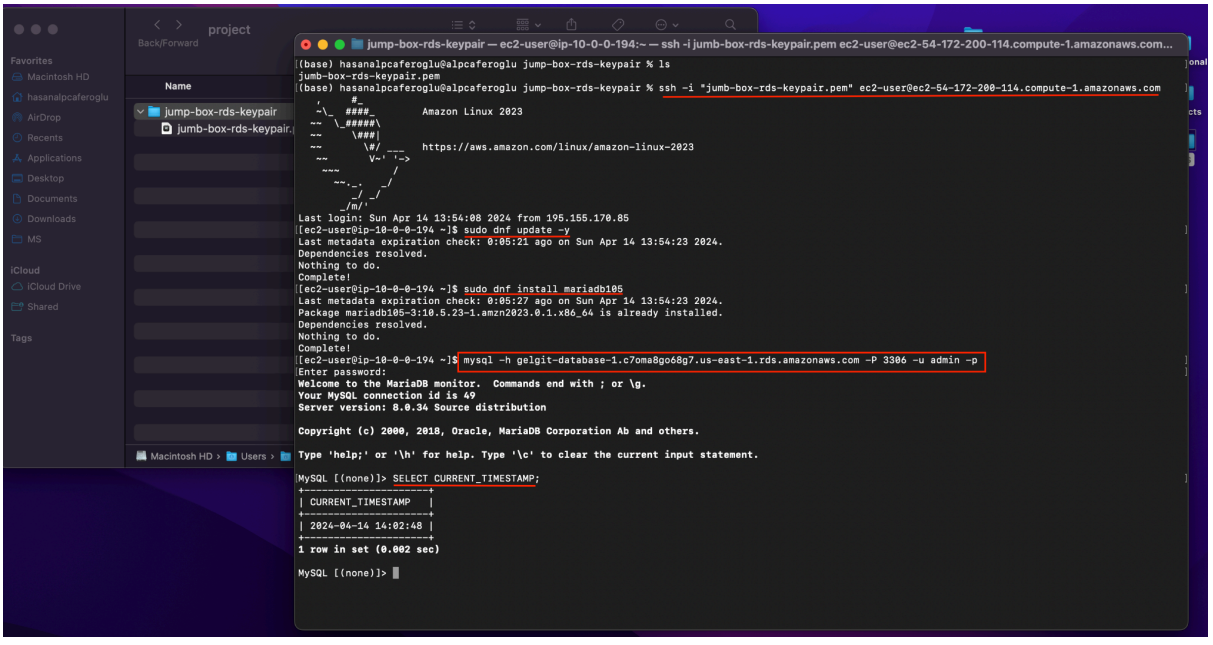
Below you see the steps for connecting MySQL database through EC2 instance.

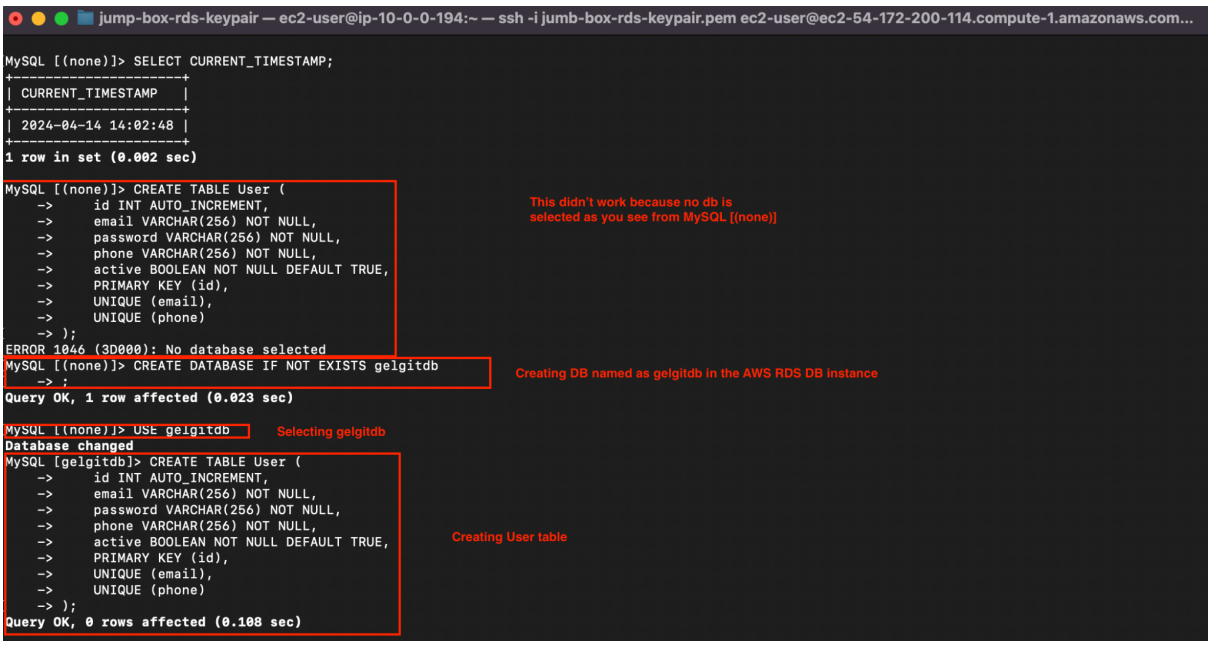


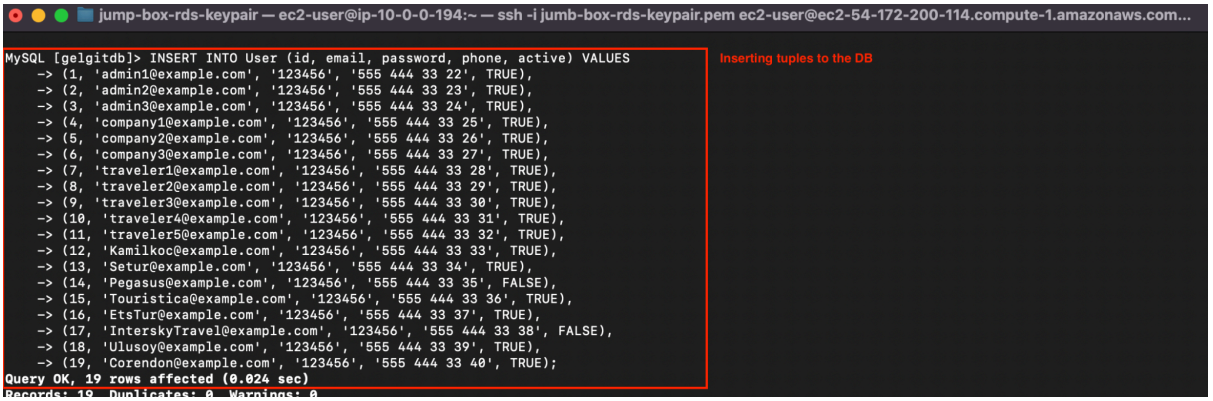


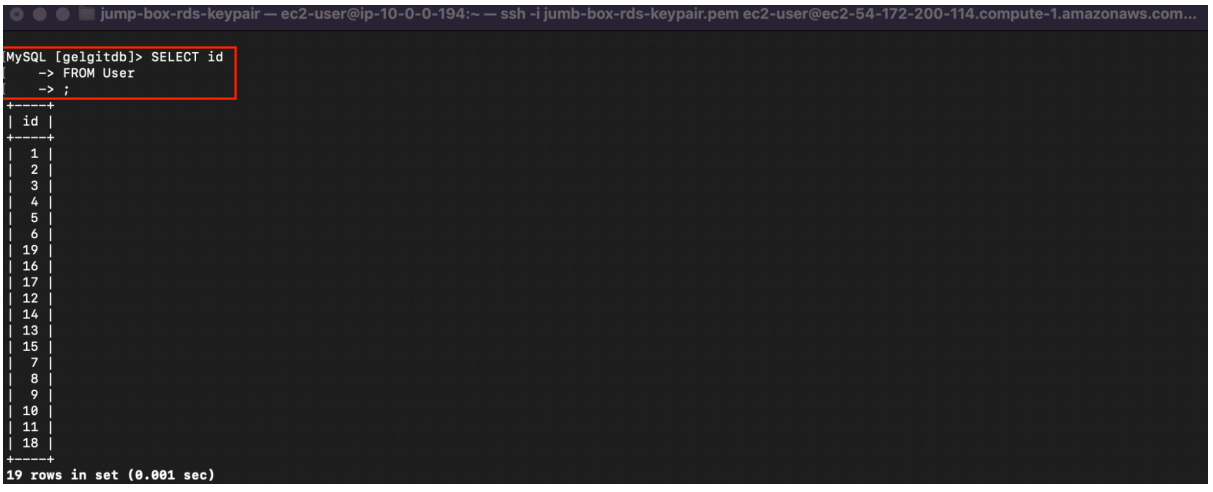

```
mysql -h gelgit-database-1.c7oma8go68g7.us-east-1.rds.amazonaws.com -P 3306 -u admin -p
```

Database Master Username: admin
Database Master password: Master2024




jump-box-rds-keypair — ec2-user@ip-10-0-0-194:~ — ssh -i jumb-box-rds-keypair.pem ec2-user@ec2-54-172-200-114.compute-1.amazonaws.com...
MySQL [(none)]> SELECT CURRENT_TIMESTAMP;
+---------------------+
| CURRENT_TIMESTAMP |
+---------------------+
| 2024-04-14 14:02:48 |
+---------------------+
1 row in set (0.002 sec)
MySQL [(none)]> CREATE TABLE User (
-> id INT AUTO_INCREMENT,
-> email VARCHAR(256) NOT NULL,
-> password VARCHAR(256) NOT NULL,
-> phone VARCHAR(256) NOT NULL,
-> active BOOLEAN NOT NULL DEFAULT TRUE,
-> PRIMARY KEY (id),
-> UNIQUE (email),
-> UNIQUE (phone)
->);
ERROR 1046 (3D000): No database selected
This didn't work because no db is selected as you see from MySQL [(none)]
MySQL [(none)]> CREATE DATABASE IF NOT EXISTS gelgitdb
-> ;
Creating DB named as gelgitdb in the AWS RDS DB instance
Query OK, 1 row affected (0.023 sec)
MySQL [(none)]> USE gelgitdb
Selecting gelgitdb
Database changed
MySQL [gelgitdb]> CREATE TABLE User (
-> id INT AUTO_INCREMENT,
-> email VARCHAR(256) NOT NULL,
-> password VARCHAR(256) NOT NULL,
-> phone VARCHAR(256) NOT NULL,
-> active BOOLEAN NOT NULL DEFAULT TRUE,
-> PRIMARY KEY (id),
-> UNIQUE (email),
-> UNIQUE (phone)
->);
Creating User table
Query OK, 0 rows affected (0.108 sec)


jump-box-rds-keypair — ec2-user@ip-10-0-0-194:~ — ssh -i jumb-box-rds-keypair.pem ec2-user@ec2-54-172-200-114.compute-1.amazonaws.com...
MySQL [gelgitdb]> INSERT INTO User (id, email, password, phone, active) VALUES
-> (1, 'admin1@example.com', '123456', '555 444 33 22', TRUE),
-> (2, 'admin2@example.com', '123456', '555 444 33 23', TRUE),
-> (3, 'admin3@example.com', '123456', '555 444 33 24', TRUE),
-> (4, 'company1@example.com', '123456', '555 444 33 25', TRUE),
-> (5, 'company2@example.com', '123456', '555 444 33 26', TRUE),
-> (6, 'company3@example.com', '123456', '555 444 33 27', TRUE),
-> (7, 'traveler1@example.com', '123456', '555 444 33 28', TRUE),
-> (8, 'traveler2@example.com', '123456', '555 444 33 29', TRUE),
-> (9, 'traveler3@example.com', '123456', '555 444 33 30', TRUE),
-> (10, 'traveler4@example.com', '123456', '555 444 33 31', TRUE),
-> (11, 'traveler5@example.com', '123456', '555 444 33 32', TRUE),
-> (12, 'Kamilkoc@example.com', '123456', '555 444 33 33', TRUE),
-> (13, 'Setur@example.com', '123456', '555 444 33 34', TRUE),
-> (14, 'Pegasus@example.com', '123456', '555 444 33 35', FALSE),
-> (15, 'Touristica@example.com', '123456', '555 444 33 36', TRUE),
-> (16, 'EtsTur@example.com', '123456', '555 444 33 37', TRUE),
-> (17, 'InterskyTravel@example.com', '123456', '555 444 33 38', FALSE),
-> (18, 'Ulusoy@example.com', '123456', '555 444 33 39', TRUE),
-> (19, 'Corendon@example.com', '123456', '555 444 33 40', TRUE);
Inserting tuples to the DB
Query OK, 19 rows affected (0.024 sec)


jump-box-rds-keypair — ec2-user@ip-10-0-0-194:~ — ssh -i jumb-box-rds-keypair.pem ec2-user@ec2-54-172-200-114.compute-1.amazonaws.com...
MySQL [gelgitdb]> SELECT id
-> FROM User
-> ;
+----+
| id |
+----+
| 1 |
| 2 |
| 3 |
| 4 |
| 5 |
| 6 |
| 19 |
| 16 |
| 17 |
| 12 |
| 14 |
| 13 |
| 15 |
| 7 |
| 8 |
| 9 |
| 10 |
| 11 |
| 18 |
+----+
19 rows in set (0.001 sec)

```
MySQL [gelgitdb]> SELECT * FROM User;
+----+---------------------------+----------+---------------+--------+
| id | email                     | password | phone         | active |
+----+---------------------------+----------+---------------+--------+
|  1 | admin1@example.com        | 123456   | 555 444 33 22 |      1 |
|  2 | admin2@example.com        | 123456   | 555 444 33 23 |      1 |
|  3 | admin3@example.com        | 123456   | 555 444 33 24 |      1 |
|  4 | company1@example.com      | 123456   | 555 444 33 25 |      1 |
|  5 | company2@example.com      | 123456   | 555 444 33 26 |      1 |
|  6 | company3@example.com      | 123456   | 555 444 33 27 |      1 |
|  7 | traveler1@example.com     | 123456   | 555 444 33 28 |      1 |
|  8 | traveler2@example.com     | 123456   | 555 444 33 29 |      1 |
|  9 | traveler3@example.com     | 123456   | 555 444 33 30 |      1 |
| 10 | traveler4@example.com     | 123456   | 555 444 33 31 |      1 |
| 11 | traveler5@example.com     | 123456   | 555 444 33 32 |      1 |
| 12 | Kamilkoc@example.com      | 123456   | 555 444 33 33 |      1 |
| 13 | Setur@example.com         | 123456   | 555 444 33 34 |      1 |
| 14 | Pegasus@example.com       | 123456   | 555 444 33 35 |      0 |
| 15 | Touristica@example.com    | 123456   | 555 444 33 36 |      1 |
| 16 | EtsTur@example.com        | 123456   | 555 444 33 37 |      1 |
| 17 | InterskyTravel@example.com | 123456   | 555 444 33 38 |      0 |
| 18 | Ulusoy@example.com        | 123456   | 555 444 33 39 |      1 |
| 19 | Corendon@example.com      | 123456   | 555 444 33 40 |      1 |
+----+---------------------------+----------+---------------+--------+
19 rows in set (0.002 sec)
```

Useful Commands for MySQL Server

```
// show the list of databases available in your MySQL server
SHOW DATABASES;

// Switch to the database you want to work with
USE <database_name>;

// switch to the no database
USE ;

// delete a database
DROP DATABASE <database_name>;

// create a database
CREATE DATABASE IF NOT EXISTS <database_name>;
```

```
// view tables within the database
SHOW TABLES;

//  To see the structure of a specific table. This will show you the columns and their data types.
DESCRIBE <table_name>;
DESC <table_name>;

//information about a table, including its size, number of rows
SHOW TABLE STATUS LIKE <table_name>;

// view a list of MySQL server system variables
SHOW VARIABLES;

// display information about the MySQL storage engines and their status
SHOW ENGINE STATUS;
```

