ORACLE

Migration Guide: Amazon Aurora to MySQL HeatWave on Amazon Web Services (AWS)

Purpose statement

This document provides an overview of the steps to migrate to MySQL HeatWave.

Disclaimer

This document in any form, software or printed matter, contains proprietary information that is the exclusive property of Oracle. Your access to and use of this confidential material is subject to the terms and conditions of your Oracle software license and service agreement, which has been executed and with which you agree to comply. This document and information contained herein may not be disclosed, copied, reproduced or distributed to anyone outside Oracle without prior written consent of Oracle. This document is not part of your license agreement nor can it be incorporated into any contractual agreement with Oracle or its subsidiaries or affiliates.

This document is for informational purposes only and is intended solely to assist you in planning for the implementation and upgrade of the product features described. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described in this document remains at the sole discretion of Oracle. Due to the nature of the product architecture, it may not be possible to safely include all features described in this document without risking significant destabilization of the code.

Table of Contents

What is MySQL HeatWave

MySQL HeatWave is a fully managed database service, powered by the integrated HeatWave in-memory query accelerator. It's the only cloud database service that combines transactions, analytics, and machine learning services into one MySQL Database, delivering real-time, secure analytics without the complexity, latency, and cost of extract, transform, and load (ETL) duplication. It's available on Oracle Cloud Infrastructure (OCI), Amazon Web Services (AWS), and Microsoft Azure.

MySQL HeatWave on AWS delivers price performance that is 7X better than Amazon Redshift and 10X better than Snowflake. On a 10 GB TPC-C workload, MySQL HeatWave offers up to 10X higher and sustained throughput compared to Amazon Aurora at high concurrency. With MySQL HeatWave ML, developers and data analysts can build, train, deploy, and explain machine learning models in MySQL HeatWave without moving data to a separate machine learning service. For machine learning, MySQL HeatWave on AWS is 25X faster than Redshift ML.

Learn more about MySQL HeatWave

Before you start

1. Using the method outlined in this migration guide, where you export your source database and then import it into MySQL HeatWave, there will be some downtime involved. The length of the downtime will mostly depend on the size of your database and checks you may want to perform before bringing your database back online.

2. You must have an account on Oracle Cloud Infrastructure (OCI) and be able to log in to it at https://cloud.oracle.com/
    - If you do not have an account on OCI, you can create one at https://www.oracle.com/mysql/free/

3. You must have enabled "MySQL HeatWave on AWS service" from the OCI Console.
    - For instructions on how to enable MySQL HeatWave on AWS from OCI, refer to the documentation https://dev.mysql.com/doc/heatwave-aws/en/heatwave-aws-sign-up.html

I. Preparing your AWS environment

Section A: Prerequisites

1. To migrate using the method that is shown in this guide, you will need a source Amazon Aurora MySQL instance based on MySQL 5.7 or above. For this guide, we have chosen an Amazon Aurora MySQL 5.7.12.
   - When applicable, you should always execute the commands shown in this guide as a root/admin user.

   You can view the Amazon Aurora MySQL version that is being used for this guide as shown in the image below:

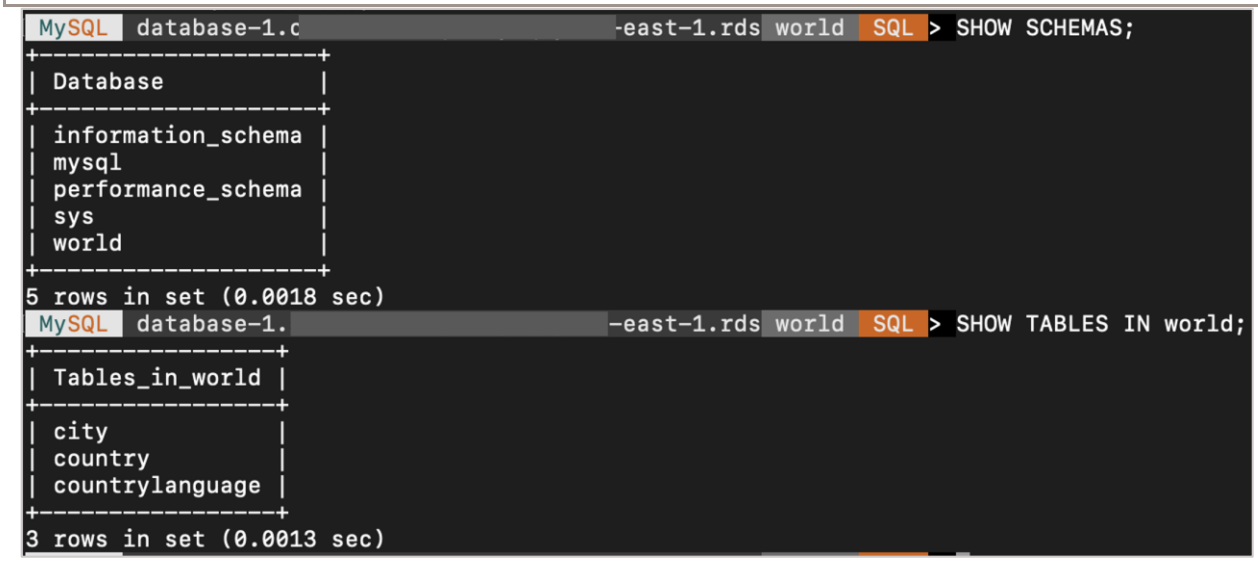
   You can check what version of MySQL you are using by logging into your Amazon Aurora MySQL Server and execute:

```
mysql> SELECT @@VERSION;
```

2. For this guide, we have some data pre-loaded on our Amazon Aurora MySQL database.
   - The sample data used in this guide is the 'world' database, which can be downloaded from here: https://dev.mysql.com/doc/index-other.html.

   You can see a list of all the databases on your Amazon Aurora MySQL Server and the tables in the world database as shown below. We will export the world database from Amazon Aurora MySQL to MySQL HeatWave on AWS.

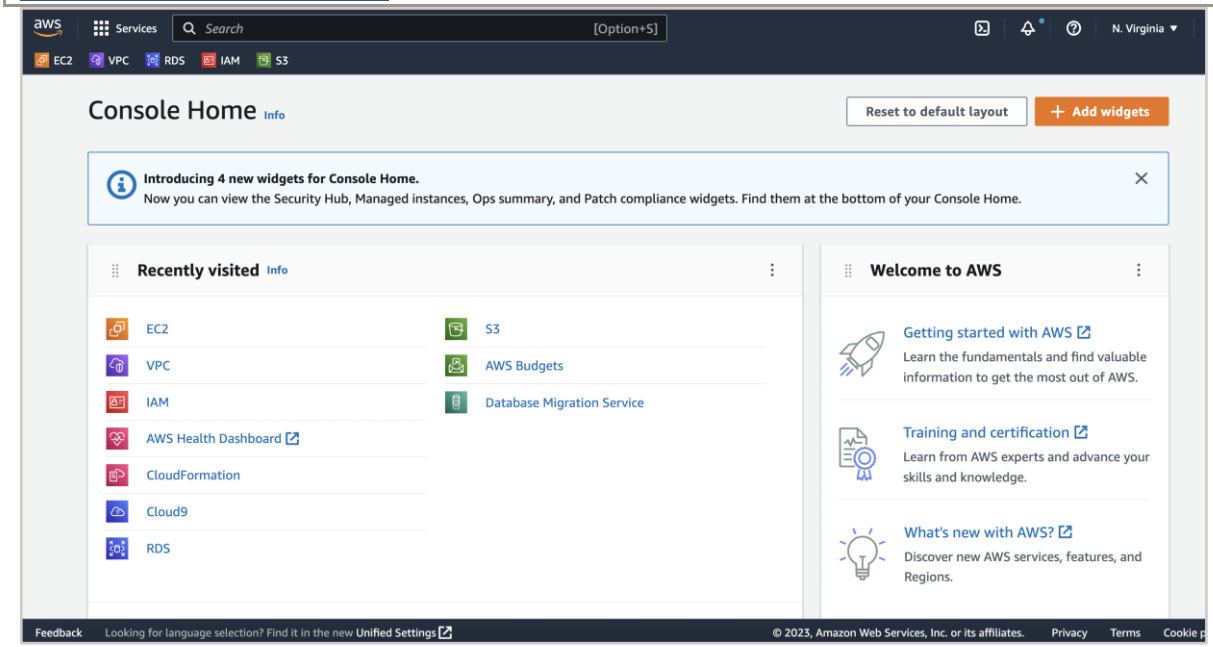
```
mysql> SHOW DATABASES;
mysql> SHOW TABLES IN world;
```



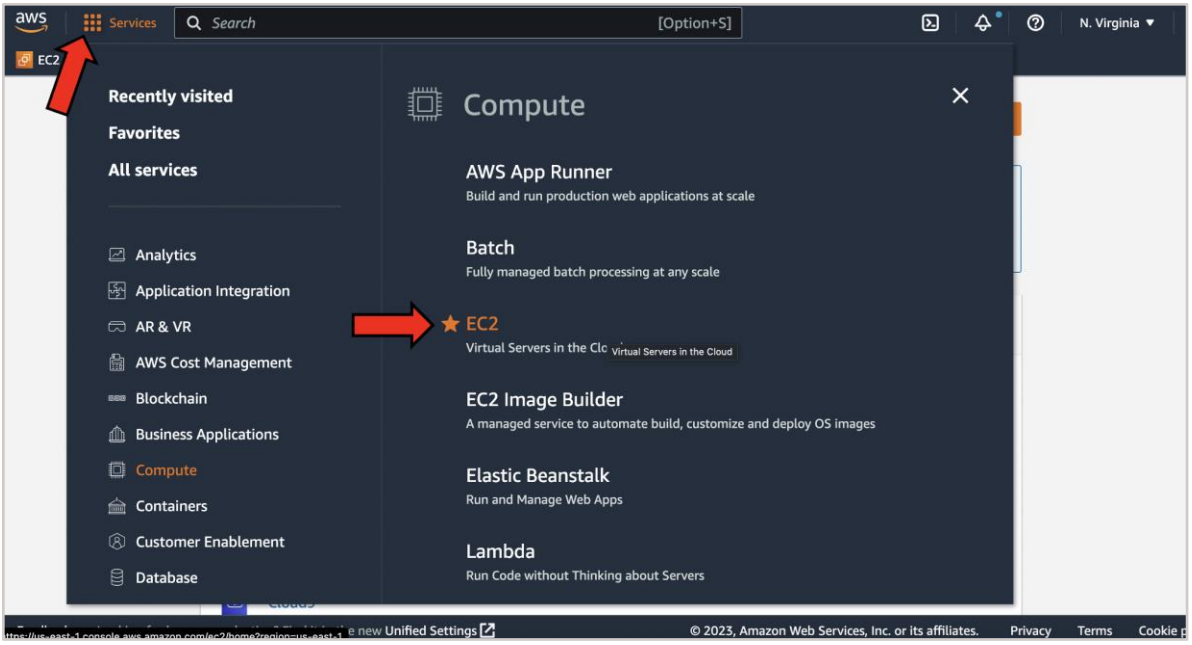
Section B: Create an EC2 Instance and configure your SSH keys

