Aurora MySQL Hands-on Lab Manual – Version 2.0

This hands-on lab manual will cover following tasks and procedure:

- ✓ **Create Aurora MySQL DB** Instance
- ✓ Grant Aurora MySQL access i.e. **modify security group** to allow access from your computer
- ✓ **Load data from S3** into Aurora MySQL database
- ✓ **Create read replica** instance & access table
- ✓ **Database copy** using Aurora "**Clone**" feature
- ✓ Perform DML query on primary DB and validate data on primary and cloned DB copy

Note:

AWS Region: Use single region (us-west-2) Aurora MySQL Cluster
No permission to access any other AWS region, delete any existing database, security group
Use your Aurora cluster database only based on your AWS user account name, since you can view other databases as well

1. Create an Amazon Aurora MySQL Database Cluster using db.t2.small instance type

- See documentation at
  https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Aurora.CreateInstance.html

  The name of the aurora cluster should be based on AWS User name like
  AWS User name: student1
  Name of cluster: aurora-student-1

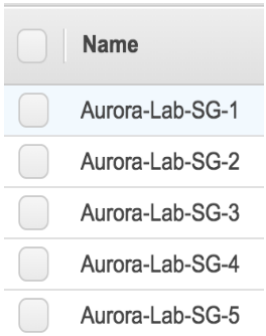
  Name of cluster: `aurora-student-lab<#>`

Configure advanced settings

VPC: Use the default VPC
Subnet: Default

Public Accessibility: Yes (should be OK for this lab)

VPC Security Groups:
Choose one of the existing VPC Security groups like Aurora-Lab-SG-1

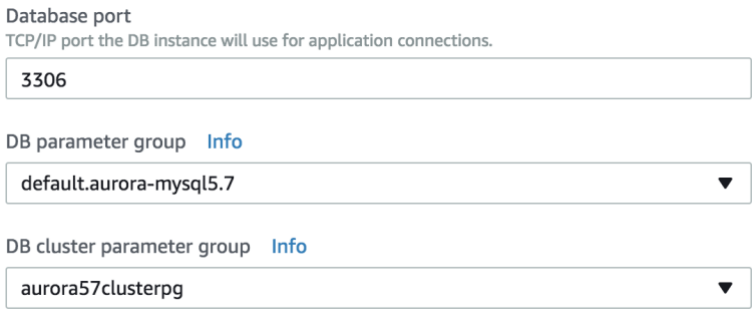


Encryption: Disable Encryption

Monitoring: Disable enhanced monitoring

DB parameter group: default.aurora-mysql5.7

DB cluster parameter group: aurora57clusterpg



aurora57clusterpg cluster level parameter customized with following settings, which will allow Aurora cluster to access customer S3 bucket with IAM role

```
+----------------------------+-----------------------------------------------+
| Variable_name              | Value                                         |
+----------------------------+-----------------------------------------------+
| aurora_load_from_s3_role   | arn:aws:iam::<AWSaccount#>:role/psa-iam-role|
| aws_default_s3_role        | arn:aws:iam::<AWSaccount#>:role/psa-iam-role|
+----------------------------+-----------------------------------------------+
```

- Choose a username and a password (and don't forget them! note down)
- Once the instance is created, find your Aurora Cluster and instance "Endpoint" or connect info

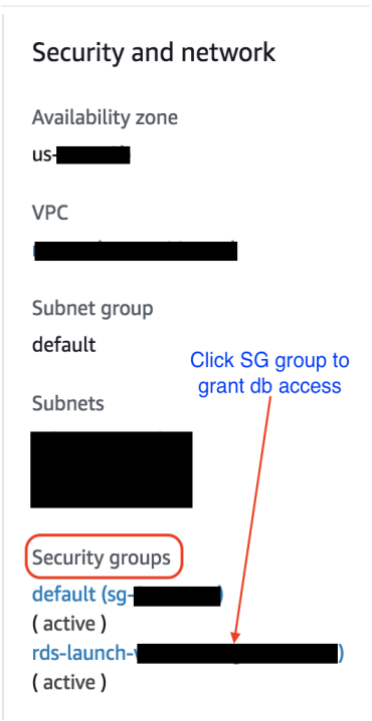
  i. On the AWS Console, choose Services, then RDS
  ii. On the RDS dashboard, choose Clusters
  iii. Select your Aurora Cluster name and click

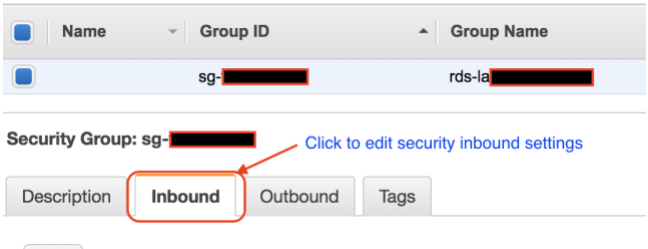
  iv. Note your endpoint name. You will need it later!