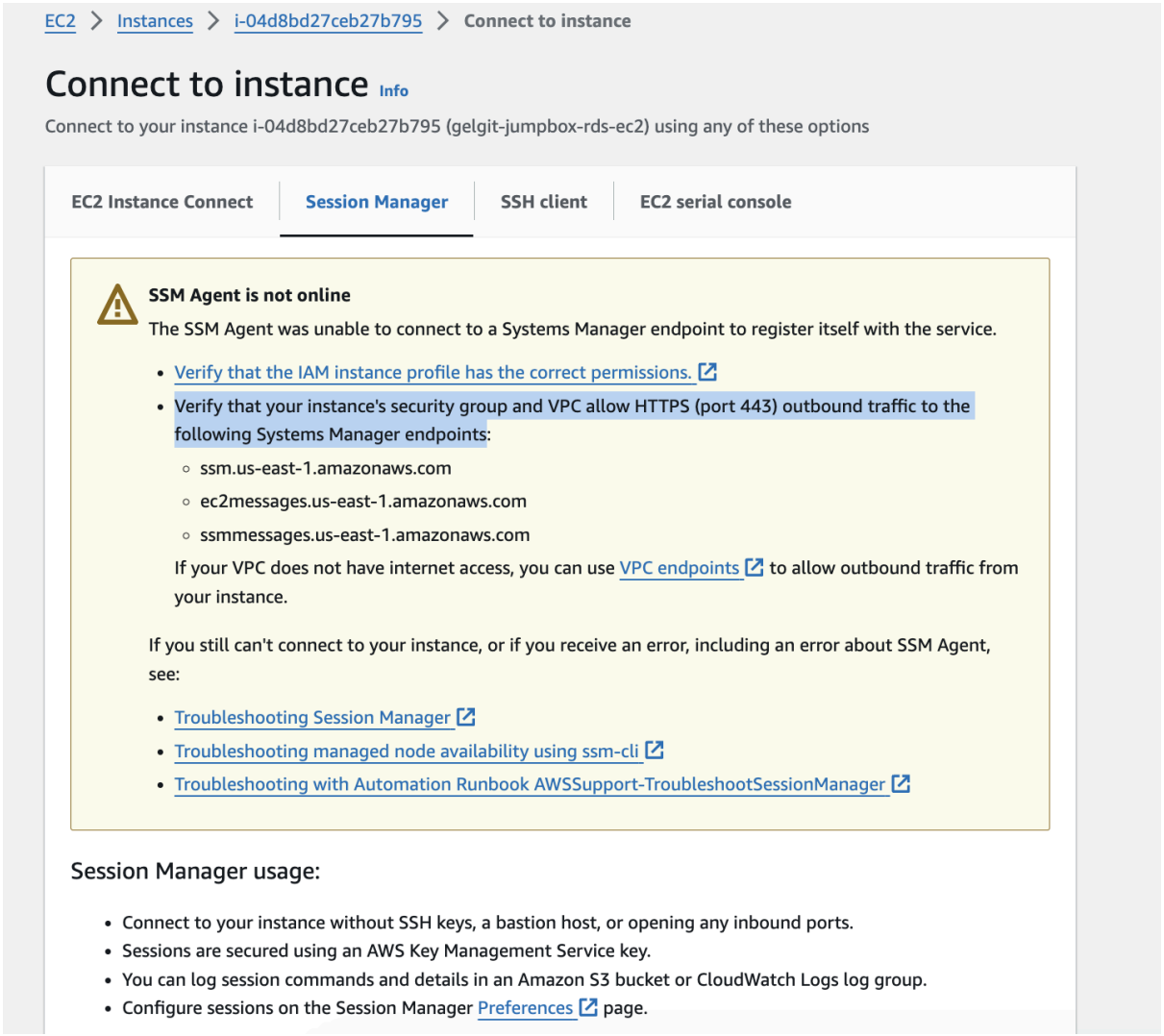
```
//  To see the views in your MySQL database;
// This query retrieves the names of all views in the specified database.
SHOW FULL TABLES IN <database_name> WHERE TABLE_TYPE LIKE 'VIEW';
// or
SELECT TABLE_SCHEMA, TABLE_NAME FROM information_schema.VIEWS WHERE
TABLE_SCHEMA LIKE 'your_db_name';
```

SHOW FULL TABLES IN gelgitdb WHERE TABLE_TYPE LIKE 'VIEW';

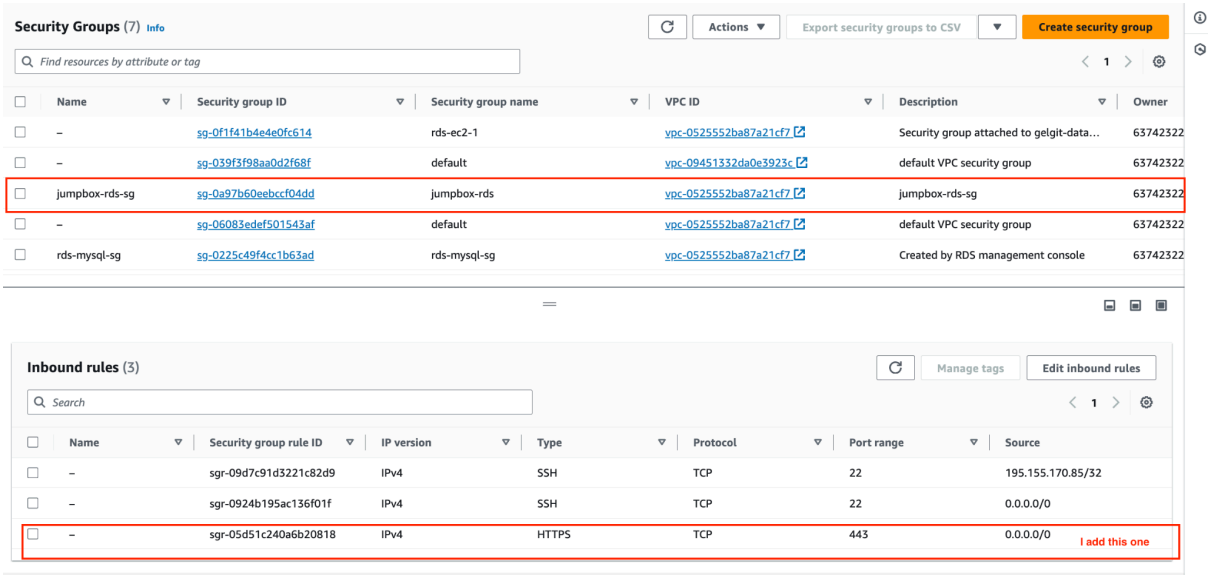
Connecting EC2 with Session Manager

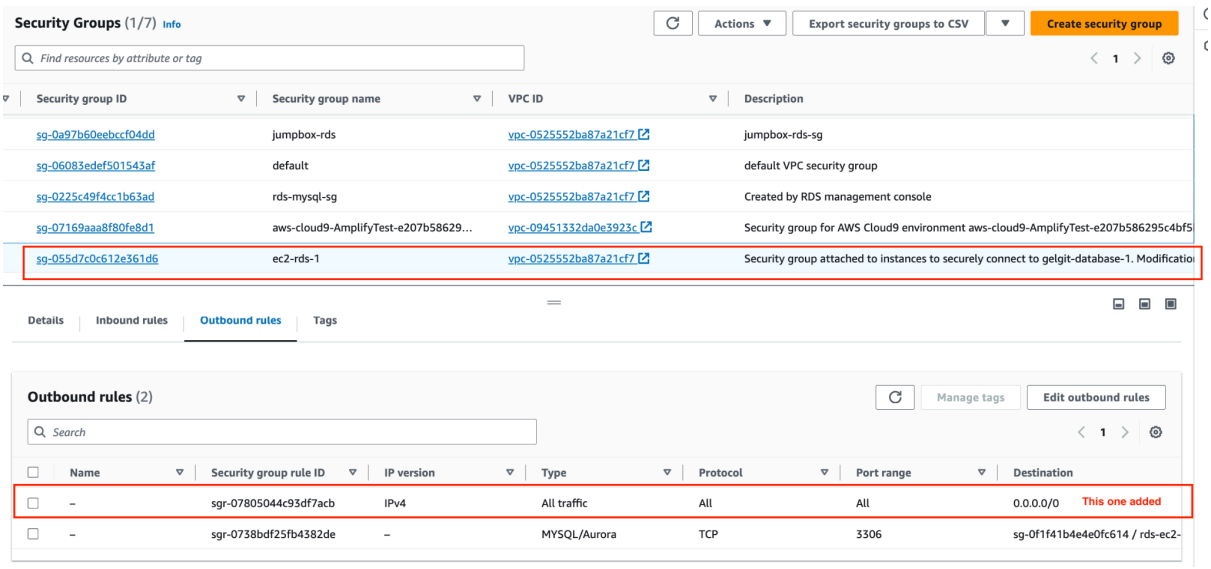
- Do the steps in homework 1. First you need to create a system manager role from IAM.
    - IAM → roles → create roles → trusted entity type: AWS service → use case: EC2 → next → add permissions: select AmazonSSMManagedInstanceCore → role name: SSMInstanceProfile → create role
- Attach the Systems Manager instance profile to an existing instance
    - EC2 console → choose ec2 instance → actions → security → modify IAM rule → IAM role: select SSMInstanceProfile → update IAM role
- Now it is time for connection using session manager
    - EC2 console → select ec2 → Connect → session manager

After these I couldn't connect with session manager. Messages below are taken

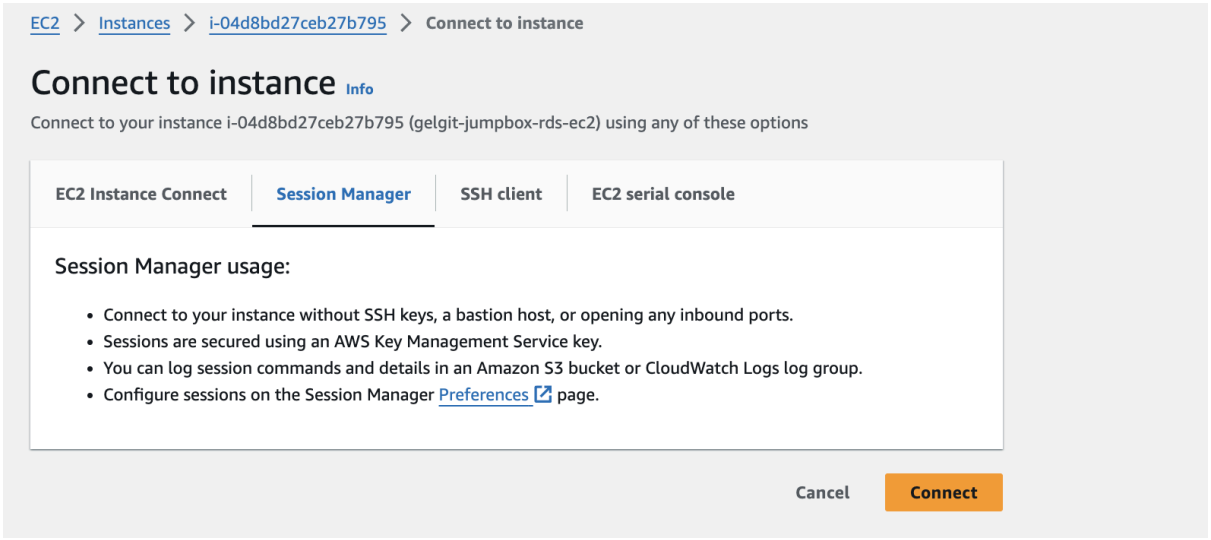


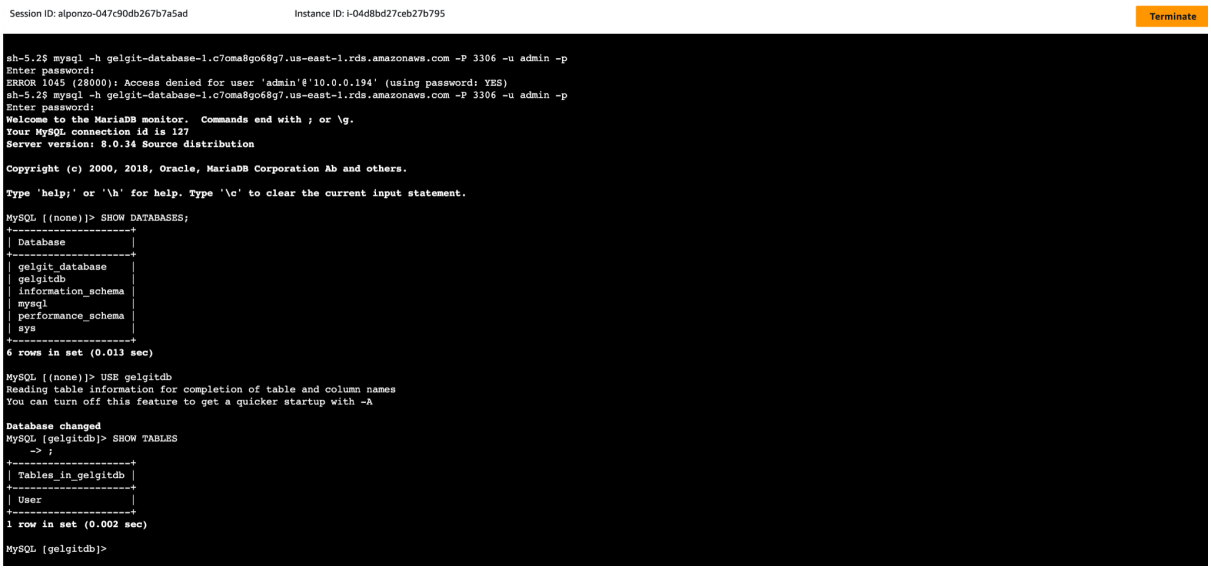
To overcome this issue, I made changes below in security groups.





After these changes,



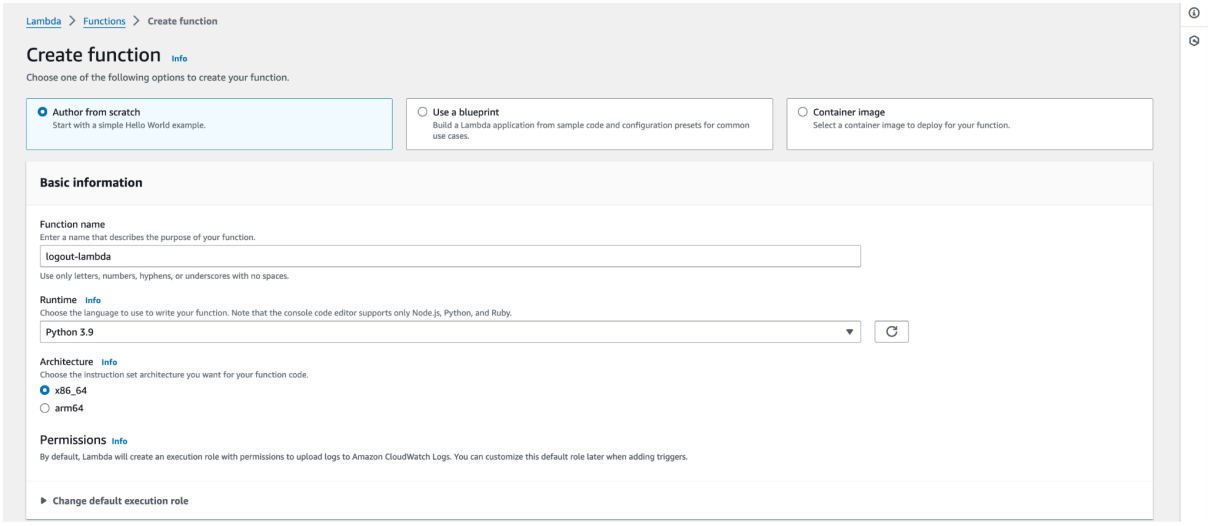


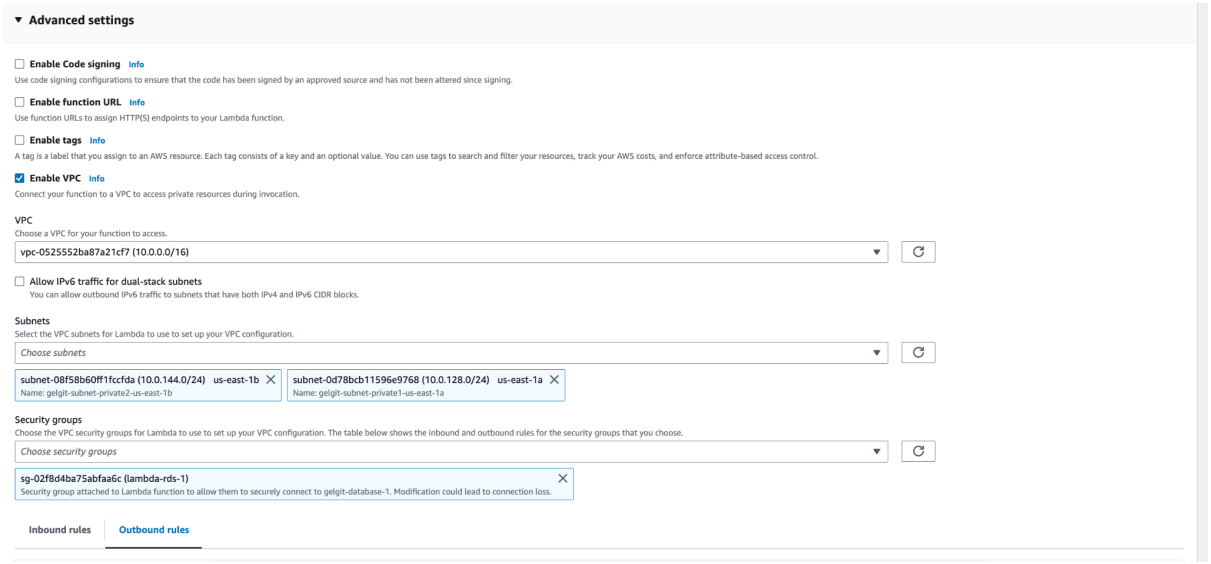
```
mysql -h gelgit-database-1.c7oma8go68g7.us-east-1.rds.amazonaws.com -P 3306 -u admin -p
```