3. Login to your AWS account.

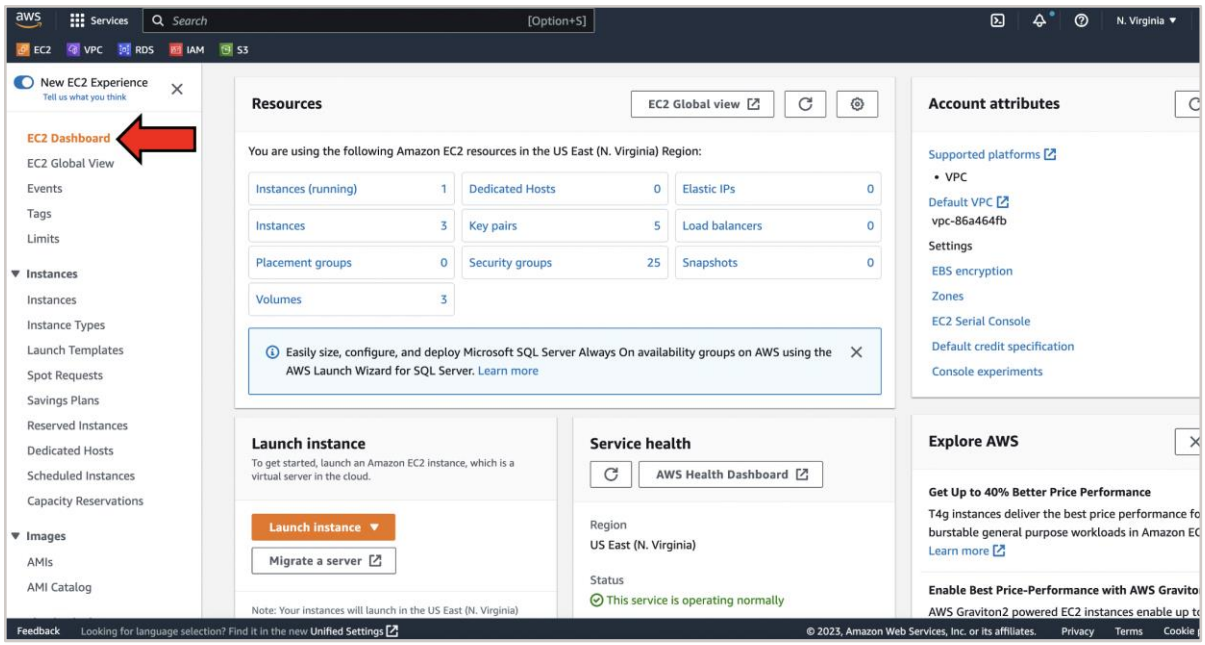
```
https://aws.amazon.com/
```

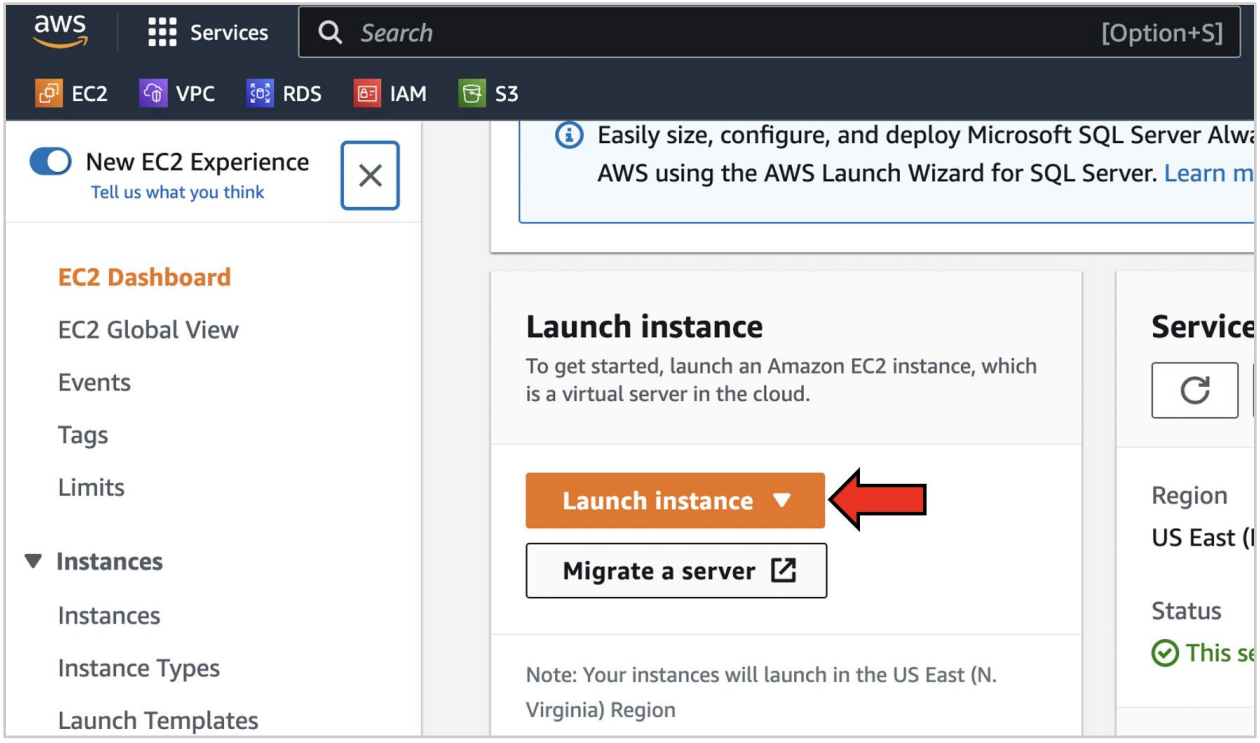


4. Click on the "Services" menu and go to "Compute" > "EC2"

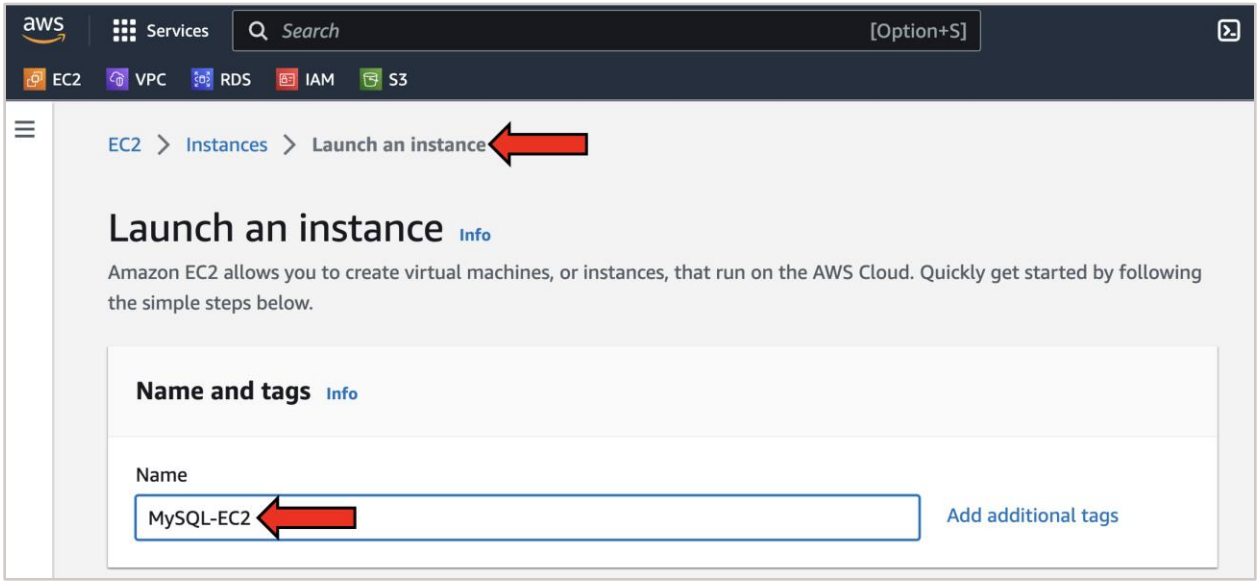


5. On the “EC2 Dashboard” page, look for the “Launch instance” button.

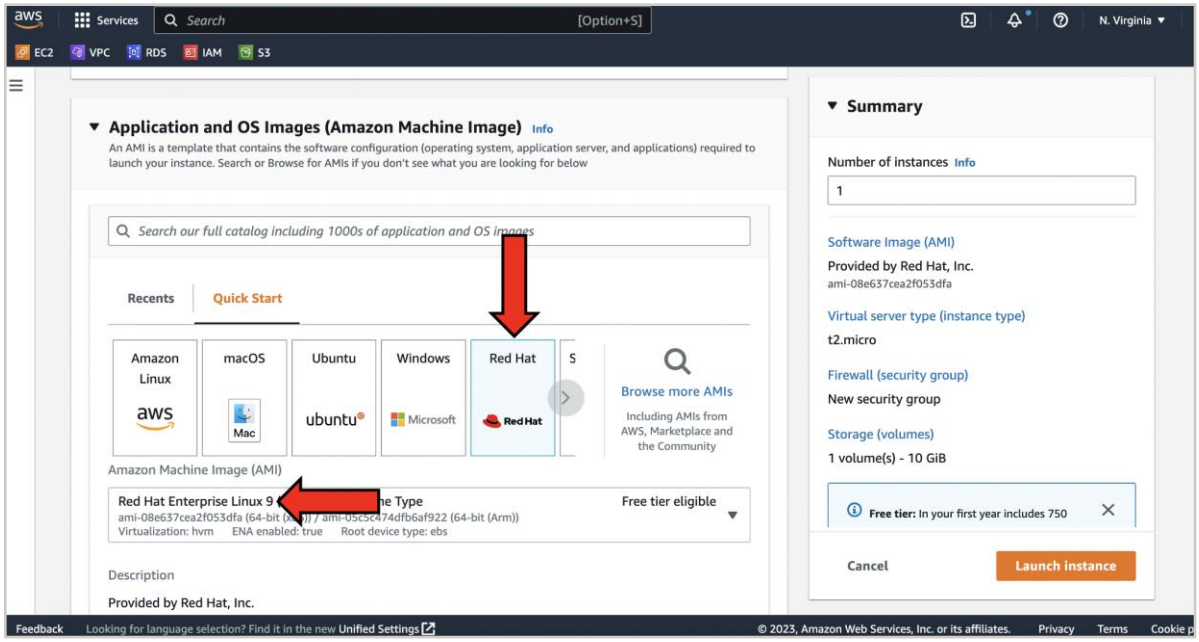




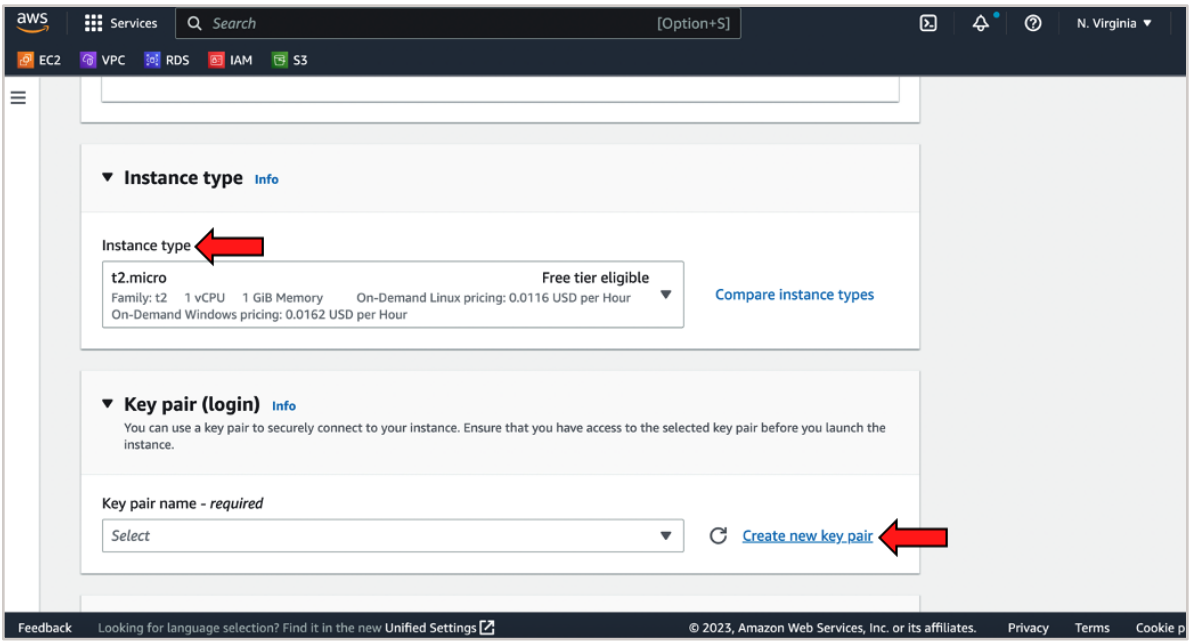
6. Click “Launch instance”. When the “Launch an instance” page opens, enter a name for your EC2 Instance. For this guide, we have chosen “MySQL-EC2”