Note: By default, Aurora Reader endpoint will create and it will use master DB instance when you create Aurora cluster without any read replica DB instance. When you will add read replica later, the reader endpoint will automatically use read replica for all read only queries.

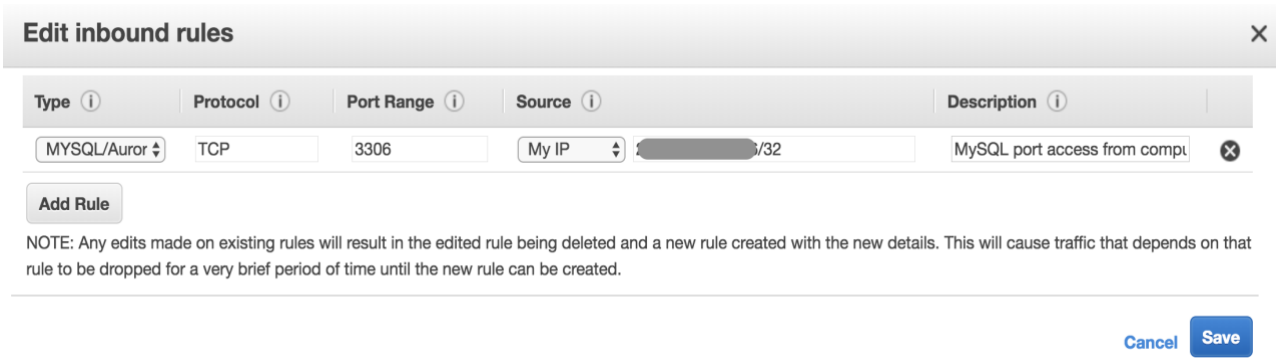
2. Grant database access from your computer

To grant MySQL DB access from your computer, modify db security group. Click on aurora primary db instance select the security group





Select Type=> MYSQL/Aurora Source=> My IP ; Automatically extract your computer IP address



Verify you can access the mysql client from your computer

```
mysql -h<endpoint name> -u<db user name> -p
```

Example:

```
mysql -h aurora-lab.xxxxxxxxx.us-west-2.rds.amazonaws.com -u<db user> -p
```

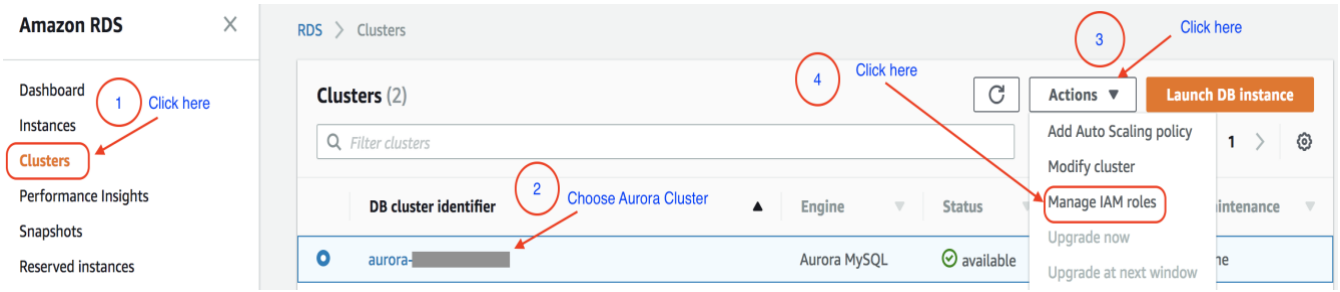
```
Enter password:

mysql>
```

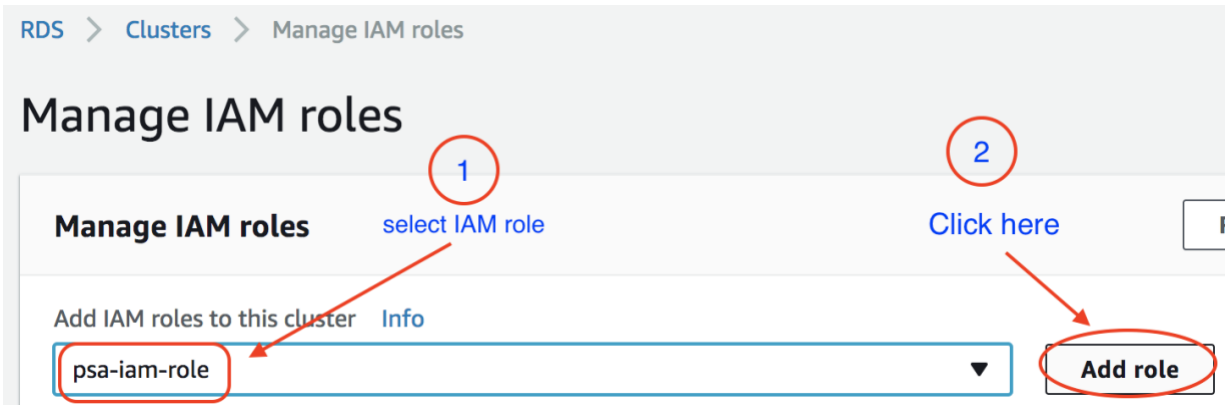
To exit from the “mysql>” prompt, use CTRL-D