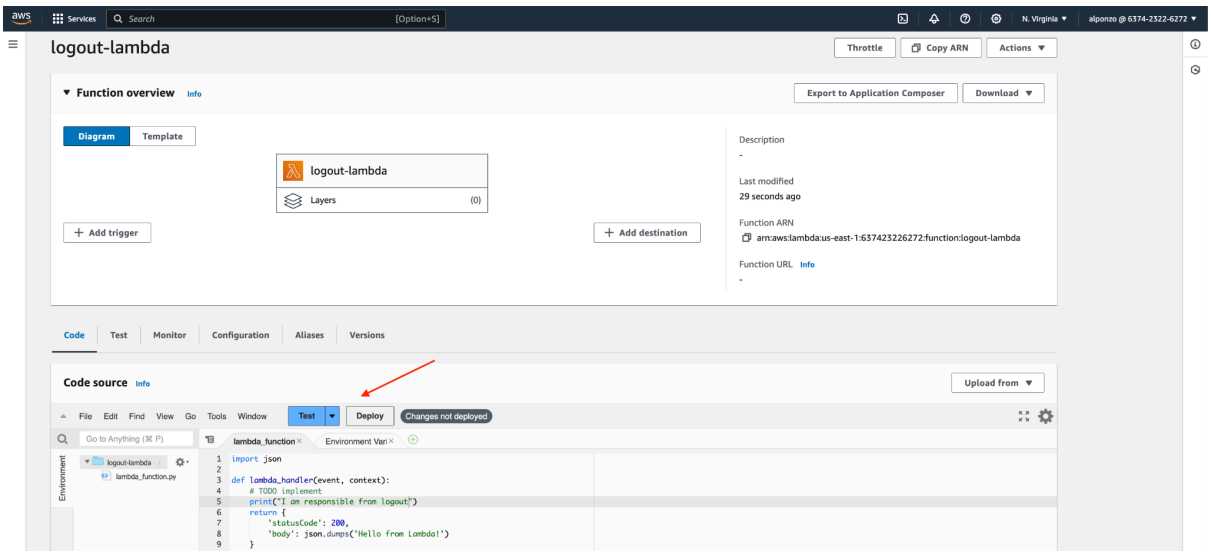
Database Master Username: admin
Database Master password: Master2024

Creating Lambda and Connecting it with API Gateway

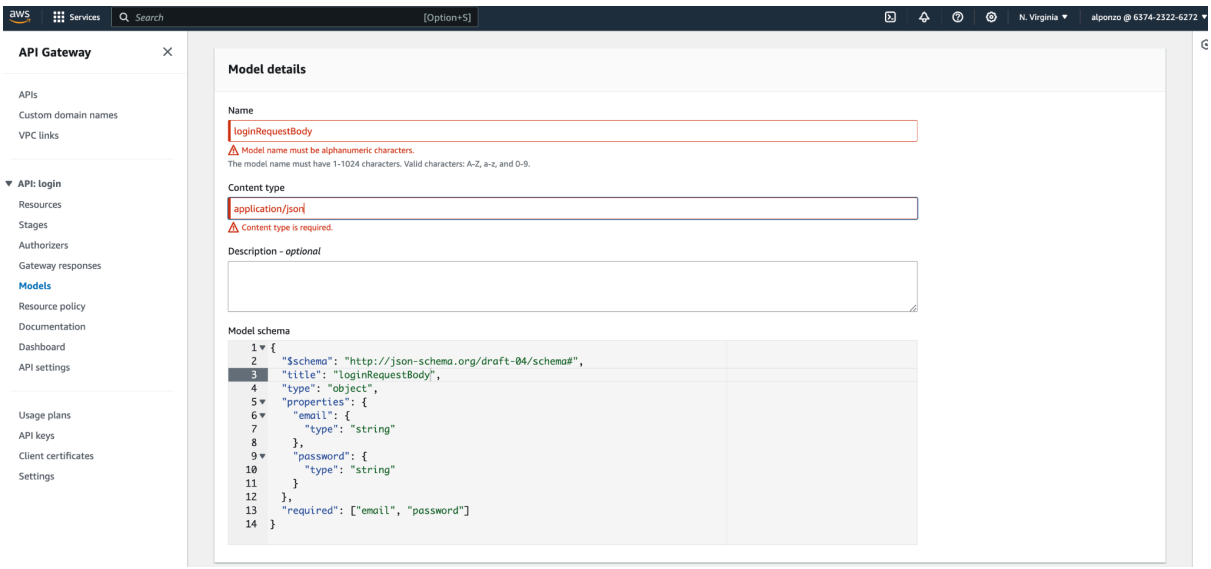
1) Create a lambda function
    a) enter lambda function name
    b) select runtime
    c) set advanced settings (for example in our project, we need to enable VPC: select vpc, select subnets for lambda, choose security group)
    d) Create lambda
        i) Don't forget to select VPC (gelgit-vpc), two private subnet and security group named as "lambda-rds-1"
    e) For the created lambda, you can change the code but you should deploy it.



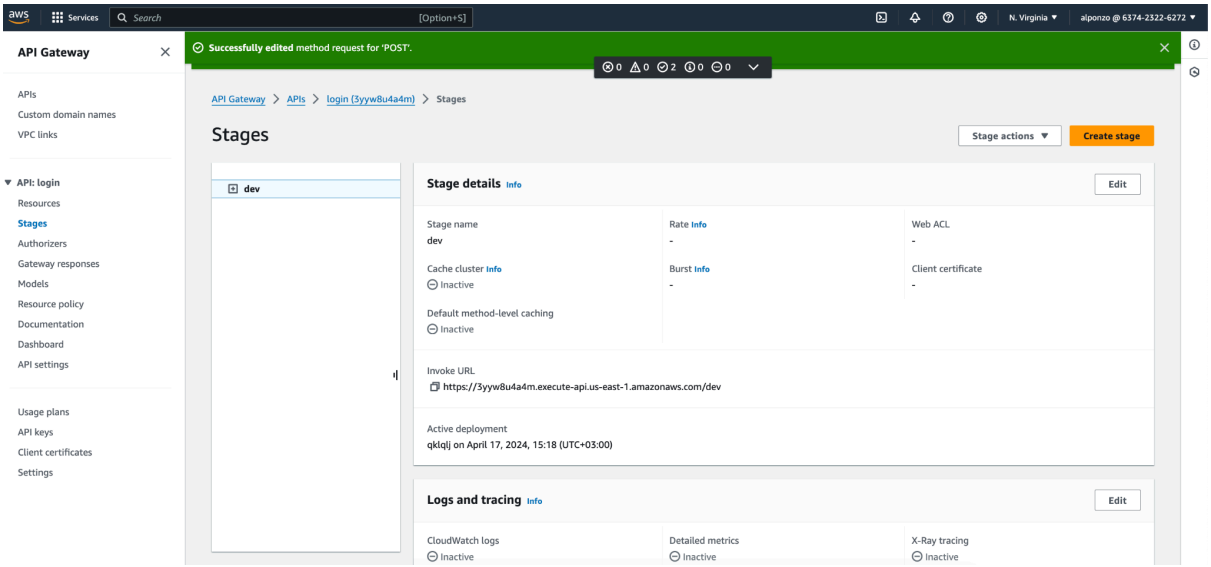


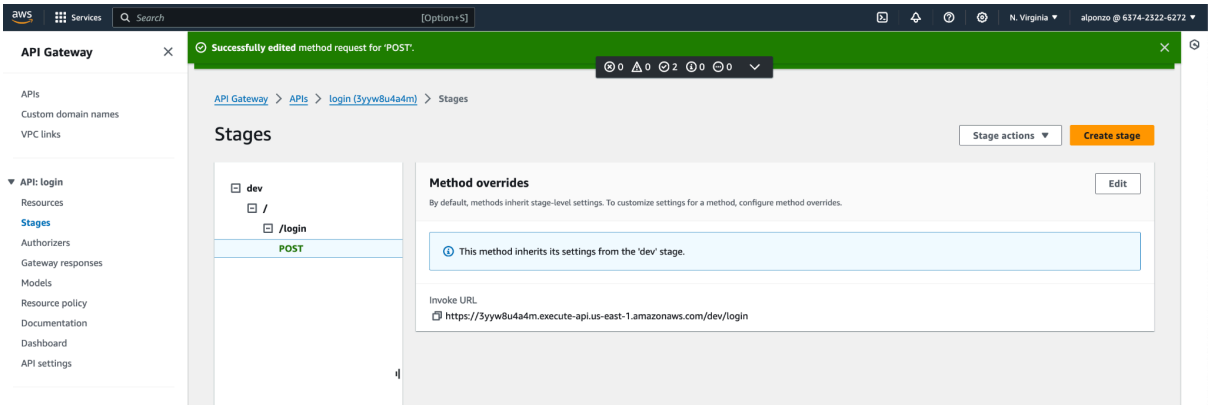


2) You may need to create a model. Go to API Gateway console and go to the model bar.
    a) Create a model
    b) Give it a name
    c) Write content type (application/json)
    d) Fill Model schema

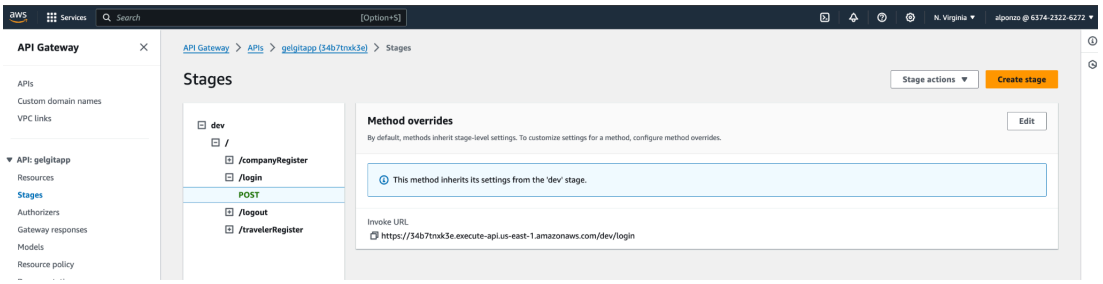


3) Navigate to the Resources bar in API Gateway.
    a) Click on Create resource
    b) Select resource path and write resource name, then create it.
4) Now, we need to create a method for that resource.
    a) Click on the created resource
    b) Click on create method
    c) Select method type
    d) Select lambda function
    e) Select Method request settings
        i) You can select AWS Cognito if you have it or you can edit the method after creating AWS Cognito
    f) Select Request body if needed from the models you created.

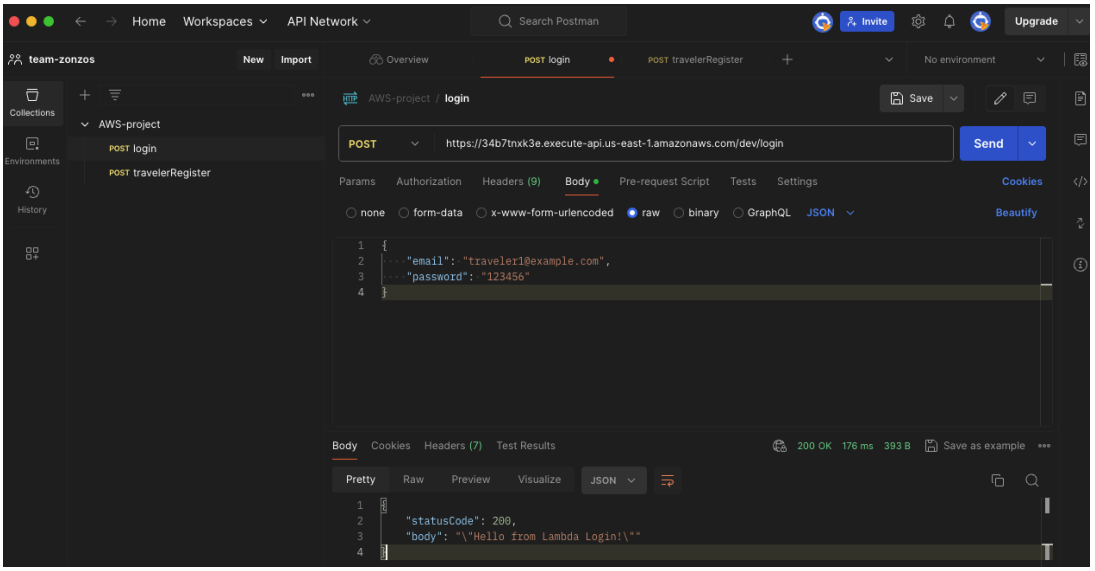




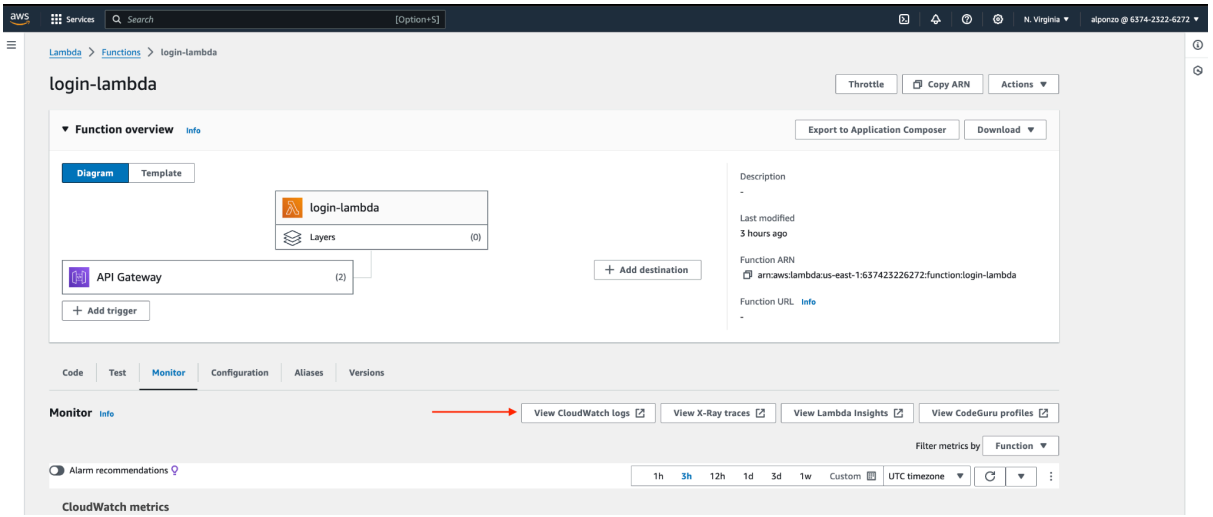
5) Deploy API to a selected stage
    a) You can select a *New Stage* and name it as you wish
6) Navigate to Stages bar from the API Gateway
    a) Select the stage
    b) Select the resource
    c) Select the method you want to invoke
    d) Copy the Invoke URL