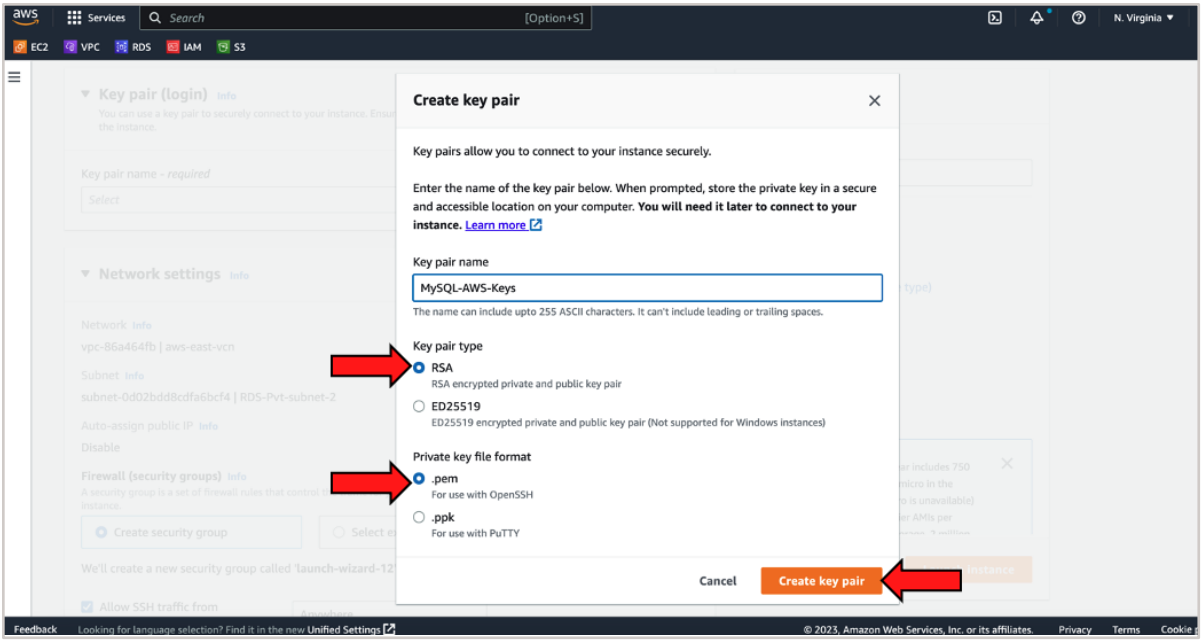
7. For Amazon Machine Image type, choose “Red Hat” and either version “Linux 8 or 9”. In the example below, we have selected Linux 9.



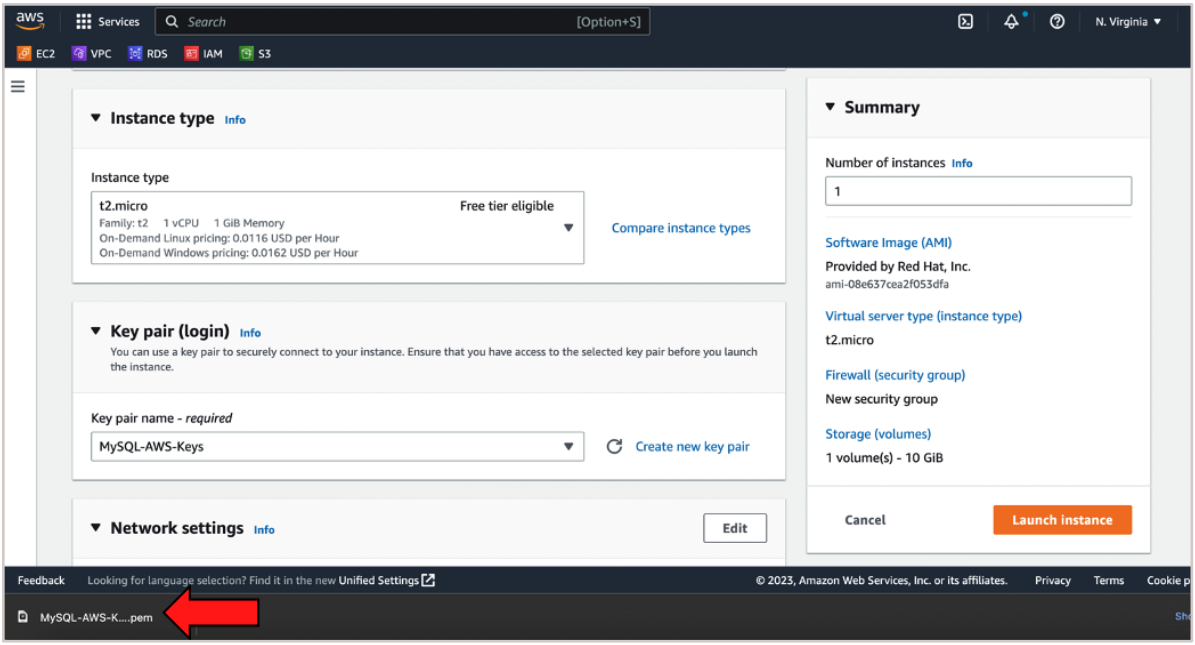
8. For “Instance type”, select one that suits your needs. Afterwards for the “Key pair” section, click on “Create new key pair”. You can also use your existing keys here.



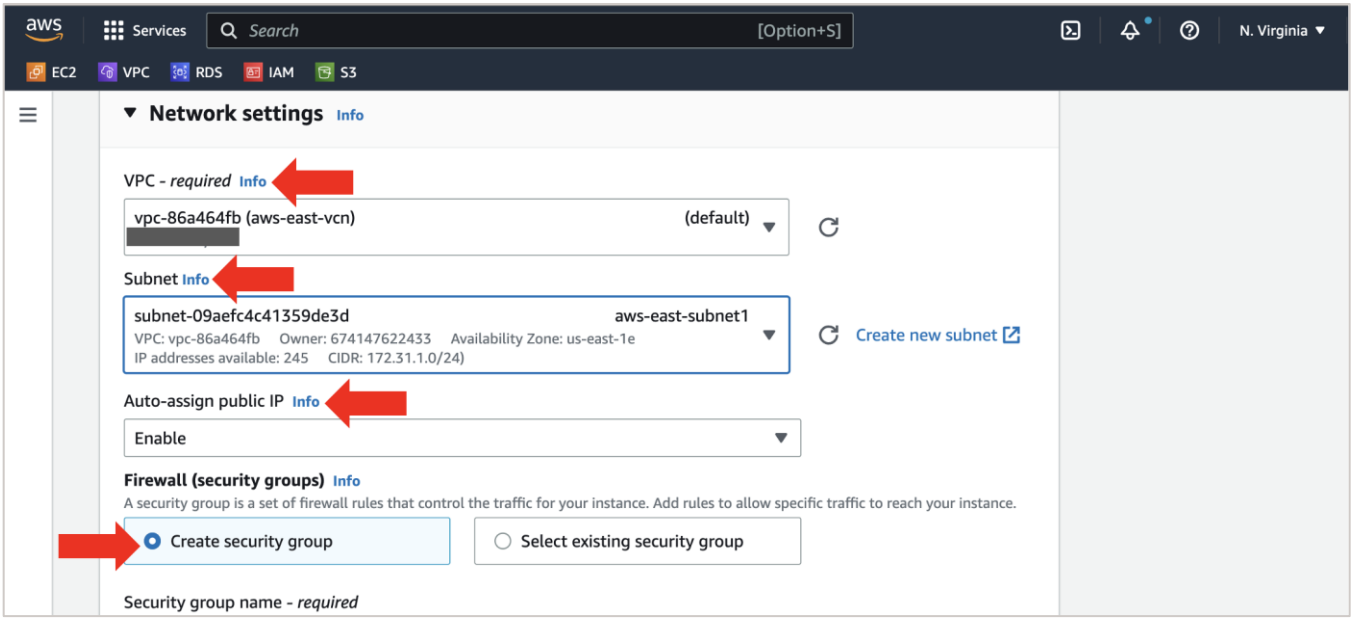
When you click “Create new key pair”, a popup will appear asking you to “Create key pair”. Give a name for your Key pair and make sure “RSA” is selected under the “Key pair type”. Under “Private key file format”, select “.pem”.

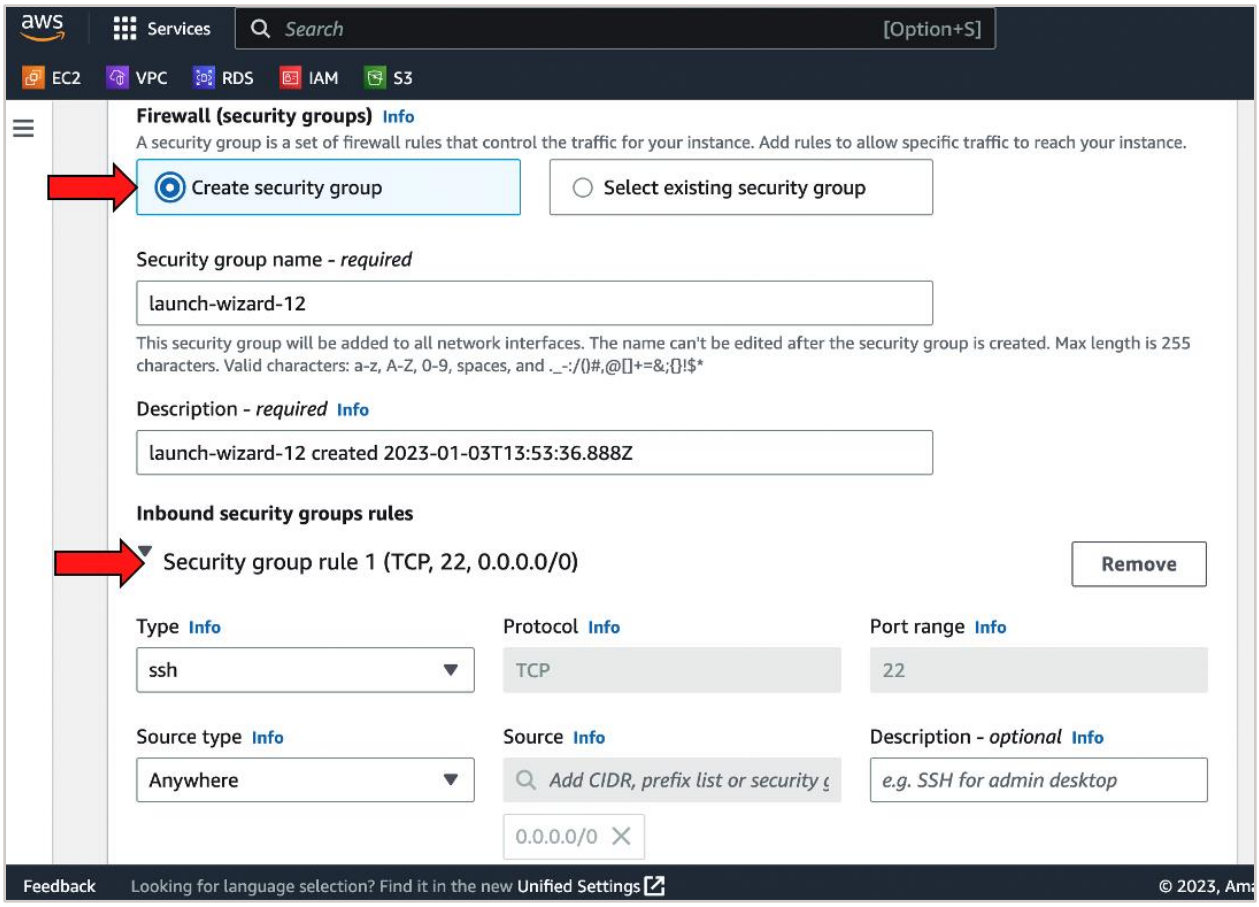


- Note: click “Create key pair” afterwards. This will close the “Create key pair” popup and will download a private SSH Key. Look below:

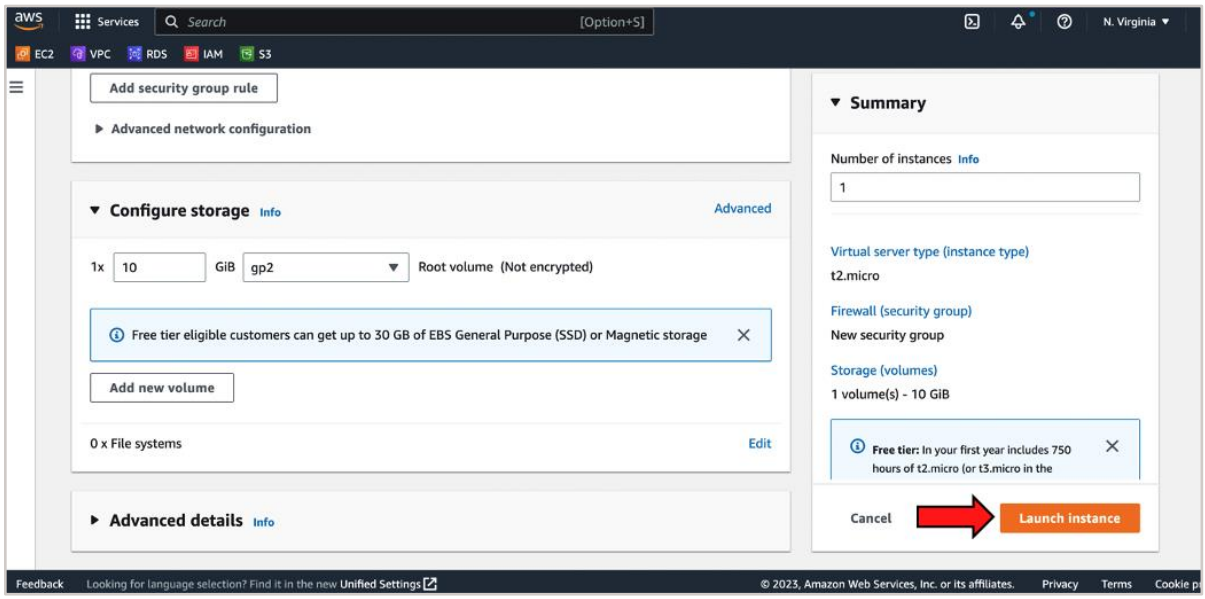


9. For your “Network settings”, select your appropriate “VPC” and “Subnet”. For “Auto-assign public IP” select “Enable”. Under the “Firewall (security groups)” tab, choose “Create security group” and have an “Inbound security group rules” like the below one which allows SSH from anywhere.

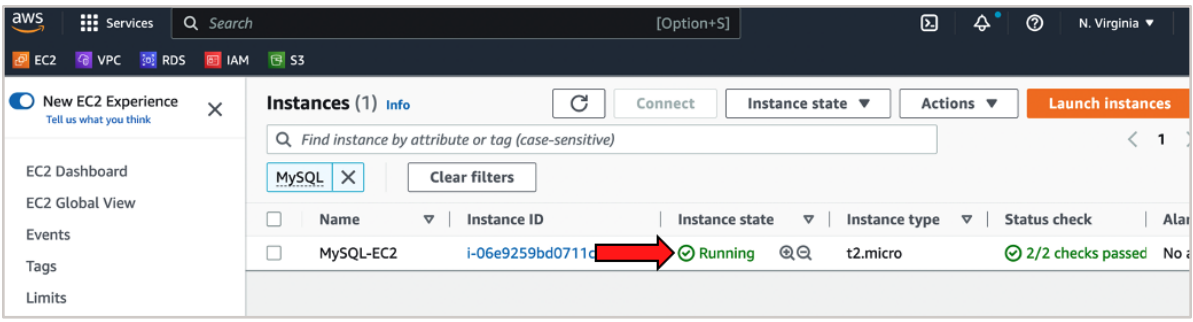




10. Once that is done, leave everything as default and click “Launch instance”

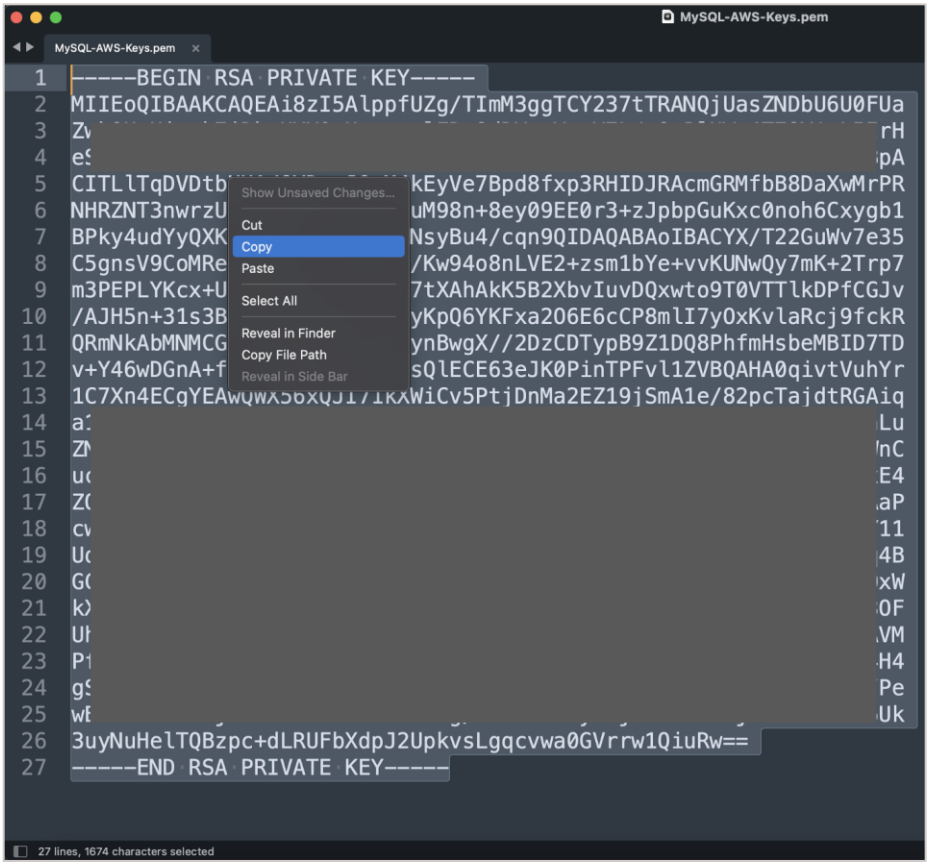


11. You will have to wait until your MySQL-EC2 “Instance state” is “Running” before you can connect to it.



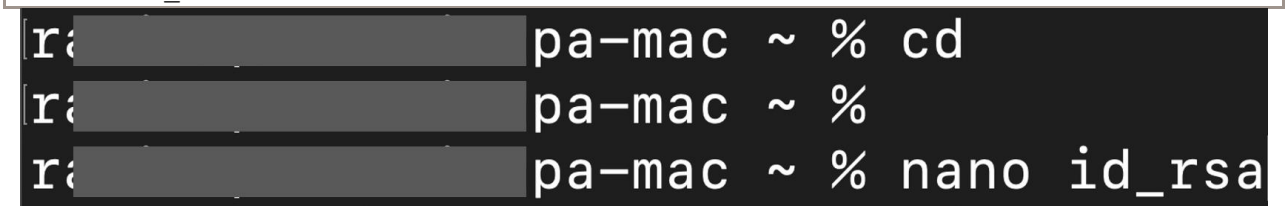
12. Once your EC2 instance is in a “Running” state, open the Private SSH Key that you downloaded in Step 8 in a text editor of your choice.

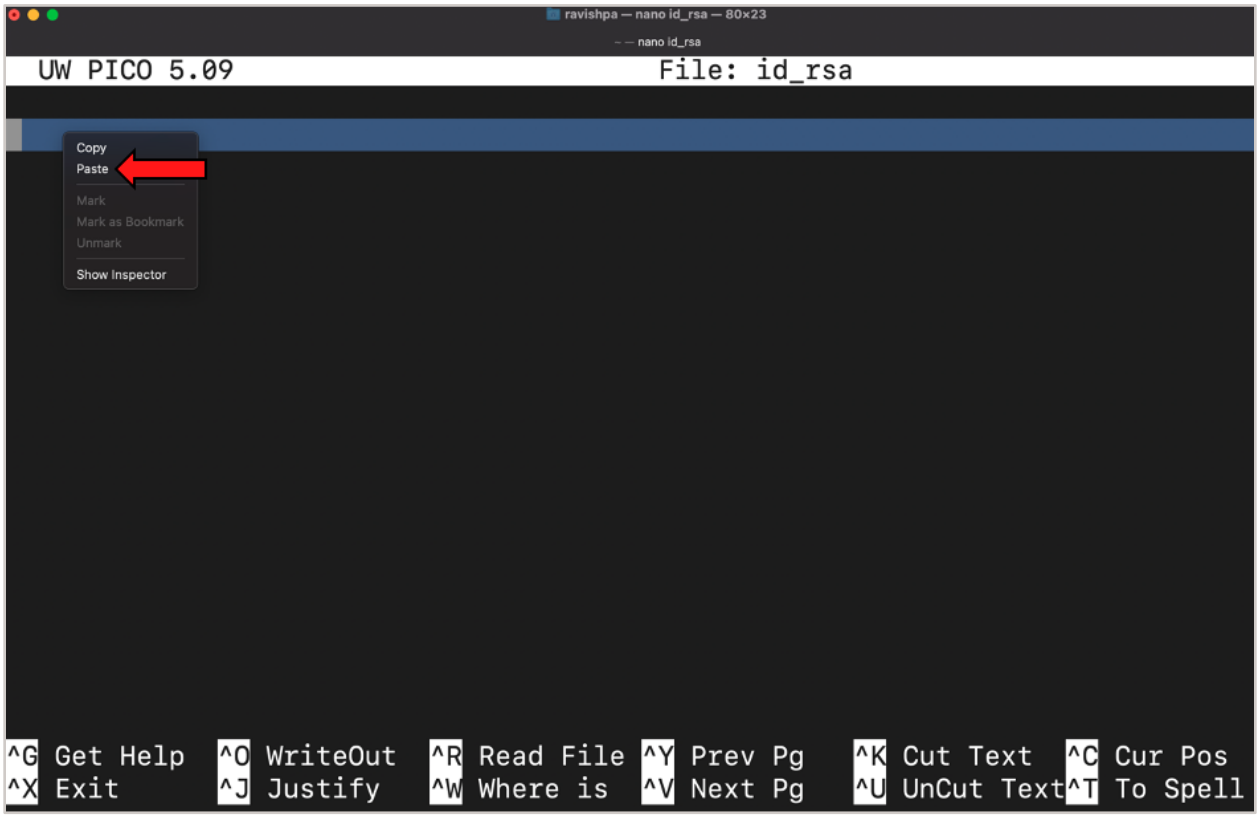
13. Once you have opened your Private SSH Key in a text editor, copy the contents of the entire file as shown below:



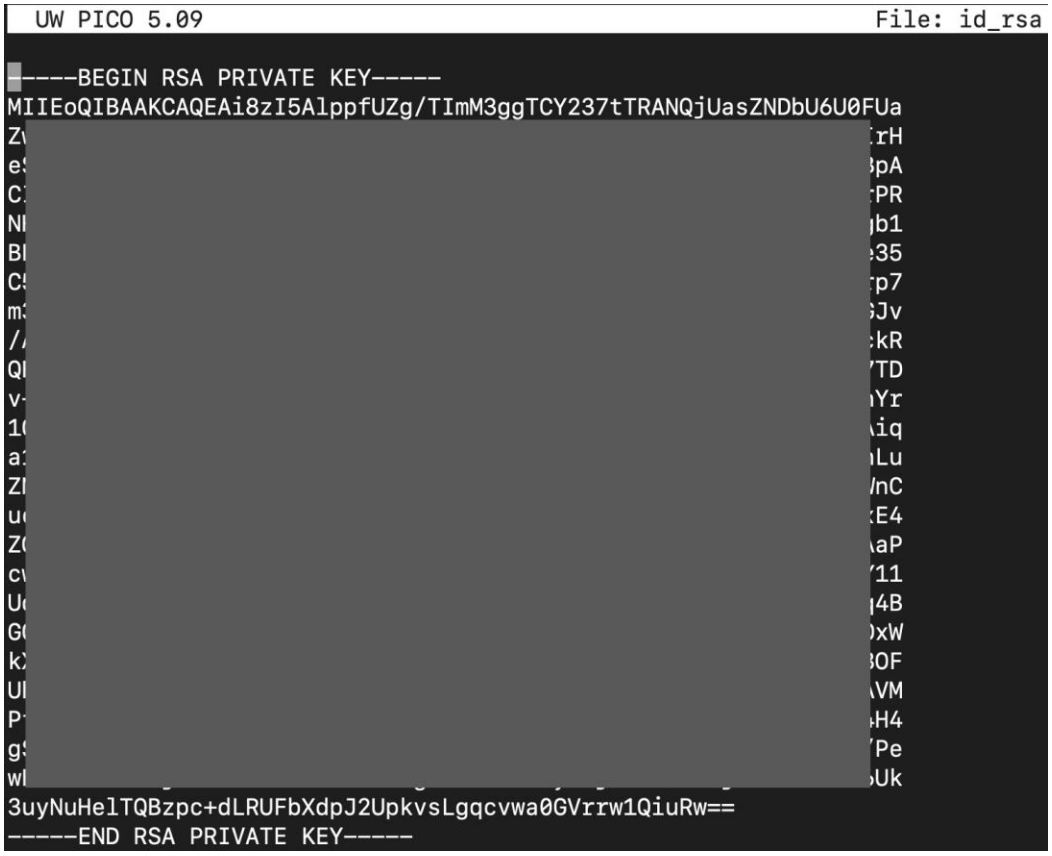
14. After copying the contents, to connect to your EC2 instance, go to your terminal where you will be accessing EC2 from. There, create a new file called `id_rsa` inside your home directory. The guide uses the “nano” text editor, use a text editor of your own choice.

```
$ cd
$ nano id_rsa
```



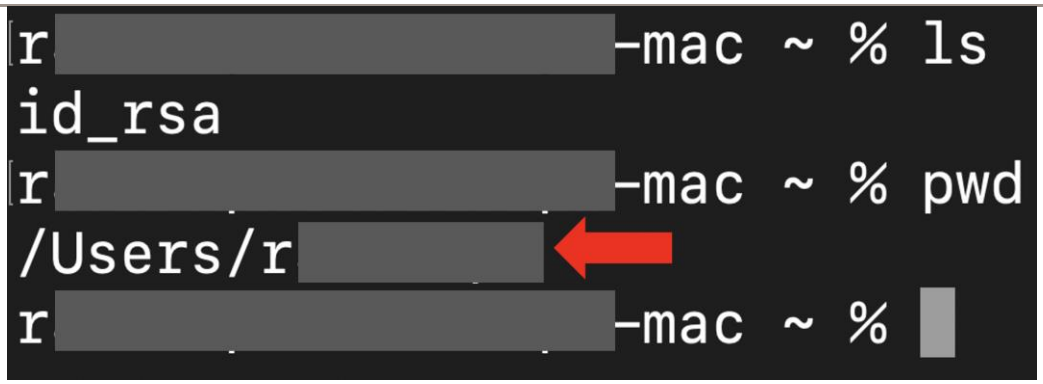


15. After pasting the contents of the private SSH key into the `id_rsa` file, save and close the file. If you are using nano:
    - to paste the copied content: `command + V`
    - to save the file: `control + O`
    - to exit the file: `control + X`



16. After you have saved the private SSH Key on your terminal, grab the file path of the `id_rsa`. To get the file path of your current working directory where you have the `id_rsa`, execute:

```
$ ls
$ pwd
```



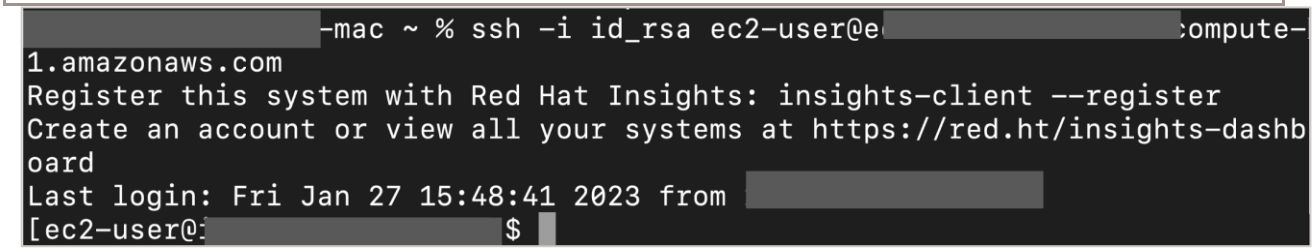
- Note: by looking at the above image, the `id_rsa` location for this guide will hence be `/Users/r***/id_rsa`

17. Once you have your SSH Key copy and pasted, make sure to change the Private SSH key's permission by executing:

```
$ chmod 400 id_rsa
```

18. You can now connect to the EC2 Instance you created earlier by executing the following from your terminal window where you have the SSH keys:

```
ssh -i <path/to/you-private-ssh-key> ec2-user@<ec2-Public-DNS>
```



- Note: after executing the above SSH command, when prompted "Are you sure you want to continue connecting (yes/no/[fingerprint])?", type "yes".

19. You are now successfully connected to your EC2 instance.

Section C: Connect to your EC2 Instance and install MySQL Shell

20. Once you have identified your Amazon Aurora MySQL version and the data you want to migrate, go to your AWS environment and connect to the EC2 instance you created in Section B. You now need to install MySQL Shell on your EC2 instance. You will use MySQL Shell to export the world database and import it into MySQL HeatWave. (MySQL Shell is an advanced client and code editor for MySQL. To learn more about MySQL Shell, visit: https://dev.mysql.com/doc/mysql-shell/8.0/en/)

    Installing MySQL Shell on Microsoft Windows:
    To install MySQL Shell on Microsoft Windows using the MSI Installer, perform the following steps:

    a) Download the Windows (x86, 64-bit), MSI Installer package from http://dev.mysql.com/downloads/shell/

    b) When prompted, click Run.

    c) Follow the steps in the Setup Wizard.

    Installing MySQL Shell on Linux:
    To install MySQL Shell on Linux, run the following command:

    ```
    sudo yum install mysql-shell
    ```

    For other Linux installation options, visit: https://dev.mysql.com/doc/mysql-shell/8.0/en/mysql-shell-install-linux-quick.html

    Installing MySQL Shell on macOS:
    To install MySQL Shell on macOS, perform the following steps:

    a) Download the package from http://dev.mysql.com/downloads/shell/.

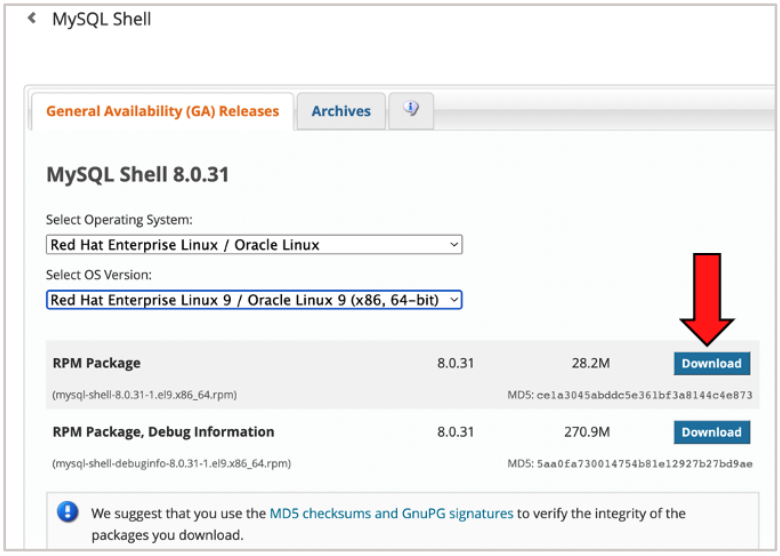
    b) Double-click the downloaded DMG to mount it. Finder opens.

    c) Double-click the .pkg file shown in the Finder window.

    d) Follow the steps in the installation wizard.

    e) When the installer finishes, eject the DMG (It can be deleted).

This is how the guide installed MySQL Shell, visit: https://dev.mysql.com/downloads/shell/. Select the latest version of the MySQL Shell and select the appropriate OS System and Version. For this guide, Red Hat Enterprise Linux 9 server is being used for the EC2 instance.

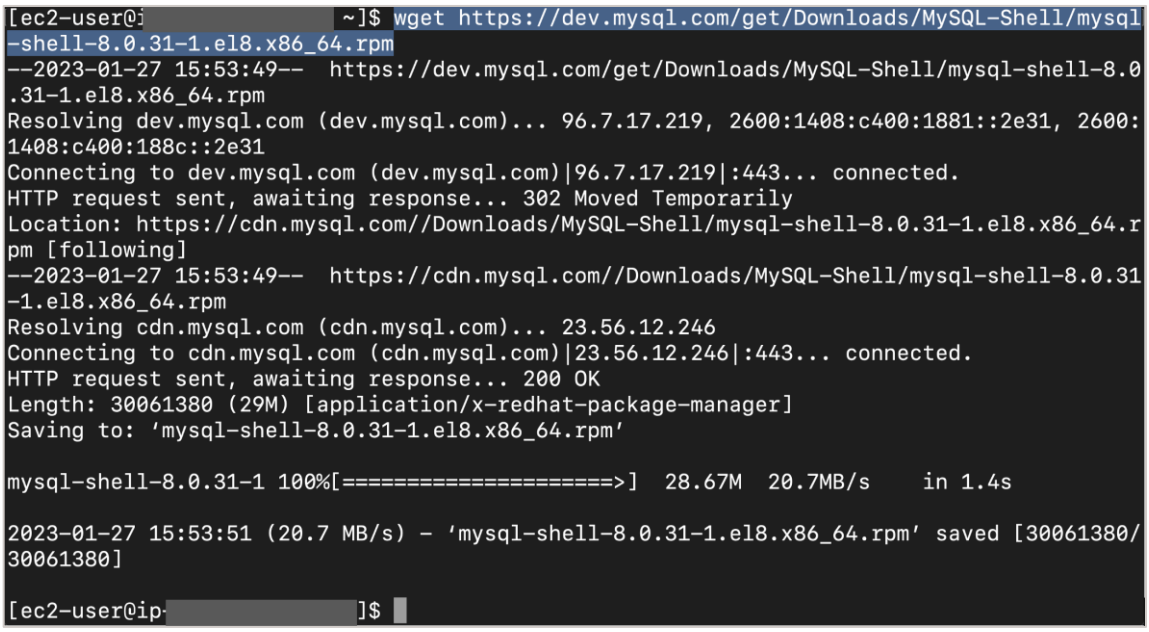


- Note: the `RPM Package` (28.2M), without the debug information was chosen for this guide. Once you have identified which MySQL Shell version you want to download, click on the "Download" button shown in the above image. A new page will popup, which is shown in the next step.