3.Associating an IAM Role with an Amazon Aurora MySQL DB Cluster

IAM Role and policy has been already created for this hands-on lab session. We will not create IAM role & policy however, we will use existing role to load data from S3 bucket.

(See the AWS doc to learn how to create IAM role later https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/AuroraMySQL.Integrating.Authorizing.IAM.S3CreatePolicy.html)





4.Data load into Aurora MySQL database

- Load to Aurora MySQL – Connect to cluster end-point or primary db instance

```
$ mysql -h aurora-lab.xxxxxxxxx.us-west-2.rds.amazonaws.com -u<db user> -p
```

Create a landsat database

```
mysql> CREATE DATABASE landsat;
mysql> USE landsat;
```

Create the scene_list table

```
CREATE TABLE `scene_list` (
  `entityId` varchar(64) DEFAULT NULL,
  `acquisitionDate` datetime DEFAULT NULL,
  `cloudCover` decimal(5,2) DEFAULT NULL,
  `processingLevel` varchar(8) DEFAULT NULL,
  `path` int(11) DEFAULT NULL,
  `row` int(11) DEFAULT NULL,
  `min_lat` decimal(8,5) DEFAULT NULL,
  `min_lon` decimal(8,5) DEFAULT NULL,
  `max_lat` decimal(8,5) DEFAULT NULL,
  `max_lon` decimal(8,5) DEFAULT NULL,
  `download_url` varchar(128) DEFAULT NULL);
```

Validate IAM S3 role settings parameter settings (should expect following output from SQL command)

```
mysql> show global variables like '%s3%';
+---------------------------+-----------------------------------------------+
| Variable_name             | Value                                         |
+---------------------------+-----------------------------------------------+
| aurora_load_from_s3_role  | arn:aws:iam::<AWSaccount#>:role/psa-iam-role|
| aurora_select_into_s3_role |                                              |
| aws_default_s3_role       | arn:aws:iam::<AWSaccount#>:role/psa-iam-role|
+---------------------------+-----------------------------------------------+
```

Load landsat data into scene_list table

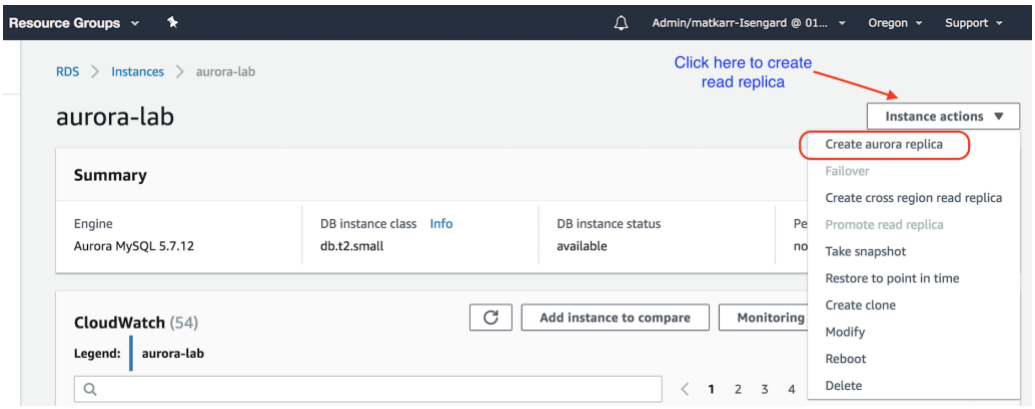
```
mysql> LOAD DATA FROM S3 's3://psa-hands-on/scene_list' INTO TABLE scene_list
FIELDS TERMINATED BY ',';
```