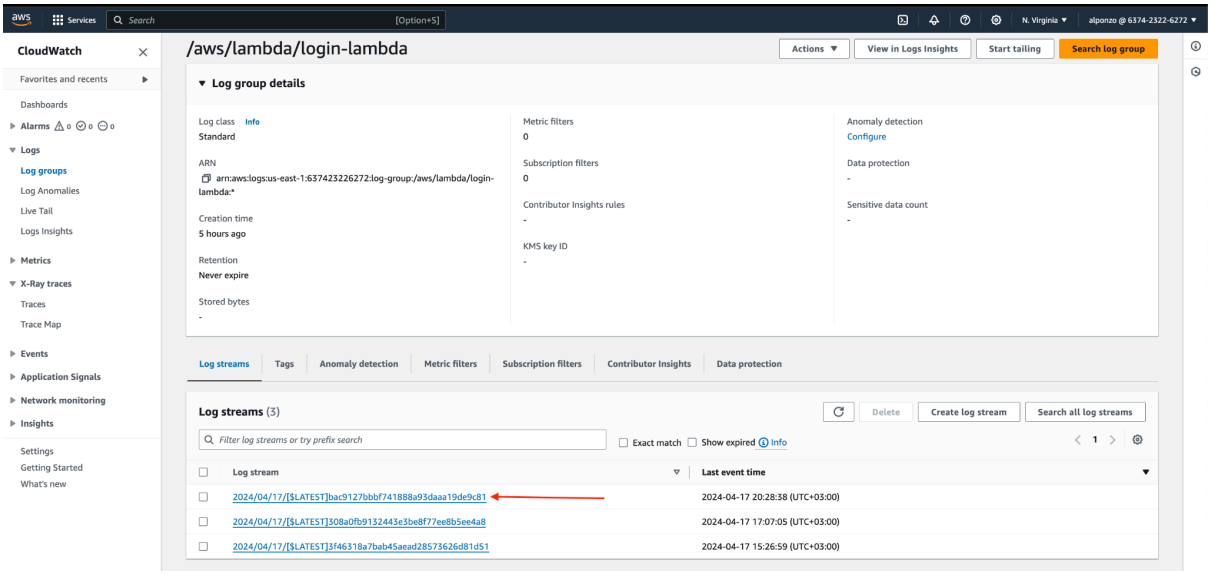


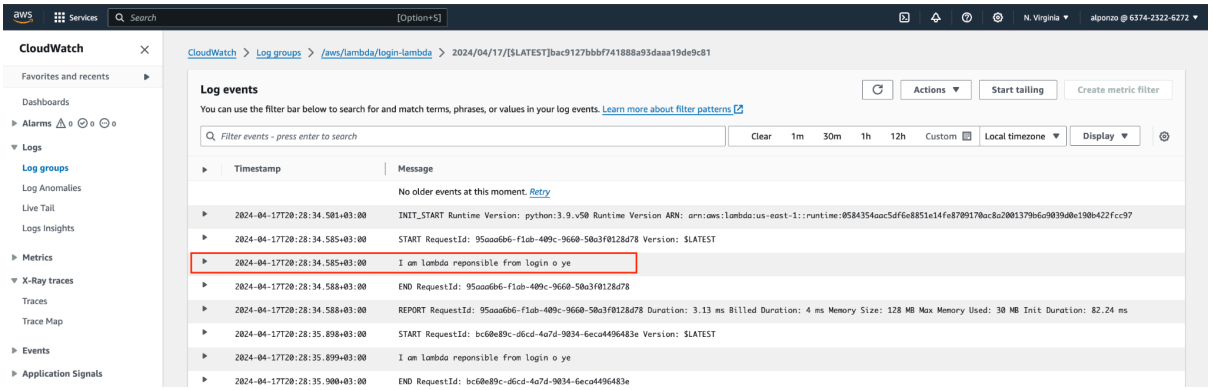
7) Create a request from postman using method's invoke url and observe



8) You can monitor Lambda Function Logs using CloudWatch logs. If you have print in your lambda function, you can see these prints using CloudWath







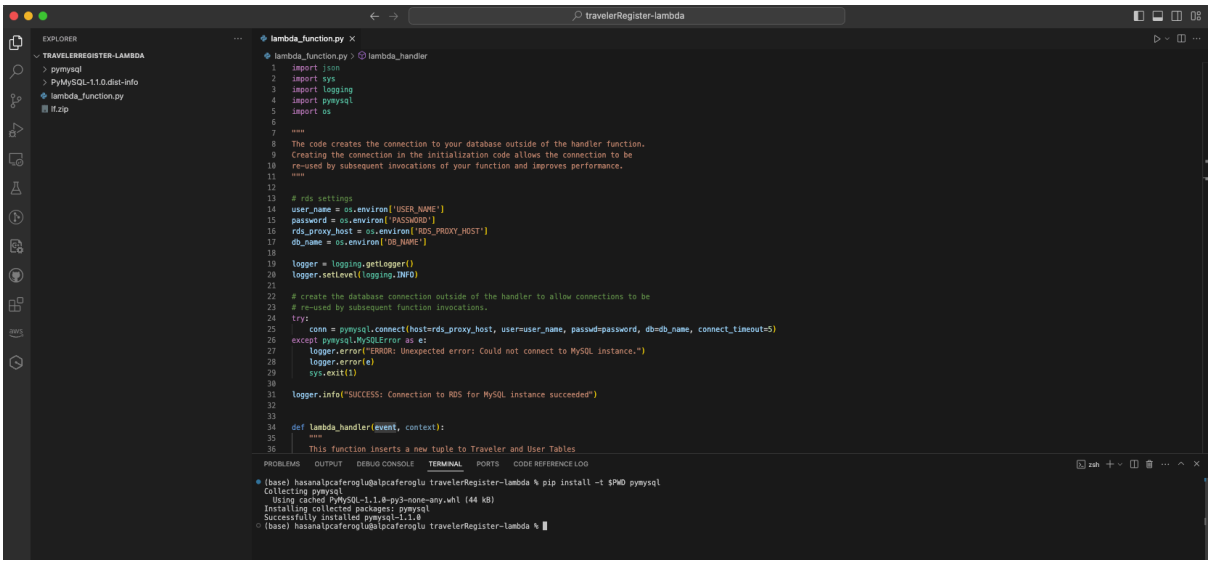
Example Lambda Function:

```
import json
import sys
import logging
import pymysql
import os

"""
The code creates the connection to your database outside of the handler function.
Creating the connection in the initialization code allows the connection to be
re-used by subsequent invocations of your function and improves performance.
"""

# rds settings
user_name = os.environ['USER_NAME']
password = os.environ['PASSWORD']
rds_proxy_host = os.environ['RDS_PROXY_HOST']
db_name = os.environ['DB_NAME']

logger = logging.getLogger()
logger.setLevel(logging.INFO)

# create the database connection outside of the handler to allow connections to be
# re-used by subsequent function invocations.
try:
    conn = pymysql.connect(host=rds_proxy_host, user=user_name, passwd=password,
db=db_name, connect_timeout=5)
except pymysql.MySQLError as e:
    logger.error("ERROR: Unexpected error: Could not connect to MySQL instance.")
    logger.error(e)
    sys.exit(1)

logger.info("SUCCESS: Connection to RDS for MySQL instance succeeded")


def lambda_handler(event, context):
    """
    This function inserts a new tuple to Traveler and User Tables
    """
    # message = event['Records'][0]['body']
    # data = json.loads(message)
    data = event
    email = data['email']
    password = data['password']
    passwordRepeat = data['passwordRepeat']
    phone = data['phone']
    TCK = data['TCK']
    name = data['name']
    surname = data['surname']
    age = data['age']
```

```
    item_count = 0
    sql_string = f"insert into Traveler (id, TCK, name, surname, age, balance) values(%s, %s, %s,
%s, %s, %s)"

    with conn.cursor() as cursor:

        cursor.execute('SELECT * FROM User WHERE email = %s ', (email, ))
        accountExistWithSameEmail = cursor.fetchone()

        cursor.execute('SELECT * FROM User WHERE phone = %s ', (phone, ))
        accountExistWithSamePhone = cursor.fetchone()

        cursor.execute('SELECT * FROM Traveler WHERE TCK = %s ', (TCK, ))
        accountExistWithSameTCK = cursor.fetchone()

        if not email or not password or not phone or not TCK or not name or not surname or not age:
            message = 'Please fill out the form!'
        elif accountExistWithSameEmail:
            message = 'There is already an account with that email!'
        elif accountExistWithSameTCK:
            message = 'There is already an account with this TCK!'
        elif accountExistWithSamePhone:
            message = 'There is already an account with this phone number!'
        elif password != passwordRepeat:
            message = 'Password mismatch. Please enter same password!'
        elif int(age) < 18:
            message = 'To register, you should be older than 18!'
        else:
            cursor.execute('INSERT INTO User (id, email, password, phone, active) VALUES (NULL,
% s, % s, % s, TRUE)', (email, password, phone, ))
            conn.commit()
            # find newly added traveler
            cursor.execute('SELECT * FROM User WHERE email = %s ', (email, ))
            newUser = cursor.fetchone()
            # get the user id and insert this user into Traveler
            newUserId = newUser[0] # index 0 corresponds to the id
            cursor.execute('INSERT INTO Traveler (id, TCK, name, surname, age, balance) VALUES
(%s, % s, % s, % s, %s, 0)', (newUserId, TCK, name, surname, age, ))
            conn.commit()
            message = 'User successfully created! Please log in.'

        logger.info(message)

    conn.commit()


    return {
        'statusCode': 200,
        'body': message
    }
```

Using a Lambda function to access an Amazon RDS database

Youtube Link: https://www.youtube.com/watch?v=vyLvmPkQZkI&t=612s
Tutorial link: https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/rds-lambda-tutorial.html

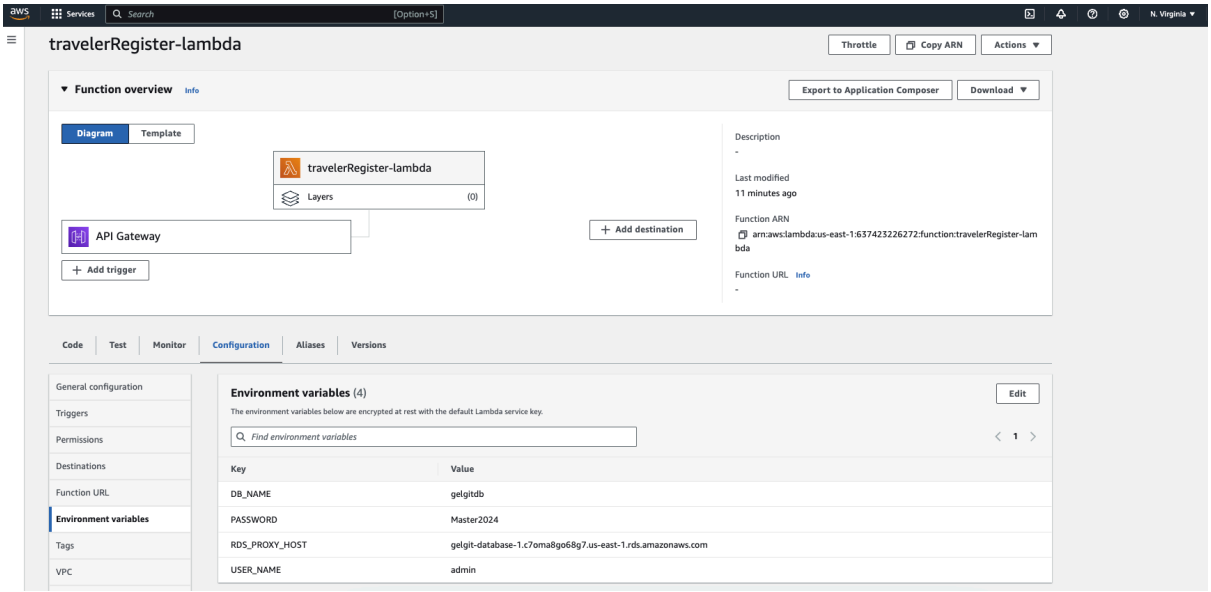


- Since we use pymysql package in our lambda, we need to instal required package
- So, in local, I open a folder and I write a lambda function. Also, I installed pymysql which creates package files. These are pymysql and PyMySQL-1.1.0.dist-info
- pip install -t $PWD pymysql
- After lambda_function.py file written and necessary package installed, zip these 3 files.

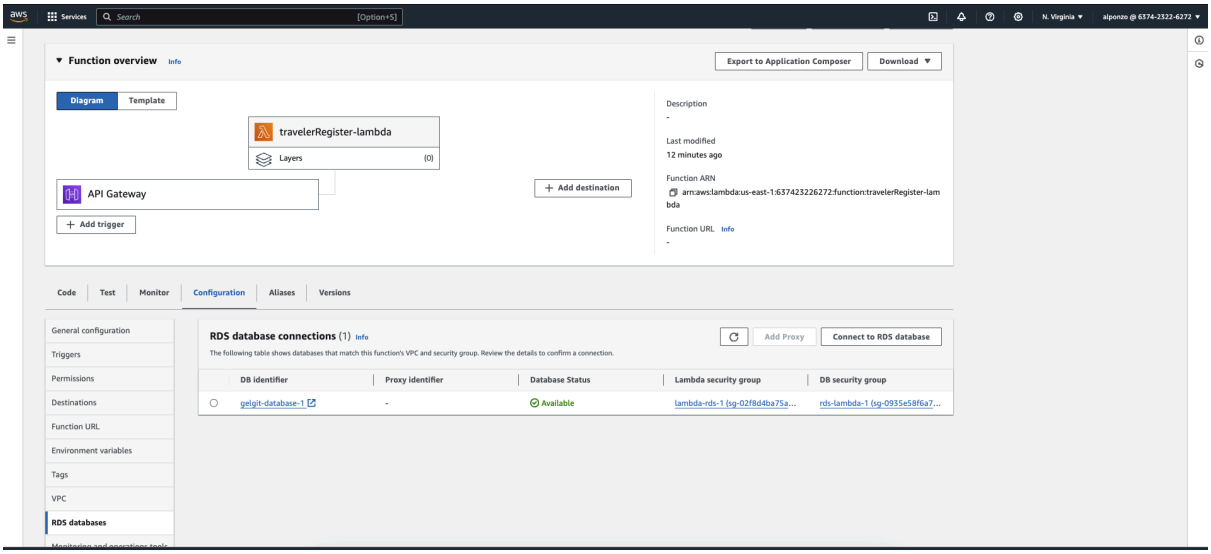
After that, click on "Up From" button and upload the zip file

Note that you have to set the environment variables from the configuration - environment variables.

Key	Value
DB_NAME	gelgitdb
PASSWORD	Master2024
RDS_PROXY_HOST	gelgit-database-1.c7oma8go68g7.us-east-1.rds.amazonaws.com
USER_NAME	admin

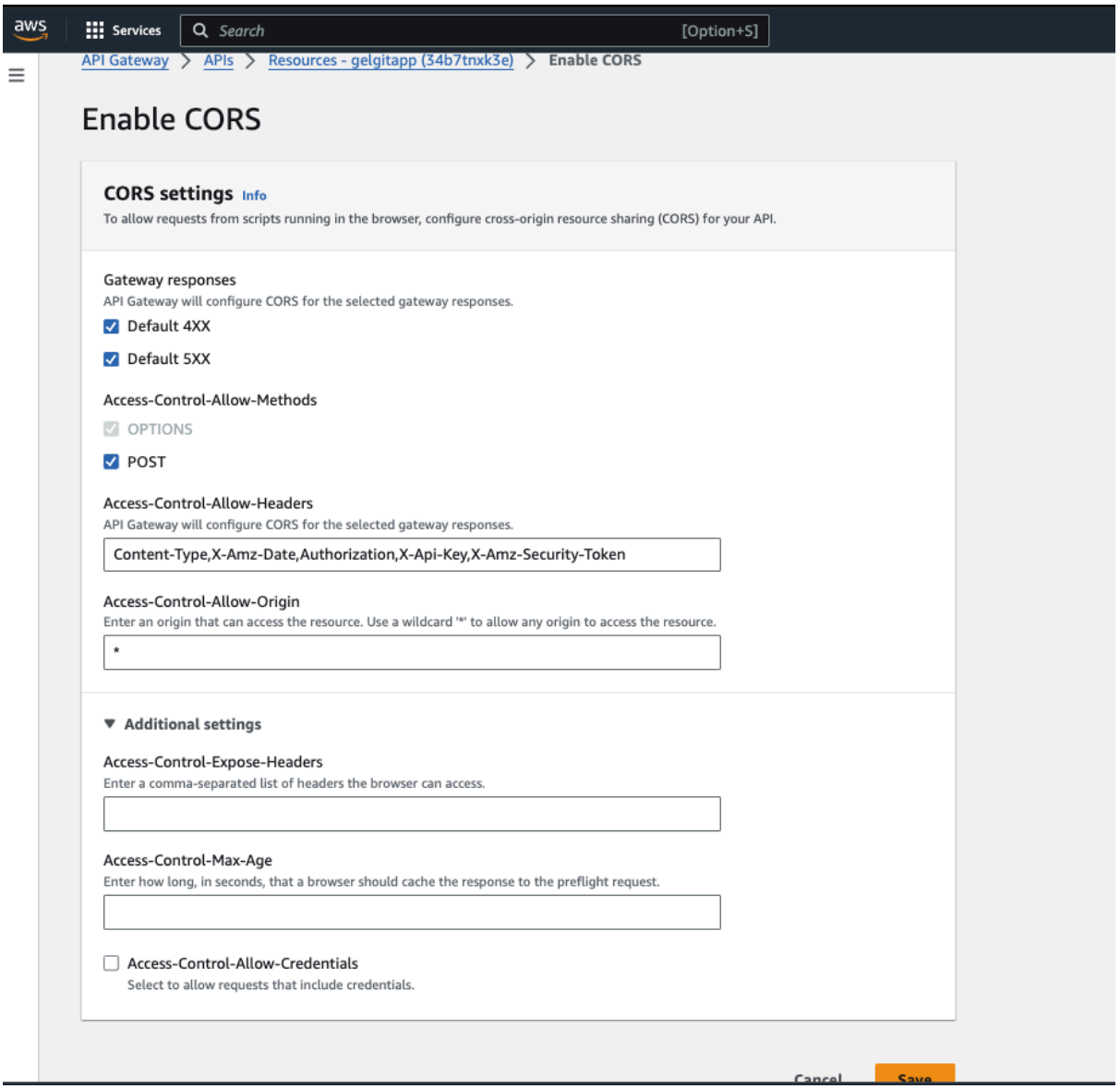


Also be sure that Lambda is connected with RDS. If not, create connection.