21. When you click "Download" as shown in Step 20, this page will come up. Right click on "`No thanks, just start my download.`" and select "`Copy Link Address`"

22. Go back to your AWS EC2 instance and download MySQL Shell via `wget` by pasting the link copied in the previous step. But first, download `wget` itself

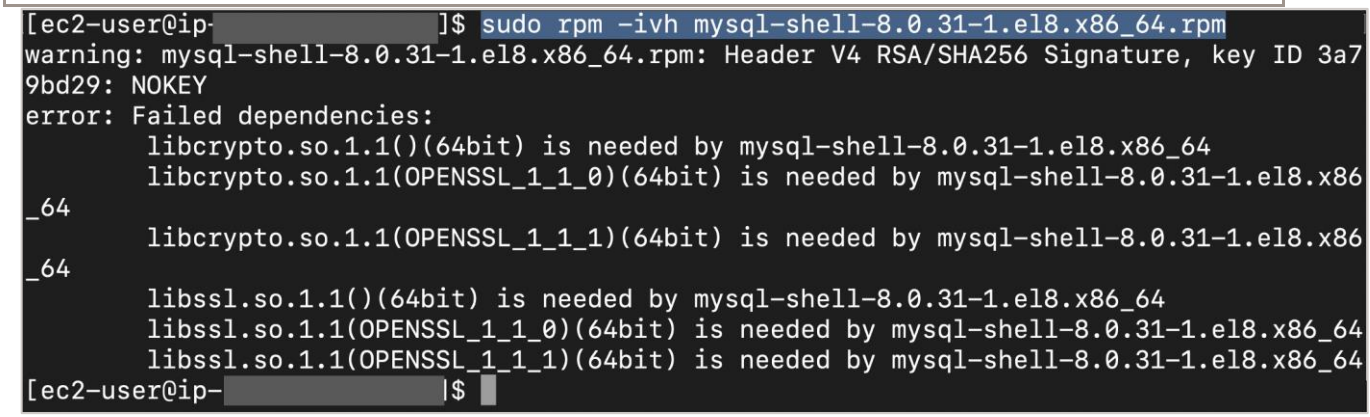
```
$ sudo yum instal wget -y
$ wget https://dev.mysql.com/get/Downloads/MySQL-Shell/mysql-shell-8.0.31-
1.el8.x86_64.rpm
```



- Note: download and install MySQL Shell by using the proper commands/files/methods required for your own Operating System.

23. Once MySQL Shell RPM file is downloaded on your your EC2 instance, extract it using

```
sudo rpm -ivh <file-name>
```



- Note: there were missing dependences when the rpm command was executed

24. To resolve the above dependency, run the following command:

```
sudo yum install compat-openssl11
```

25. Once all the required dependencies are installed, execute the same rpm command from Step 23

```
sudo rpm -ivh <file-name>
```

```
[ec2-user@ip-                ]$ sudo rpm -ivh mysql-shell-8.0.31-1.el8.x86_64.rpm
warning: mysql-shell-8.0.31-1.el8.x86_64.rpm: Header V4 RSA/SHA256 Signature, key ID 3a7
9bd29: NOKEY
Verifying...                          ################################# [100%]
Preparing...                          ################################# [100%]
Updating / installing...
   1:mysql-shell-8.0.31-1.el8         ################################# [100%]
[ec2-user@ip-                ]$
```

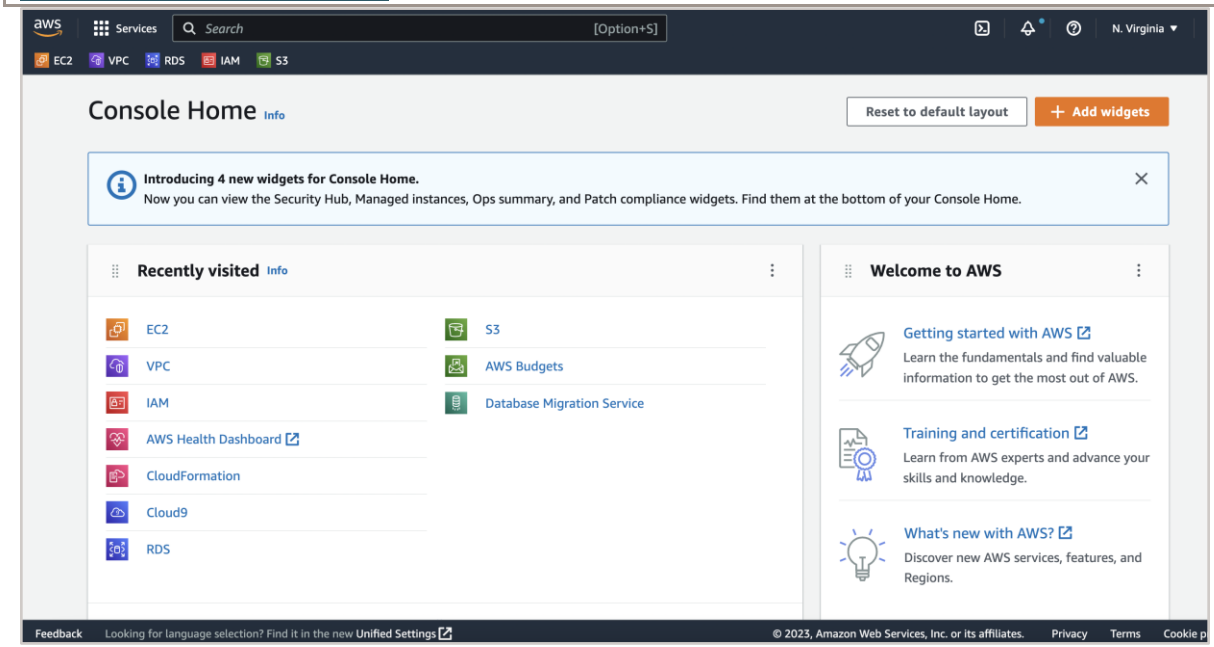
- Note: MySQL Shell was properly installed after all the dependencies were solved

II. Exporting the database

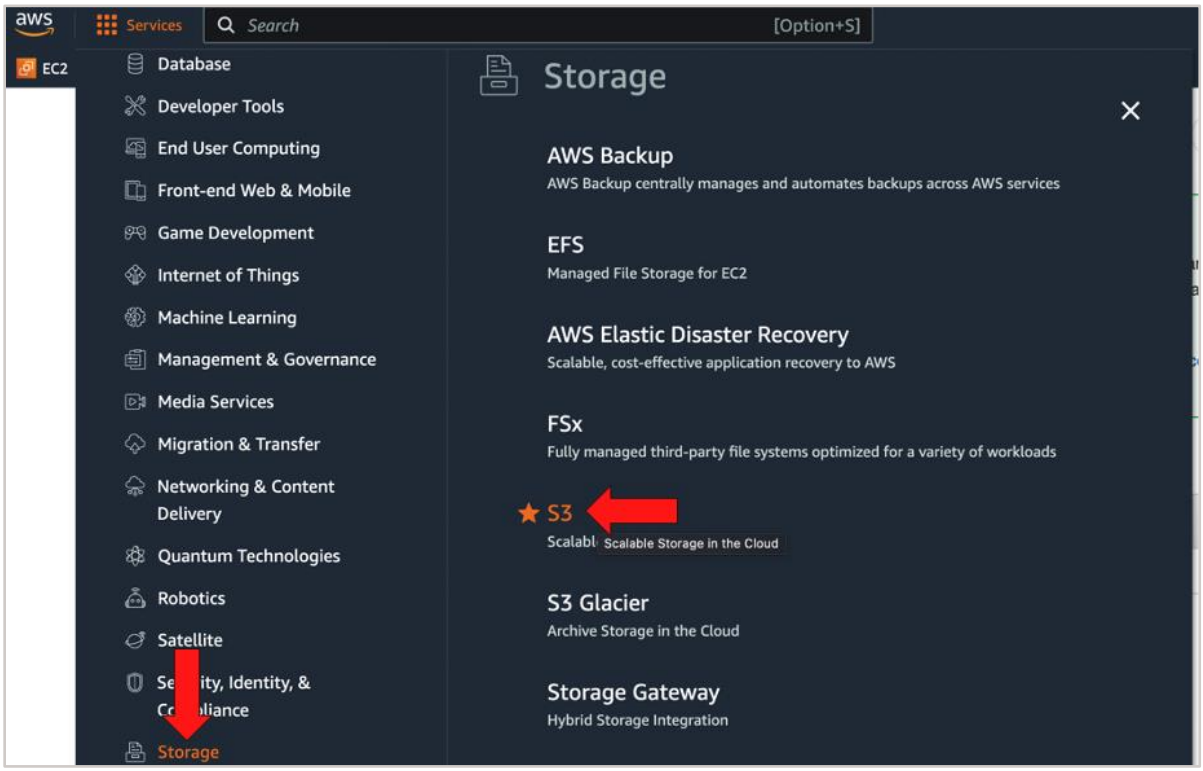
Section D: In AWS, create an S3 Storage Bucket