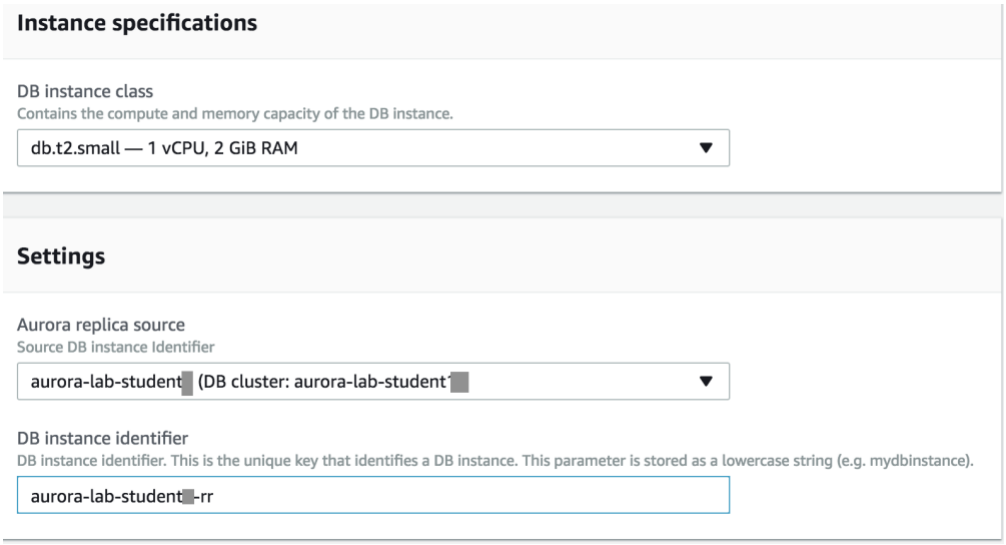
Run SQL query against scene_list table:

```
mysql> select count(*) from scene_list;

mysql> select * from scene_list limit 5;
```

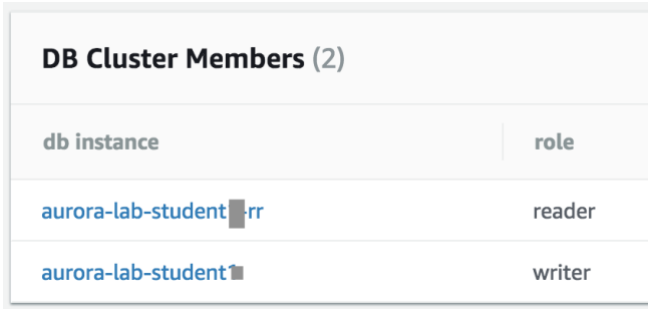
5.Create a new Read replica called "**aurora-lab-student<#>-rr**" within a region. Use same security group as used from primary Aurora DB instance.

To create new read replica, choose primary database > Instance actions > Create aurora replica





After the read replica creation, the Aurora Cluster will show:



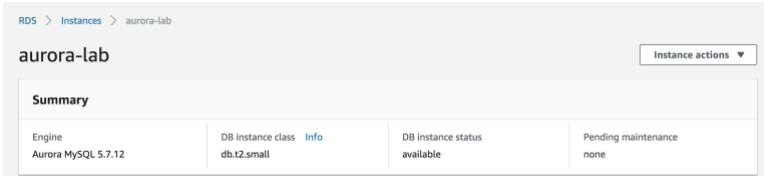
- Use Reader endpoint to run SELECT query; connect read replica from your computer

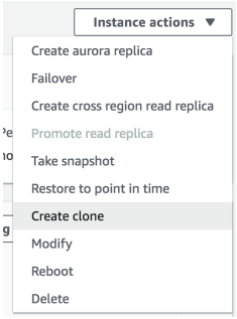
```
mysql -haurora-lab-student<#>.cluster-ro-xxxxxxx.us-west-2.rds.amazonaws.com
    -u <username> -p

   mysql> use landsat;

   mysql> select count(*) from scene_list;
```

6.Create database copy by using Clone Database

Create a clone database called "**autora-lab-student<#>-clone**" using db.t2.small instance type.

Select db instance name (writer mode) > Instance actions > Create clone





7.Run DELETE query on primary DB instance and validate number of rows from primary and cloned database.

Connect to primary MySQL database using Aurora cluster end-point from your computer & run delete query:

```
$ mysql -haurora-lab.studentxxxxx.us-xxxx.rds.amazonaws.com -u <username> -p

mysql> use landsat;

mysql> select count(*) from scene_list;
+----------+
| count(*) |
+----------+
|  1059757 |
     +----------+

mysql> delete from scene_list limit 25;
Query OK, 25 rows affected (0.01 sec)

mysql> select count(*) from scene_list;
+----------+
| count(*) |
+----------+
|  1059732 |
+----------+
```

Connect to cloned MySQL database from your computer & validate number of table rows:

```
$ mysql -haurora-lab-student<#>-clone.xxxxx.us-xxxxxx.rds.amazonaws.com -u<username>
-p


mysql> use landsat;

 mysql> select count(*) from scene_list;
+-----------+
| count(*) |
+-----------+
|  1059757 |
     +-----------+
```