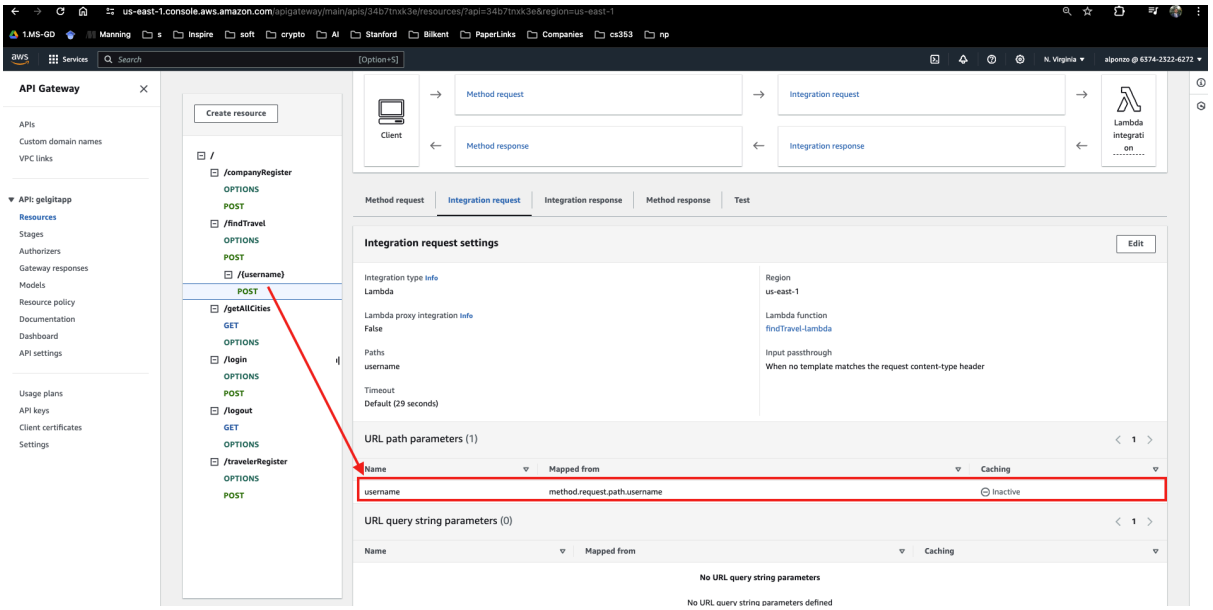


Enable CORS on a resource using the API Gateway console

https://docs.aws.amazon.com/apigateway/latest/developerguide/how-to-cors-console.html



Path Variable



https://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-create-api-step-by-step.html

Created API Gateway connected to Lambda Functions are as follows

- [x] login
- [x] logout
- [x] companyRegister
- [x] travelerRegister
- [ ] travels (@app.route('/travel/<string:vehicle_type>/from:<string:departure_city>/to:<string:arrival_city>/date:<string:departure_date>/extra_date:<string:extra_date>', methods=['GET']))
- [x] findTravel (@app.route('/findTravel', methods=['GET', 'POST']))
- [x] myTravels (@app.route('/myTravels', methods=['GET', 'POST']))
- [ ] makeComment(travel_id) (@app.route('/makeComment/<int:travel_id>', methods = ['GET', 'POST']))
- [ ] deleteComment(travel_id, traveler_id, company_id) (@app.route('/deleteComment/<int:travel_id>/<int:traveler_id>/<int:company_id>', methods = ['GET', 'POST']))
- [x] buy_travel(travel_id)(@app.route('/travel/buy/<int:travel_id>/', methods=['GET', 'POST']))

- [ ] coupons() @app.route('/coupons', methods=['GET', 'POST'])
- [ ] userProfile() @app.route('/userProfile', methods=['GET', 'POST'])
- [ ] updateTravelerProfile() @app.route('/updateTravelerProfile/', methods=['GET', 'POST'])
- [ ] balance() @app.route('/balance', methods = ['GET', 'POST'])
- [ ] journeys() @app.route('/journeys', methods = ['GET', 'POST'])
- [ ] buy_all() app.route('/buy_all', methods=['POST'])
- [ ] companysAllTravels(upcomingOrPast) @app.route('/companysAllTravels/<string:upcomingOrPast>', methods = ['GET', 'POST'])
- [ ] addCompanyTravel(travelVehicleType) @app.route('/addCompanyTravel/<string:travelVehicleType>', methods = ['GET', 'POST'])
- [ ] aTravelDetails(travelId) @app.route('/aTravelDetails/<int:travelId>', methods = ['GET', 'POST'])
- [ ] commentsOnATravel(travelId) @app.route('/commentsOnATravel/<int:travelId>', methods = ['GET', 'POST'])
- [ ] editUpcomingTravel(travelId) @app.route('/editUpcomingTravel/<int:travelId>', methods = ['GET', 'POST'])
- [ ] deleteATravel(travelId) @app.route('/deleteATravel/<int:travelId>', methods = ['GET', 'POST'])
- [ ] deleteAndRefundAPurchase(PNRToBeDeleted) @app.route('/deleteAndRefundAPurchase/<string:PNRToBeDeleted>', methods=['GET', 'POST'])
- [ ] deleteWORefundAPurchase(PNRToBeDeleted) @app.route('/deleteWORefundAPurchase/<string:PNRToBeDeleted>', methods=['GET', 'POST'])
- [ ] deleteAndGiveFreeTravel(PNRToBeDeleted) @app.route('/deleteAndGiveFreeTravel/<string:PNRToBeDeleted>', methods=['GET', 'POST'])
- [ ] deleteAReservation(PNRToBeDeleted) @app.route('/deleteAReservation/<string:PNRToBeDeleted>', methods=['GET', 'POST'])
- [ ] companyProfile(companyId) @app.route('/companyProfile/<int:companyId>', methods=['GET', 'POST'])
- [ ] editCompanyProfile(companyId) @app.route('/editCompanyProfile/<int:companyId>', methods=['GET', 'POST'])
- [ ] BELOW ARE ADMIN RELATED ROUTES
- [ ] makePurchaseOnBehalfOfTraveler(travelId) @app.route('/makePurchaseOnBehalfOfTraveler/<int:travelId>', methods = ['GET', 'POST'])
- [ ] companies() @app.route('/companies', methods=['GET', 'POST'])
- [ ] deactivateCompany(companyId) @app.route('/deactivateCompany/<int:companyId>', methods = ['GET', 'POST'])
- [ ] deleteCompany(companyId) @app.route('/deleteCompany/<int:companyId>', methods = ['GET', 'POST'])
- [ ] activateCompany(companyId) @app.route('/activateCompany/<int:companyId>', methods = ['GET', 'POST'])
- [ ] validateCompany(companyId) @app.route('/validateCompany/<int:companyId>', methods = ['GET', 'POST'])

- [x] couponManagement() @app.route('/couponManagement', methods = ['GET', 'POST'])
- [x] createCoupon() @app.route('/createCoupon', methods = ['GET', 'POST'])
- [x] deleteACoupon(couponId) @app.route('/deleteACoupon/<int:couponId>', methods = ['GET', 'POST'])
- [ ] vehicleManagement() @app.route('/vehicleManagement', methods = ['GET', 'POST'])
- [ ] createVehicleType() @app.route('/createVehicleType', methods = ['GET', 'POST'])
- [ ] deleteAVehicleType(vehicleTypeId) @app.route('/deleteAVehicleType/<int:vehicleTypeId>', methods = ['GET', 'POST'])
- [ ] terminalManagement() @app.route('/terminalManagement', methods = ['GET', 'POST'])
- [ ] deleteTerminal(terminalId) @app.route('/deleteTerminal/<int:terminalId>', methods = ['GET', 'POST'])
- [ ] createTerminal() @app.route('/createTerminal', methods = ['GET', 'POST'])
- [ ] reportManagement() @app.route('/reportManagement', methods = ['GET', 'POST'])
- [ ] reportDetails(report_id) @app.route('/reportDetails/<int:report_id>', methods = ['GET', 'POST'])
- [ ] printReportDetails(report_id) @app.route('/printReportDetails/<int:report_id>', methods=['GET', 'POST'])
- [ ] createReport() @app.route('/createReport', methods = ['GET', 'POST'])
- [ ] companysAllTravelsByAdmin(upcomingOrPast, companyId) @app.route('/companysAllTravelsByAdmin/<int:companyId>/<string:upcomingOrPast>', methods = ['GET', 'POST'])

Cognito

What is Amazon Cognito?
https://docs.aws.amazon.com/cognito/latest/developerguide/what-is-amazon-cognito.html

Common Amazon Cognito Scenarios:
https://docs.aws.amazon.com/cognito/latest/developerguide/cognito-scenarios.html

Using tokens with user pools:
https://docs.aws.amazon.com/cognito/latest/developerguide/amazon-cognito-user-pools-using-tokens-with-identity-providers.html

Authentication with a user pool:
https://docs.aws.amazon.com/cognito/latest/developerguide/authentication.html

Integrating Amazon Cognito authentication and authorization with web and mobile apps:
https://docs.aws.amazon.com/cognito/latest/developerguide/cognito-integrate-apps.html

Amplify Auth Concepts:
https://docs.amplify.aws/react/build-a-backend/auth/concepts/

Token And Credentials:
https://docs.amplify.aws/react/build-a-backend/auth/concepts/tokens-and-credentials/

Set up Amplify Auth:
https://docs.amplify.aws/react/build-a-backend/auth/set-up-auth/

- Instead of Authenticator, you can bring your own UI and leverage the library from aws-amplify to handle authentication flows manually. Since we have custom ui, we will probably use this library.

Enable sign-up, sign-in, and sign-out:
https://docs.amplify.aws/gen1/react/prev/build-a-backend/auth/enable-sign-up/
https://docs.amplify.aws/gen1/react/build-a-backend/auth/enable-sign-up/

Configure Amplify:
https://docs.amplify.aws/gen1/react/build-a-backend/auth/set-up-auth/
https://docs.amplify.aws/gen1/react/tools/libraries/configure-categories/

Manage user sessions
https://docs.amplify.aws/react/build-a-backend/auth/connect-your-frontend/manage-user-sessions/

Manage User Profiles
https://docs.amplify.aws/gen1/react/build-a-backend/auth/manage-user-profile/

Using the ID Token
https://docs.aws.amazon.com/cognito/latest/developerguide/amazon-cognito-user-pools-using-the-id-token.html

Using the access token:
https://docs.aws.amazon.com/cognito/latest/developerguide/amazon-cognito-user-pools-using-the-access-token.html

Configure cross-account Amazon Cognito authorizer for a REST API using the API Gateway console:
https://docs.aws.amazon.com/apigateway/latest/developerguide/apigateway-cross-account-cognito-authorizer.html

Integrate a REST API with an Amazon Cognito user pool: (THIS IS IMPORTANT !!!)
https://docs.aws.amazon.com/apigateway/latest/developerguide/apigateway-enable-cognito-user-pool.html

Create an Amazon Cognito user pool for a REST API:
https://docs.aws.amazon.com/apigateway/latest/developerguide/apigateway-create-cognito-user-pool.html

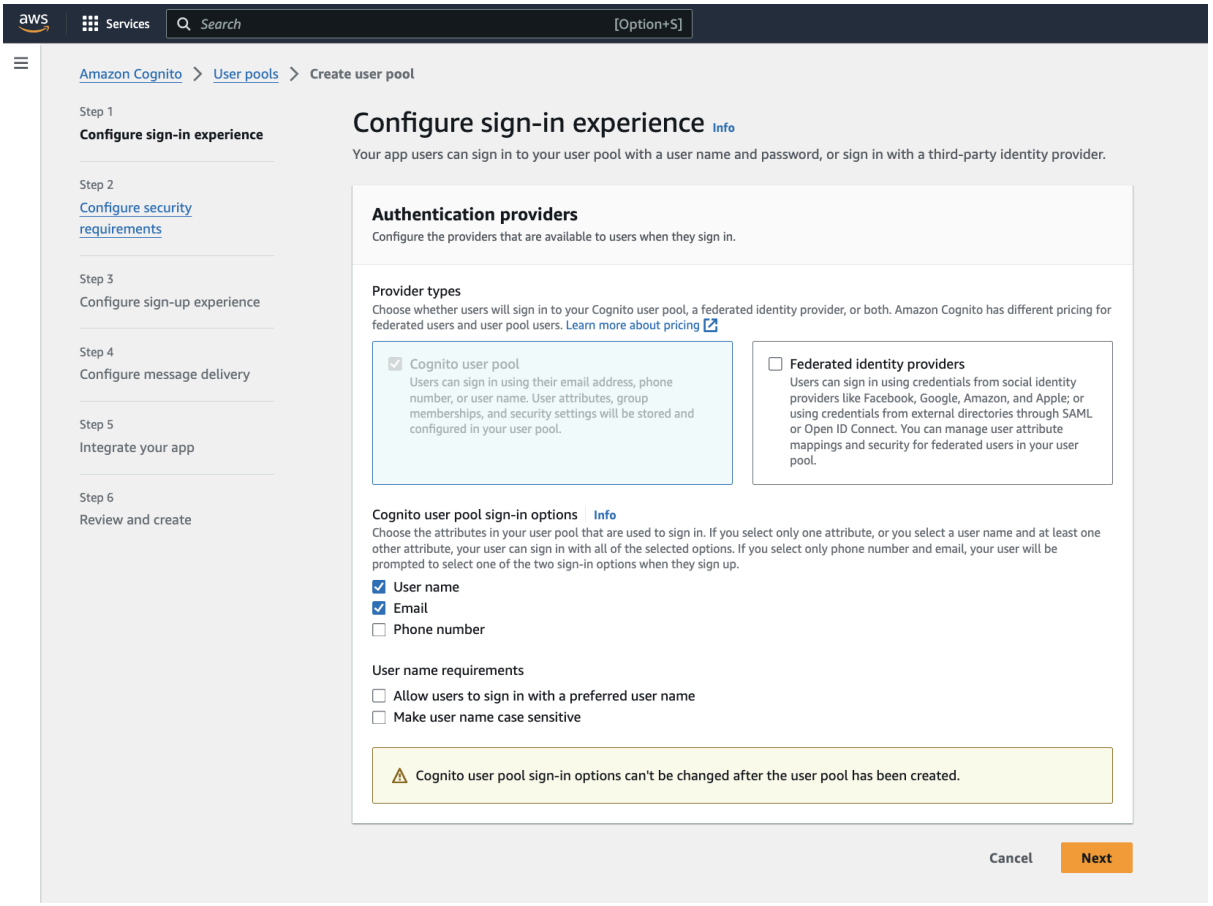
IMPORTANT - how-to-change-user-status-force-change-password
https://stackoverflow.com/questions/40287012/how-to-change-user-status-force-change-password

Some Youtube Videos:

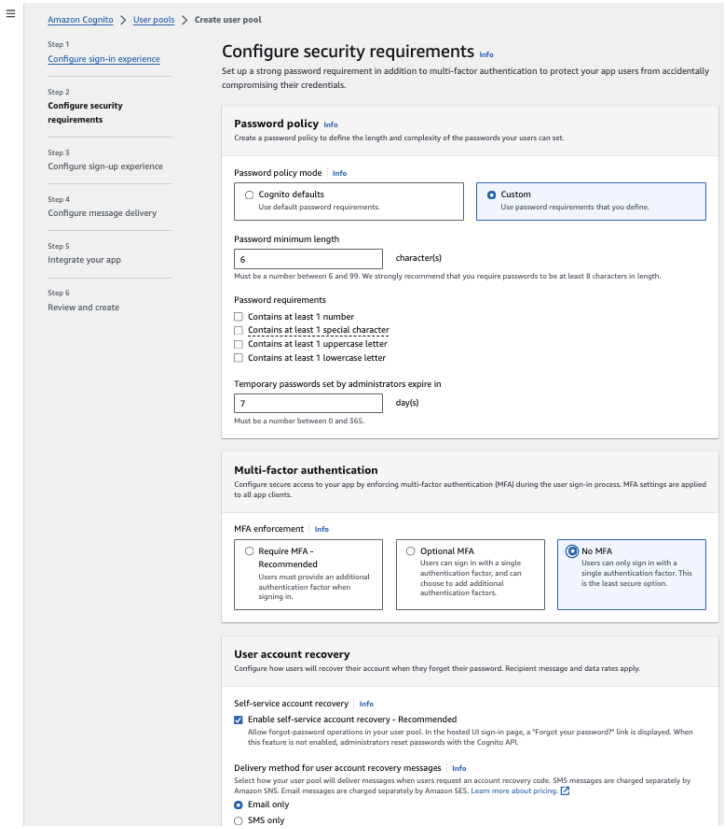
- https://www.youtube.com/watch?v=ma1FA2be8Ac
- https://www.youtube.com/watch?v=8a0vtkWJIA4
- https://www.youtube.com/watch?v=LI31QxfAgho

Creating Cognito User Pool

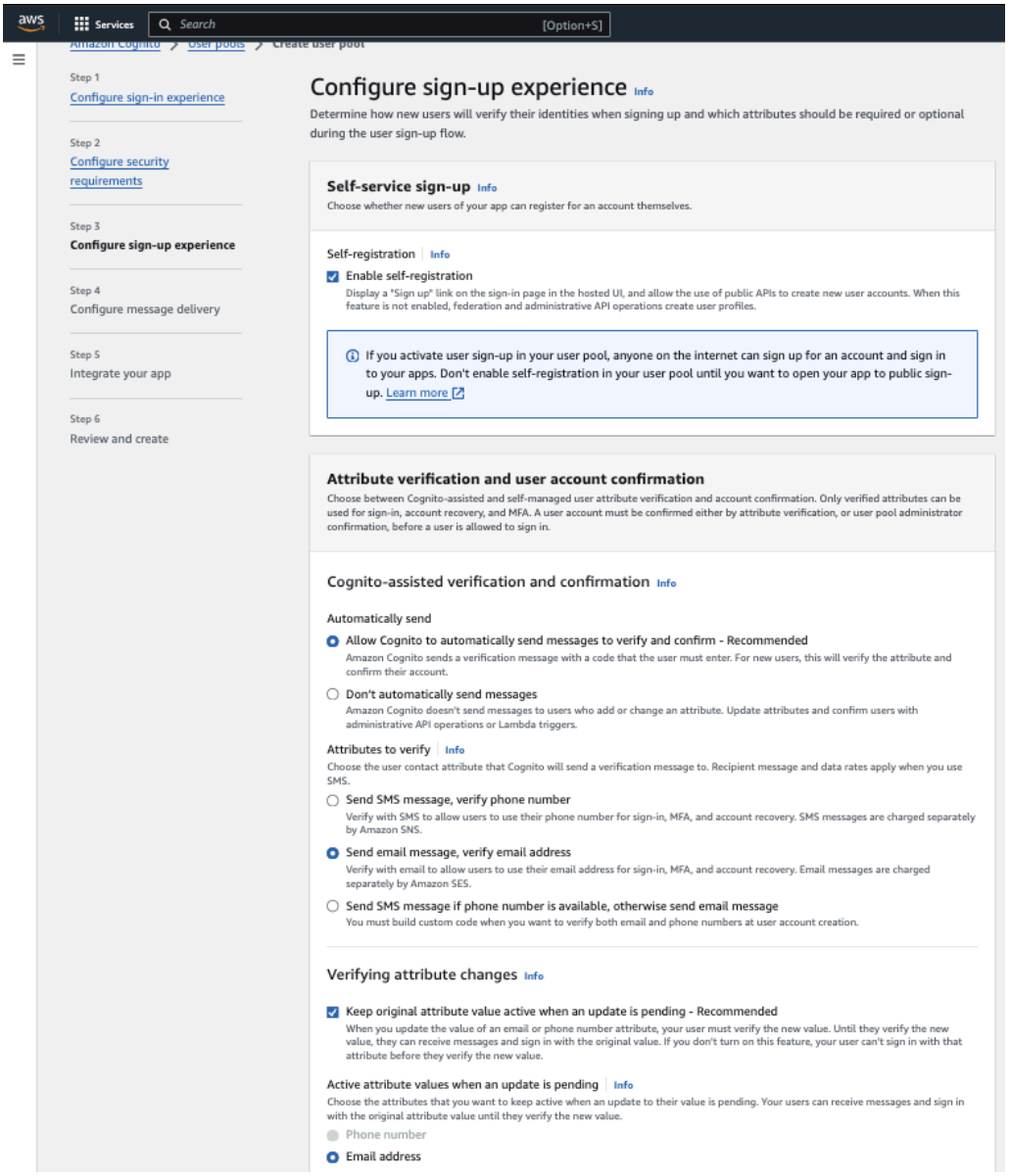
Select user pool sign-in options.AWS requires username for signup (later I only select username)

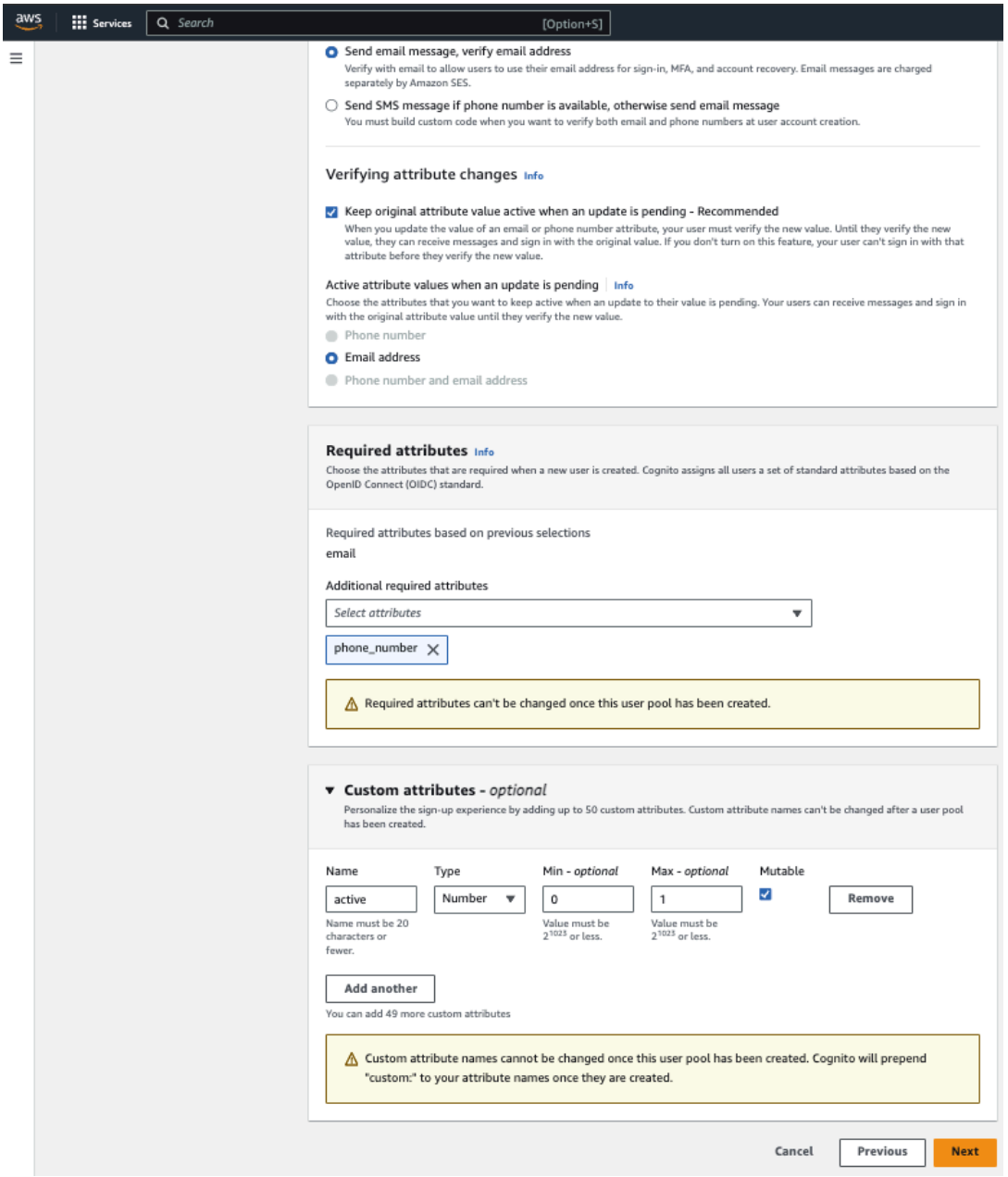


Now, it is time for security requirements configuration. Since our users passwords are all 123456 for the demos. I select custom password policy and remove all the password requirements like min number of special character etc. No multi-factor authentication is selected since in our app we don't use it. In real life, it would be nice to use it. For the user account recovery, "email only" is. Note that SMS messages are charged separately by Amazon SNS. Email messages are charged separately by Amazon SES.

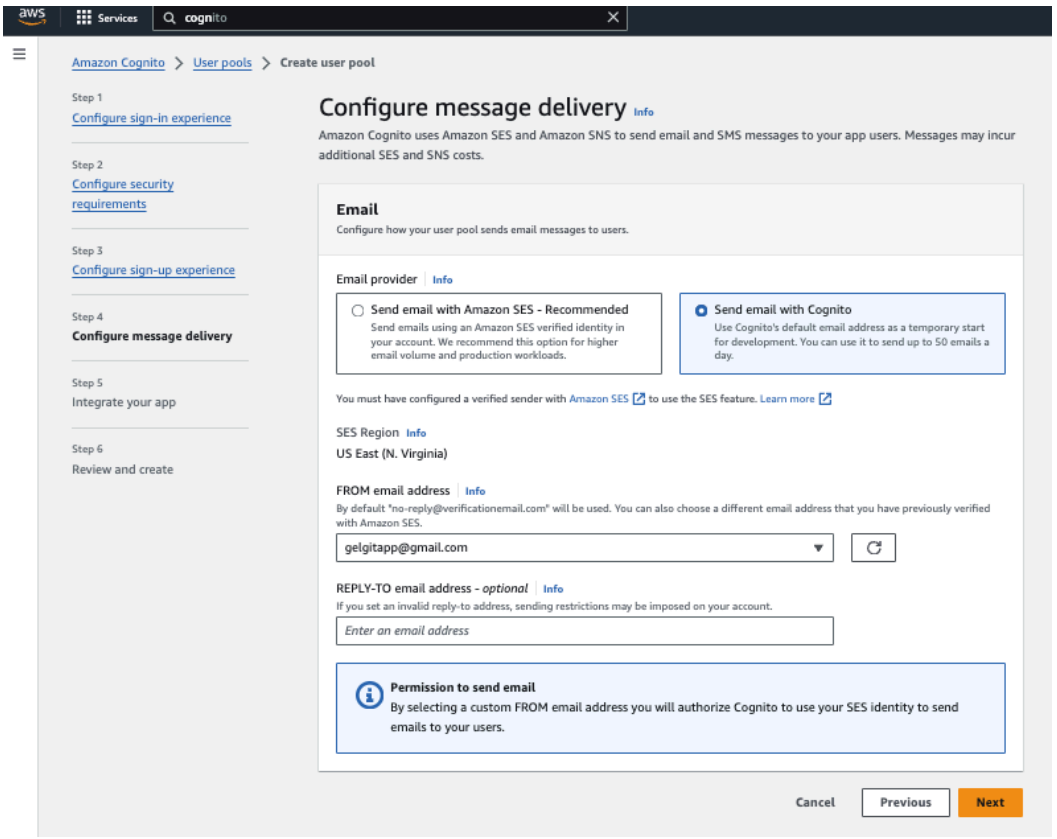


For the sign-up experience, the following configuration is set. I allow automatic send messages to verify and confirm with email. And then, I set phone_number as the required attribute. (Update, I remove the phone_number from the required attributes) Also, I add custom attribute named as "active" since our user table in the database contains a column named "active".

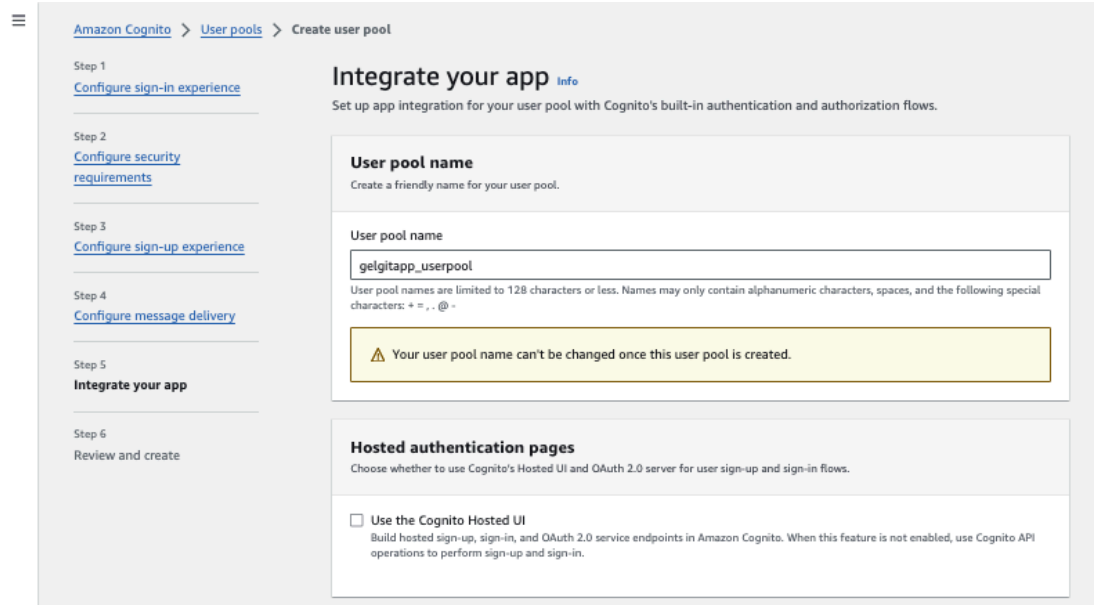

aws
Services
Search
[Option+S]
Amazon Cognito > User pools > Create user pool
Step 1
Configure sign-in experience
Step 2
Configure security requirements
Step 3
Configure sign-up experience
Step 4
Configure message delivery
Step 5
Integrate your app
Step 6
Review and create
Configure sign-up experience Info
Determine how new users will verify their identities when signing up and which attributes should be required or optional during the user sign-up flow.
Self-service sign-up Info
Choose whether new users of your app can register for an account themselves.
Self-registration Info
Enable self-registration
Display a "Sign up" link on the sign-in page in the hosted UI, and allow the use of public APIs to create new user accounts. When this feature is not enabled, federation and administrative API operations create user profiles.
If you activate user sign-up in your user pool, anyone on the internet can sign up for an account and sign in to your apps. Don't enable self-registration in your user pool until you want to open your app to public sign-up. Learn more
Attribute verification and user account confirmation
Choose between Cognito-assisted and self-managed user attribute verification and account confirmation. Only verified attributes can be used for sign-in, account recovery, and MFA. A user account must be confirmed either by attribute verification, or user pool administrator confirmation, before a user is allowed to sign in.
Cognito-assisted verification and confirmation Info
Automatically send
Allow Cognito to automatically send messages to verify and confirm - Recommended
Amazon Cognito sends a verification message with a code that the user must enter. For new users, this will verify the attribute and confirm their account.
Don't automatically send messages
Amazon Cognito doesn't send messages to users who add or change an attribute. Update attributes and confirm users with administrative API operations or Lambda triggers.
Attributes to verify Info
Choose the user contact attribute that Cognito will send a verification message to. Recipient message and data rates apply when you use SMS.
Send SMS message, verify phone number
Verify with SMS to allow users to use their phone number for sign-in, MFA, and account recovery. SMS messages are charged separately by Amazon SNS.
Send email message, verify email address
Verify with email to allow users to use their email address for sign-in, MFA, and account recovery. Email messages are charged separately by Amazon SES.
Send SMS message if phone number is available, otherwise send email message
You must build custom code when you want to verify both email and phone numbers at user account creation.
Verifying attribute changes Info
Keep original attribute value active when an update is pending - Recommended
When you update the value of an email or phone number attribute, your user must verify the new value. Until they verify the new value, they can receive messages and sign in with the original value. If you don't turn on this feature, your user can't sign in with that attribute before they verify the new value.
Active attribute values when an update is pending Info
Choose the attributes that you want to keep active when an update to their value is pending. Your users can receive messages and sign in with the original attribute value until they verify the new value.
Phone number
Email address