26. Login to your AWS account.

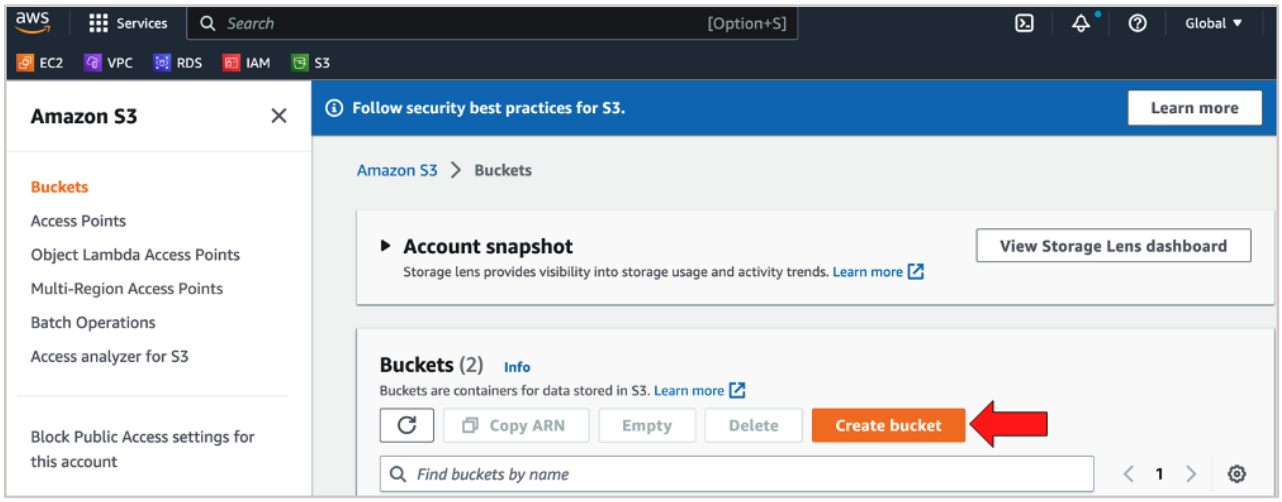
```
https://aws.amazon.com/
```



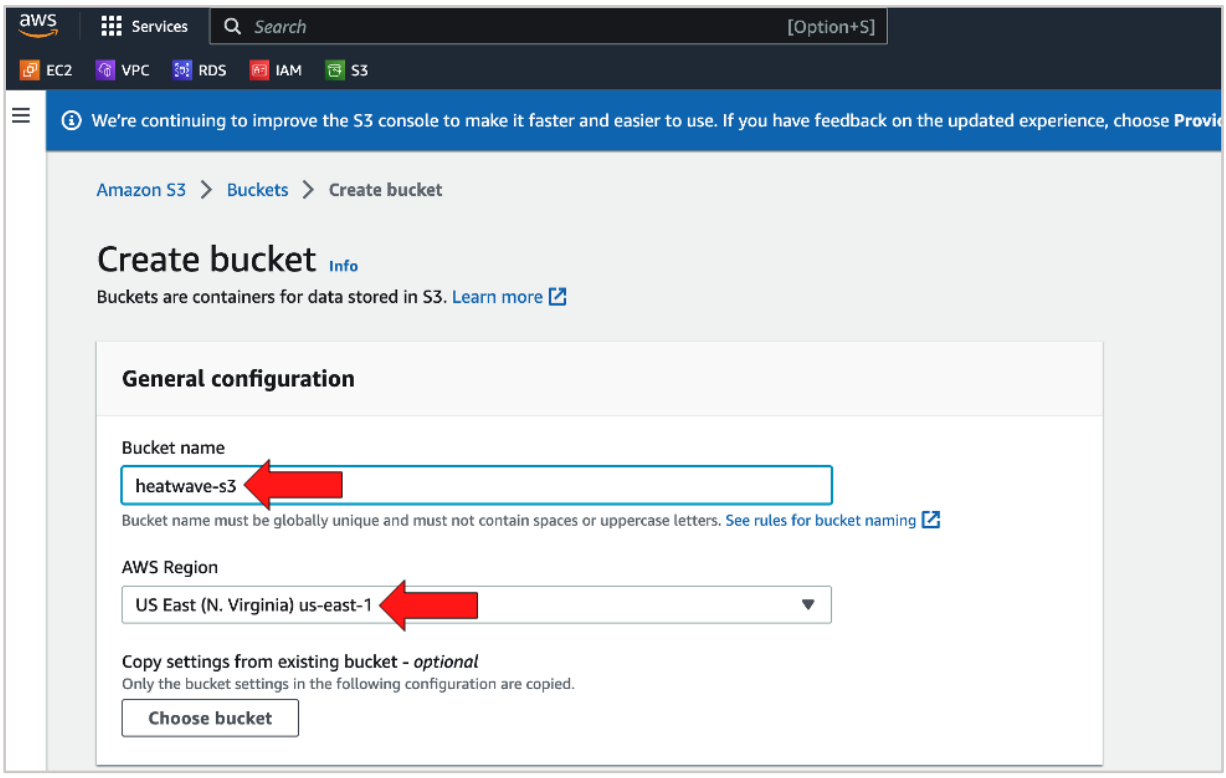
27. Click the 'Services' menu on the top-left corner. From there, navigate to 'Storage' and click on "S3"



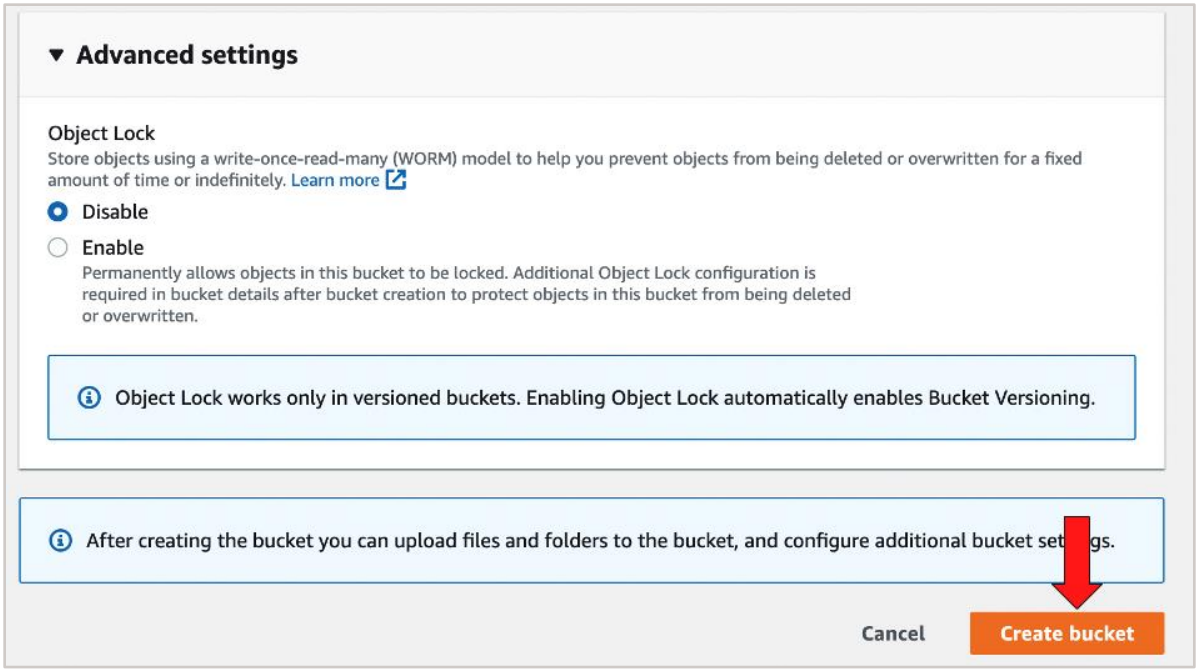
28. Once you are on the ‘S3’ Buckets page, click the “Create bucket” button. In a later step, you will export your Amazon Aurora MySQL database to AWS in this bucket.



29. On the ‘Create bucket’ page, give a name for your bucket and select “US East (N. Virginia)” as the ‘AWS Region’



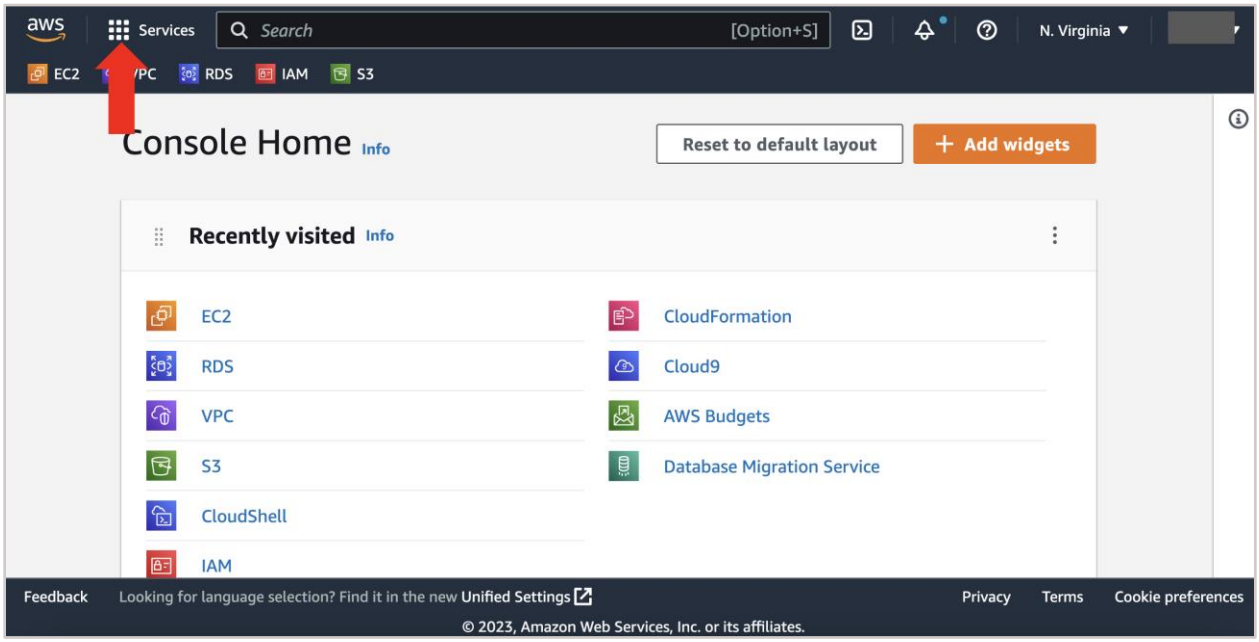
30. Leave the other fields as-is and click the “Create bucket” button.

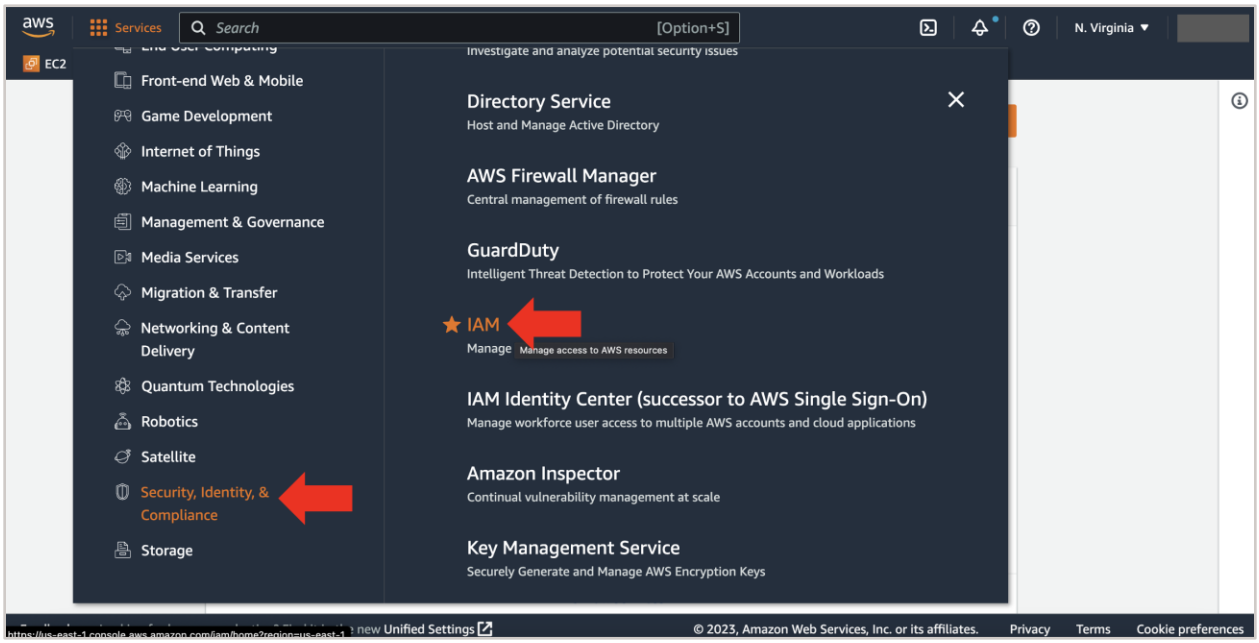


- Note: once the S3 bucket is created, save the bucket name in a notepad for later use.