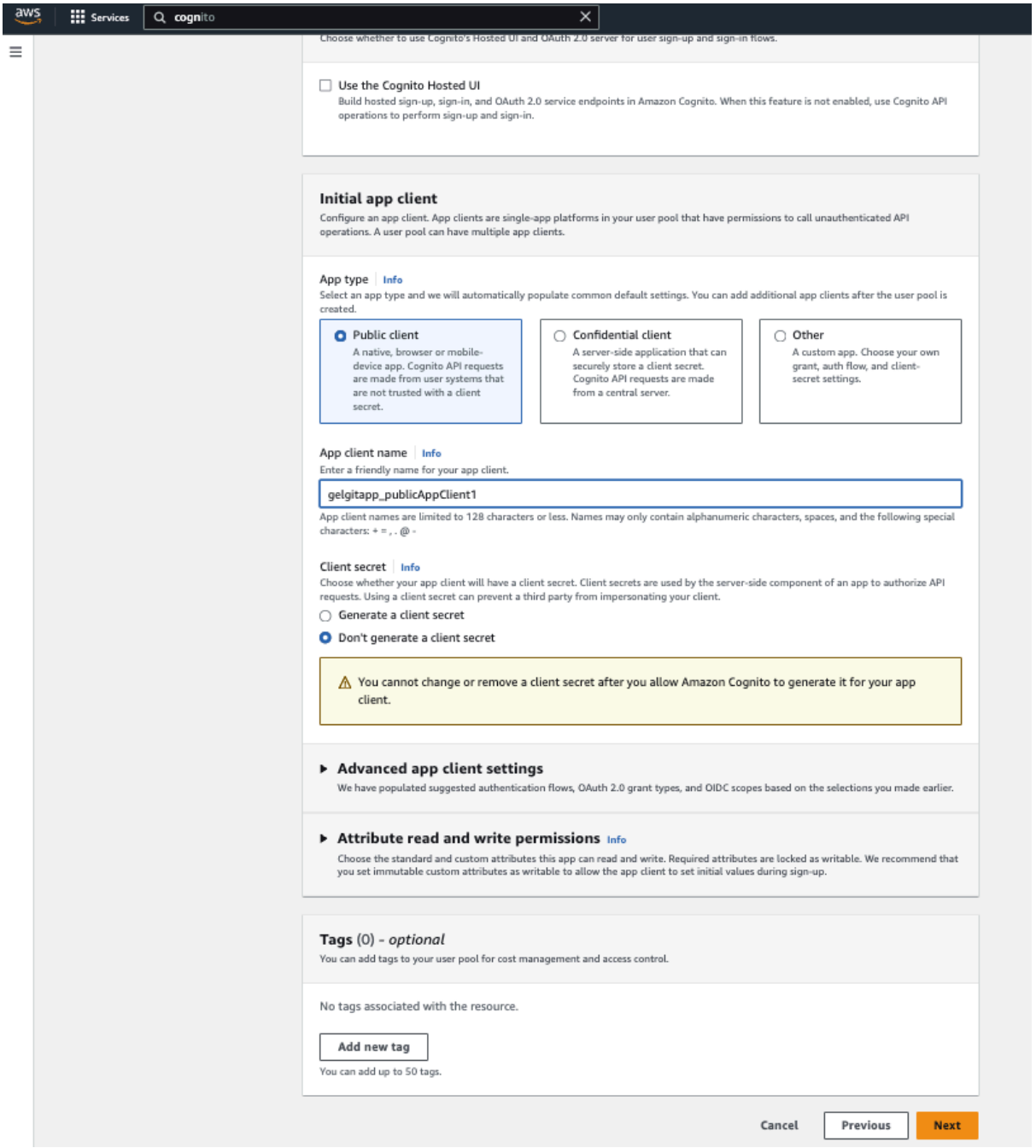


For the message delivery configuration, I selected "send email with Cognito" although SES (Simple Email Service) is recommended. This setting can be changed.

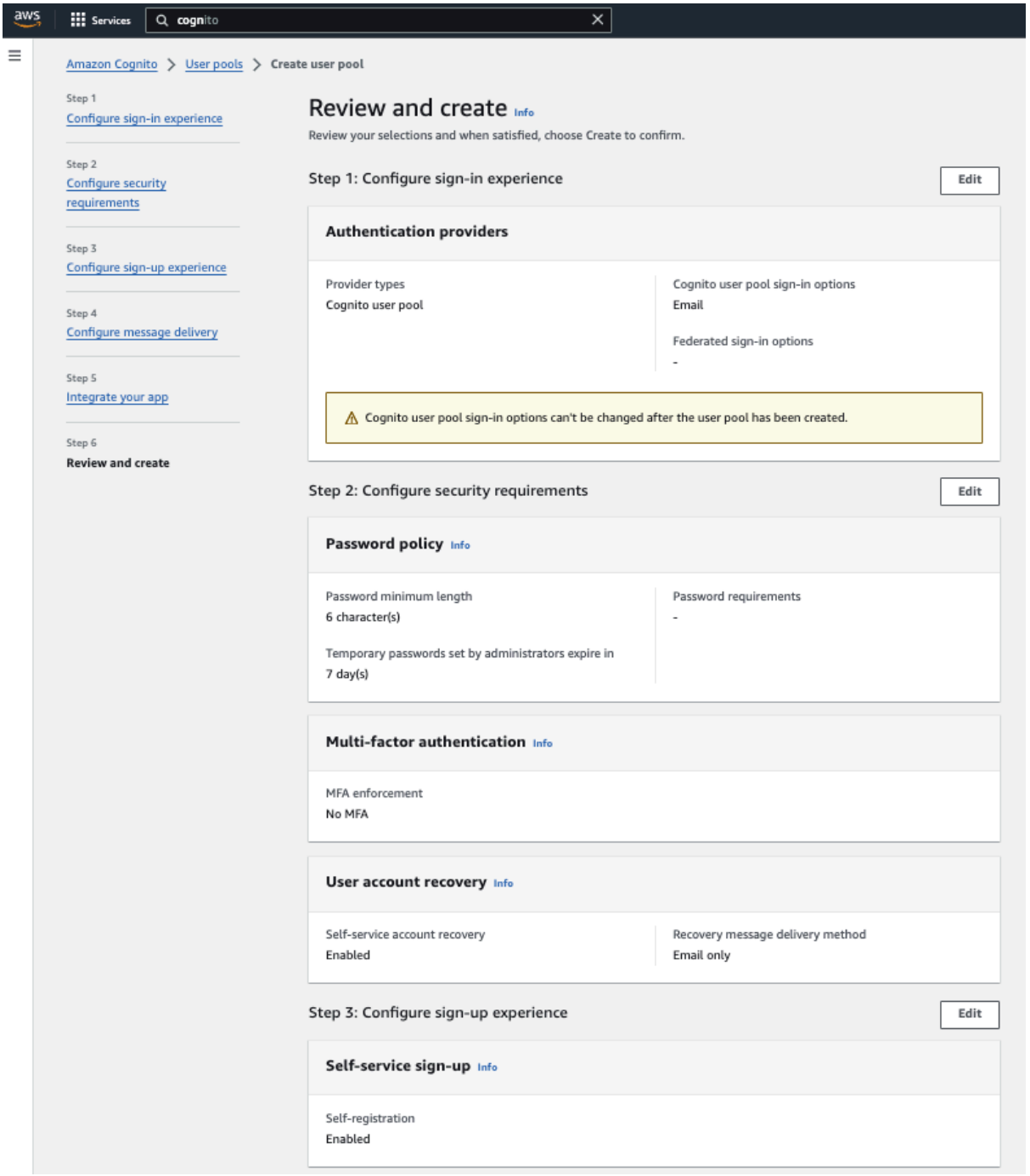


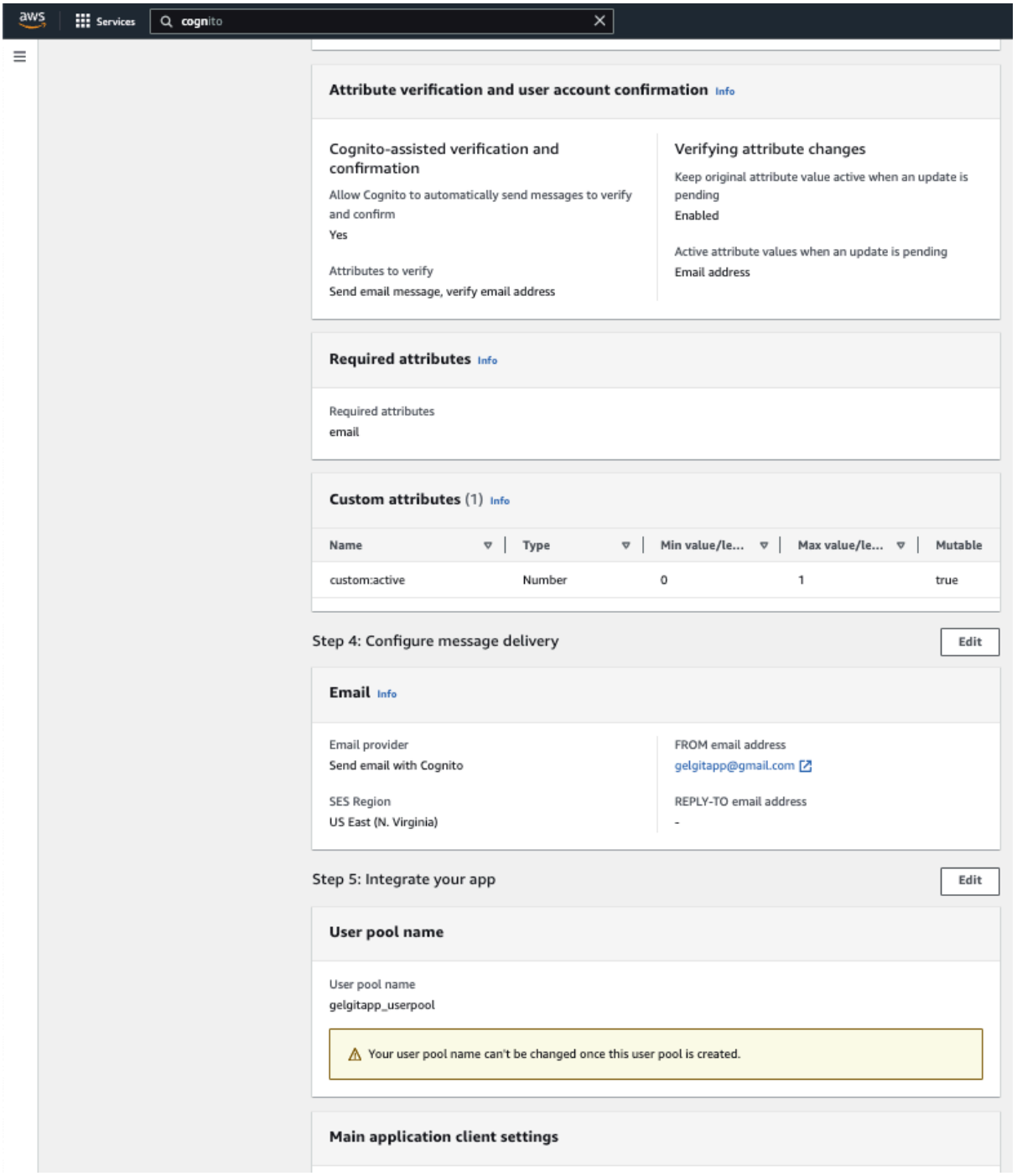
Now, it is time to configure the integration part.
We didn't use the Cognito Hosted UI since we have a ui. Then, I created an app client and selected not to generate a client secret.

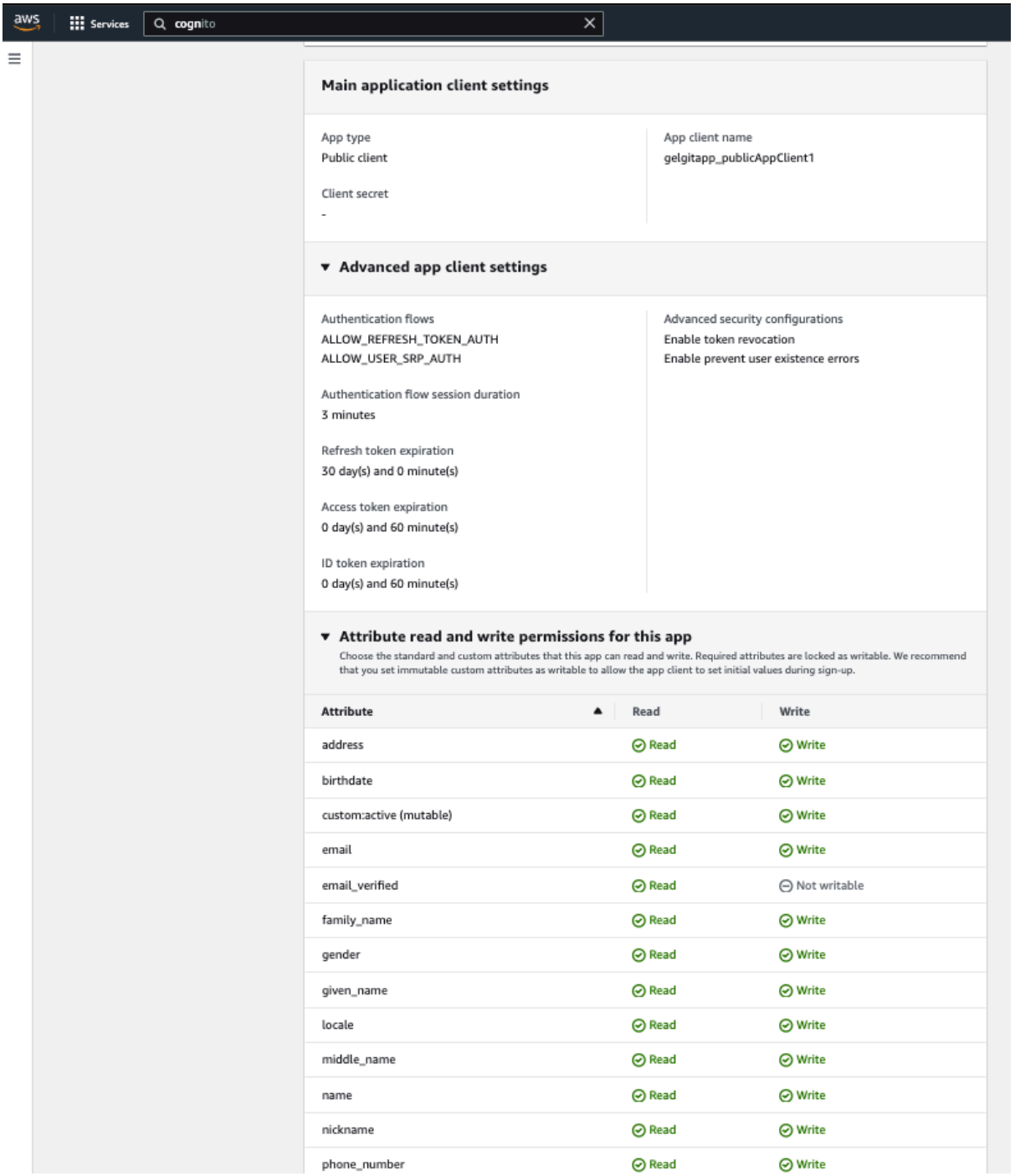




Reviewing the user pool settings


aws
Services
cognito
Amazon Cognito > User pools > Create user pool
Step 1
Configure sign-in experience
Step 2
Configure security requirements
Step 3
Configure sign-up experience
Step 4
Configure message delivery
Step 5
Integrate your app
Step 6
Review and create
Review and create Info
Review your selections and when satisfied, choose Create to confirm.
Step 1: Configure sign-in experience
Edit
Authentication providers
Provider types
Cognito user pool
Cognito user pool sign-in options
Email
Federated sign-in options
-
Cognito user pool sign-in options can't be changed after the user pool has been created.
Step 2: Configure security requirements
Edit
Password policy Info
Password minimum length
6 character(s)
Password requirements
-
Temporary passwords set by administrators expire in
7 day(s)
Multi-factor authentication Info
MFA enforcement
No MFA
User account recovery Info
Self-service account recovery
Enabled
Recovery message delivery method
Email only
Step 3: Configure sign-up experience
Edit
Self-service sign-up Info
Self-registration
Enabled

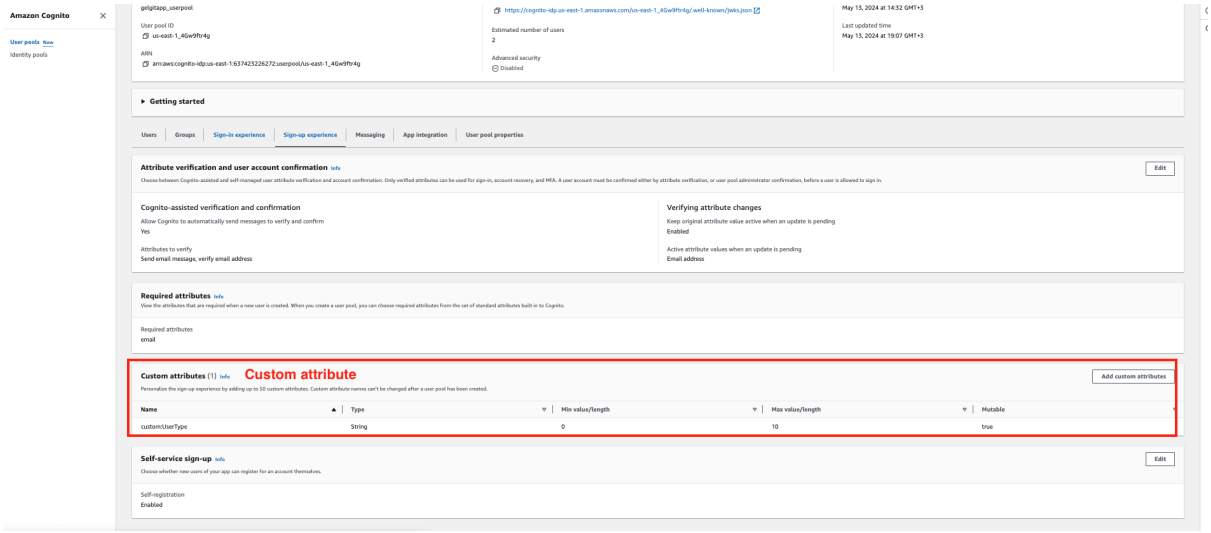

aws
Services
cognito
Attribute verification and user account confirmation Info
Cognito-assisted verification and confirmation
Allow Cognito to automatically send messages to verify and confirm
Yes
Attributes to verify
Send email message, verify email address
Verifying attribute changes
Keep original attribute value active when an update is pending
Enabled
Active attribute values when an update is pending
Email address
Required attributes Info
Required attributes
email
Custom attributes (1) Info
Name
Type
Min value/le...
Max value/le...
Mutable
custom:active
Number
0
1
true
Step 4: Configure message delivery
Edit
Email Info
Email provider
Send email with Cognito
SES Region
US East (N. Virginia)
FROM email address
gelgitapp@gmail.com
REPLY-TO email address
-
Step 5: Integrate your app
Edit
User pool name
User pool name
gelgitapp_userpool
Your user pool name can't be changed once this user pool is created.
Main application client settings



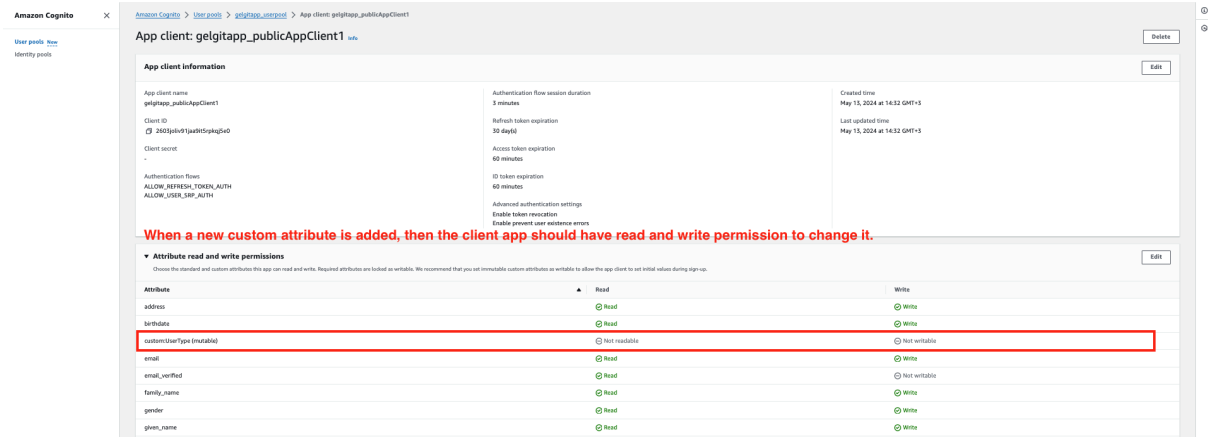
In order to change user verification code to a link do the following:

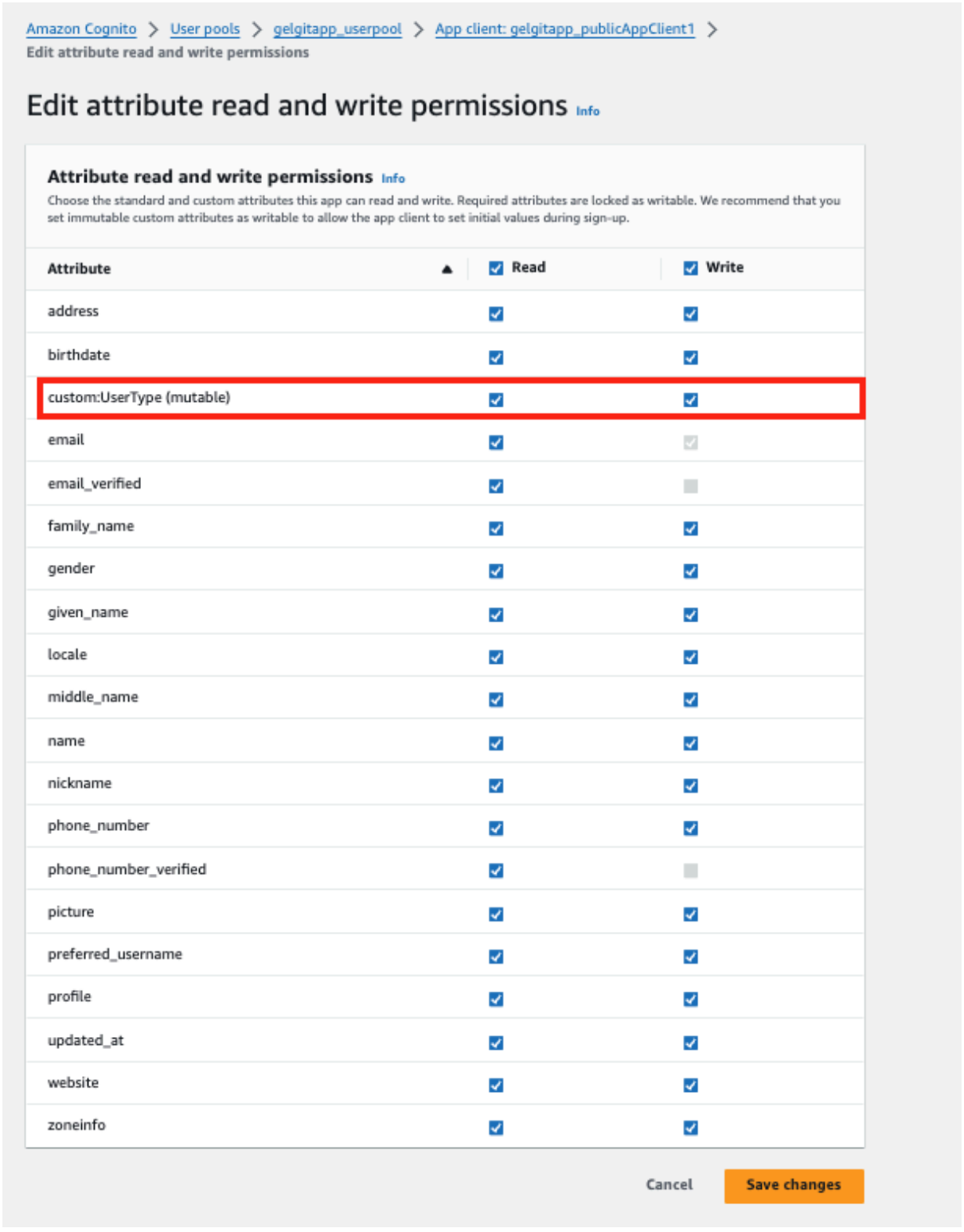
"User confirmed their account with a verification link sent to their phone or email. In order to enable this option you need to go to the Amazon Cognito console, look for your userpool, then go to the Messaging tab and enable link mode inside the Verification message option. Finally you need to define the signUpVerificationMethod to link inside the Cognito option of your Auth config"

Adding custom attribute.

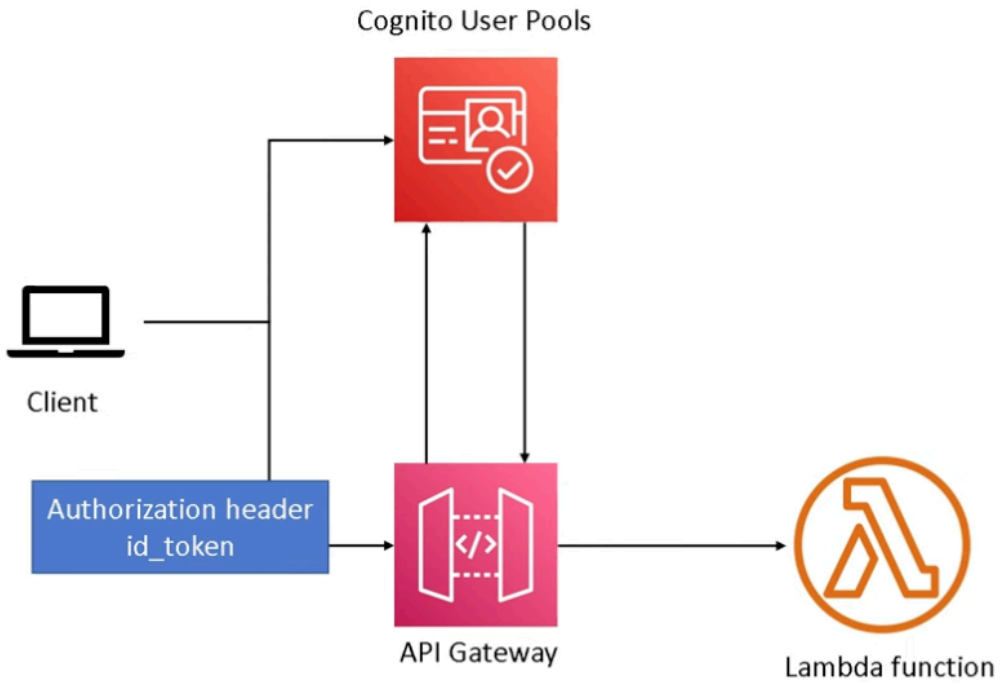


When a custom attribute is added, then the client app should have read and write permission to store/read/write.





Using Cognito for Authorization for the API Gateway



Adding authorization to an API Gateway resource
Read the article below
Integrate a REST API with an Amazon Cognito user pool: (THIS IS IMPORTANT !!!)
https://docs.aws.amazon.com/apigateway/latest/developerguide/apigateway-enable-cognito-user-pool.html

API Gateway > APIs > Resources - gelgitapp (34b7tnxk3e) > Edit method request

Edit method request

Method request settings

Authorization

gelgitapp_userpool_authorizer

Authorization Scopes

Add a scope | Add

Request validator

None

API key required

Operation name - optional

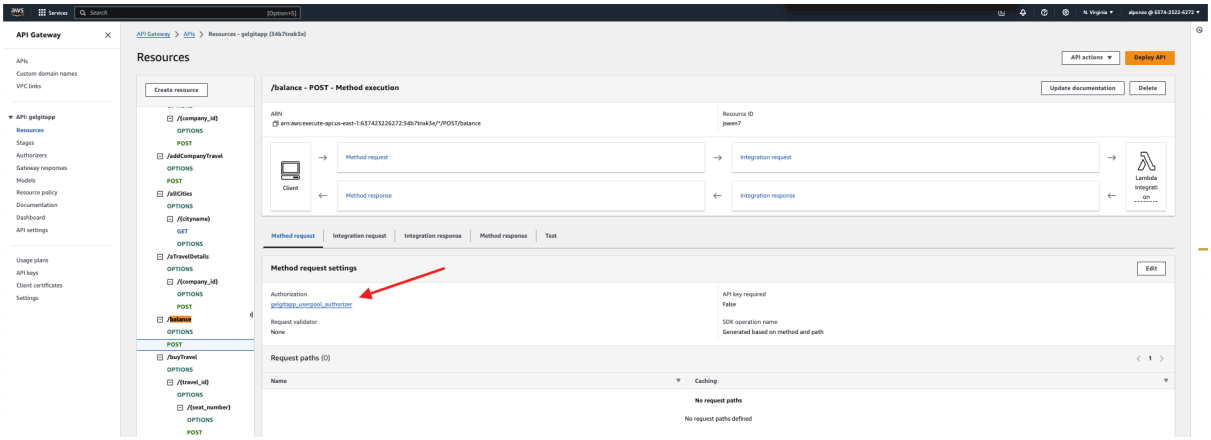
GetPets

URL query string parameters

HTTP request headers

Request body

Cancel | Save



CI/CD

ci/cd with amplify:
https://aws.amazon.com/blogs/mobile/complete-guide-to-full-stack-ci-cd-workflows-with-aws-amplify/
CI/CD pipelines with AWS Amplify
https://dev.to/this-is-learning/cicd-pipelines-with-aws-amplify-5a1b

References:

- AWS official post: Create an Amazon RDS MySQL Database
  - https://docs.aws.amazon.com/opsworks/latest/userguide/customizing-rds-connect-create.html
- AWS official post: Creating and connecting to a MySQL DB instance
  - https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_GettingStarted.CreatingConnecting.MySQL.html
- AWS official post: Connecting to a DB instance running the MySQL database engine
  - https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_ConnectToInstance.html
- Unofficial post:
  - https://beabetterdev.com/2022/12/13/how-to-connect-to-an-rds-or-aurora-database-in-a-private-subnet/
- AWS official post: Automatically connecting a Lambda function and a DB instance
  - https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/lambda-rds-connect.html
- AWS official post: Tutorial: Using a Lambda function to access an Amazon RDS database
  - https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/rds-lambda-tutorial.html
-
- https://stackoverflow.com/questions/20343259/how-to-create-table-in-rds-database-on-amazon-web-service
- Understanding data models
  - https://docs.aws.amazon.com/apigateway/latest/developerguide/models-mappings-models.html
- Enabling CORS in AWS API Gateway
  - https://docs.aws.amazon.com/apigateway/latest/developerguide/how-to-cors-console.html
- AWS Secret Manager:
  - https://docs.aws.amazon.com/secretsmanager/latest/userguide/intro.html
- PyMySQL
  - https://pymysql.readthedocs.io/en/latest/modules/index.html
  - https://pymysql.readthedocs.io/en/latest/user/examples.html
  - https://pymysql.readthedocs.io/en/latest/modules/cursors.html#pymysql.cursors.Cursor.fetchone