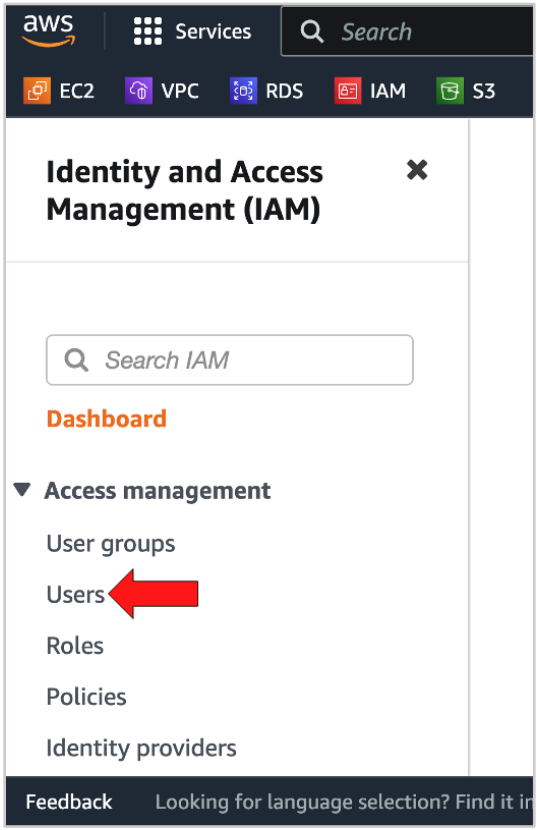
Section E: Add an IAM user and download the .csv file

31. From the AWS Console, navigate to the 'Services' menu. From there, navigate to 'Security, Identity, & Compliance' and look for "IAM"

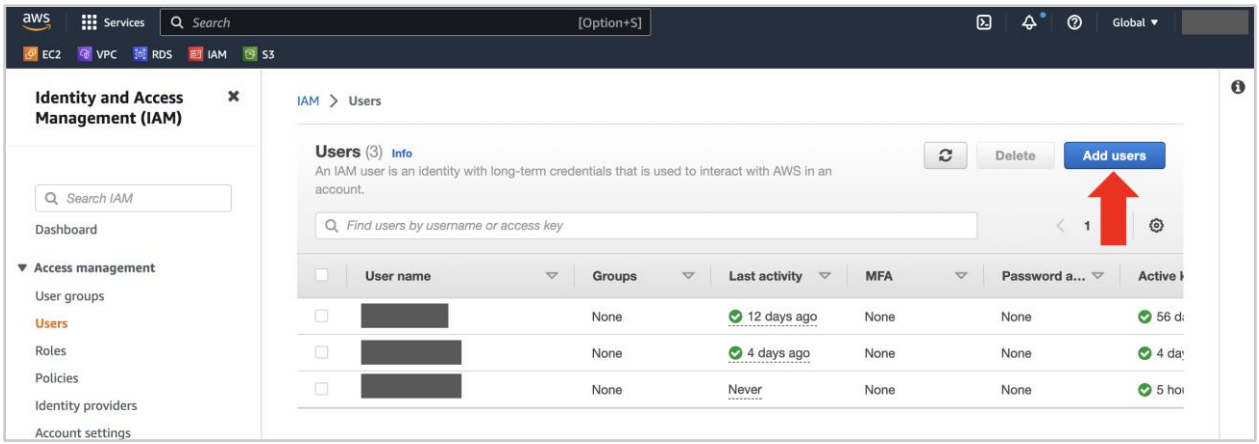




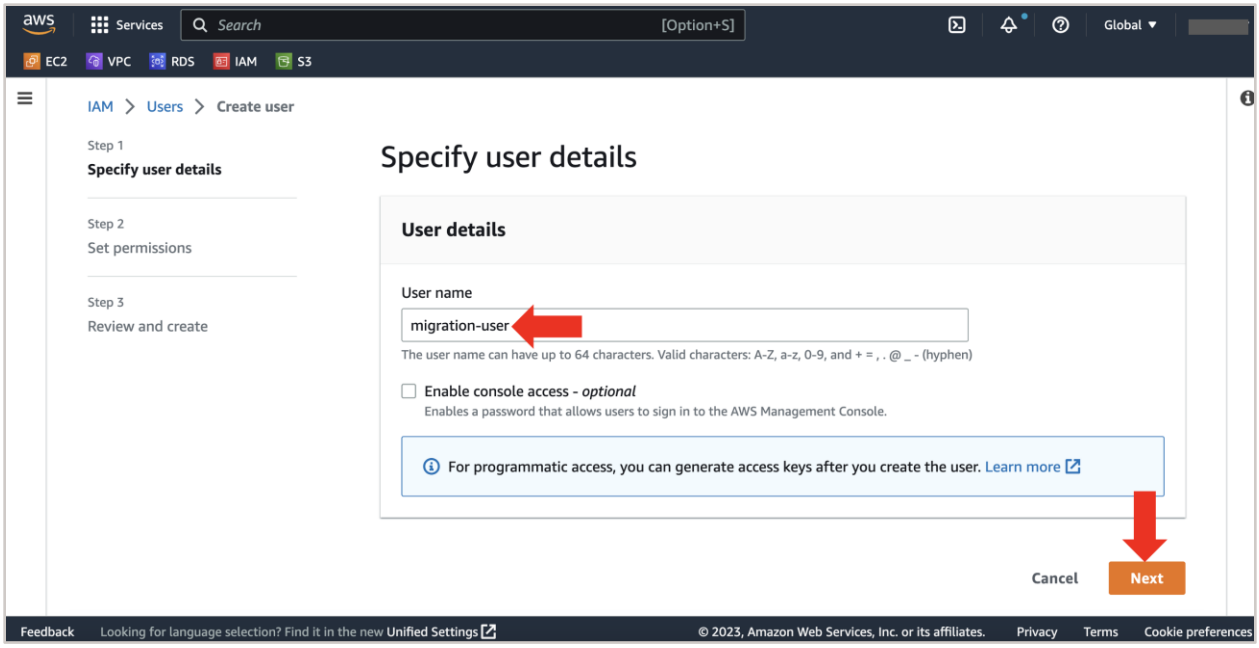
32. From the 'Identity and Access Management (IAM)' dashboard page, click on "Users" under 'Access management'



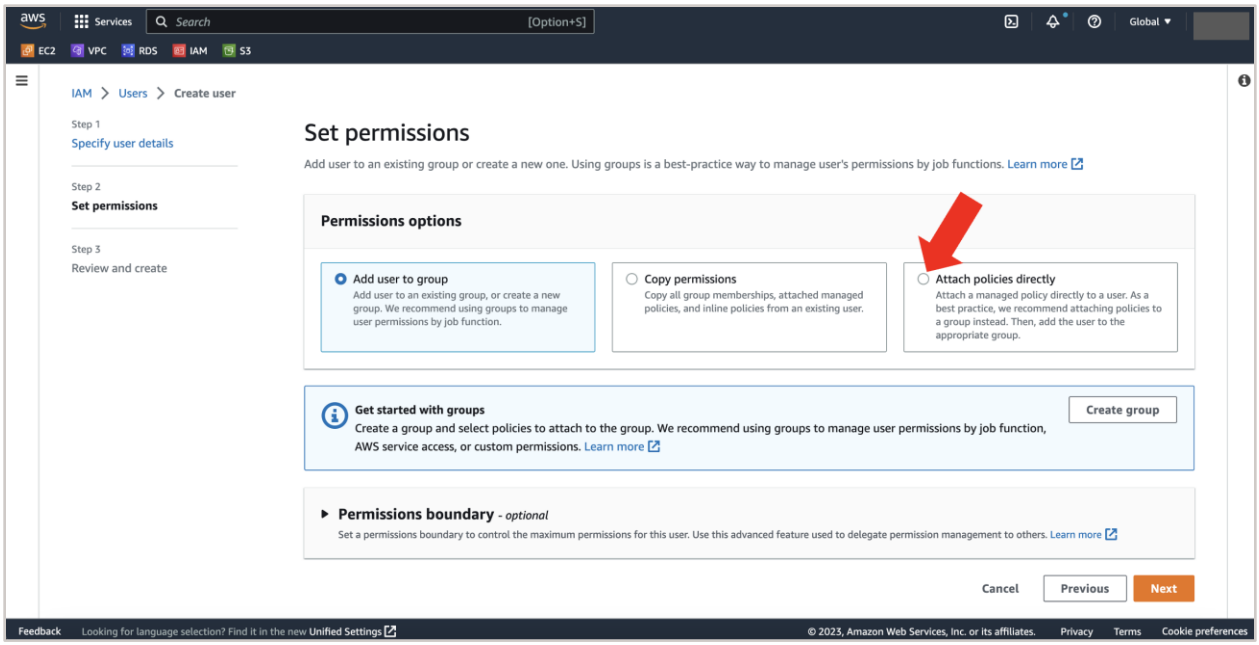
33. After landing on the 'Users' page, click "Add users"

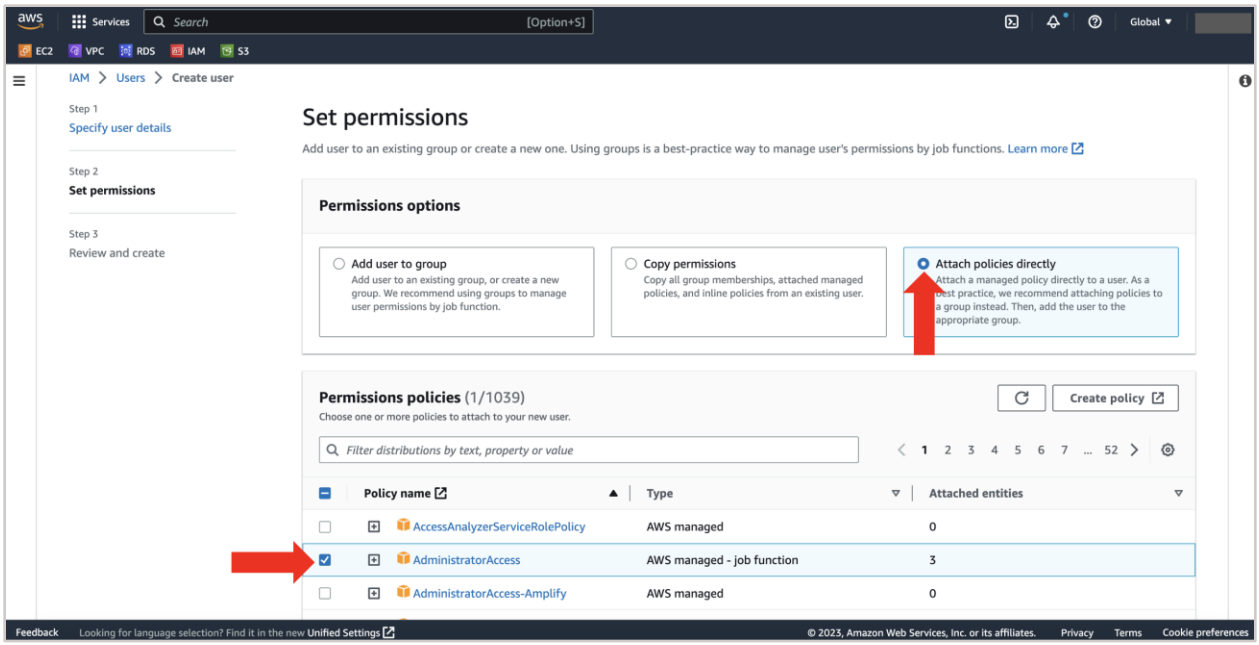


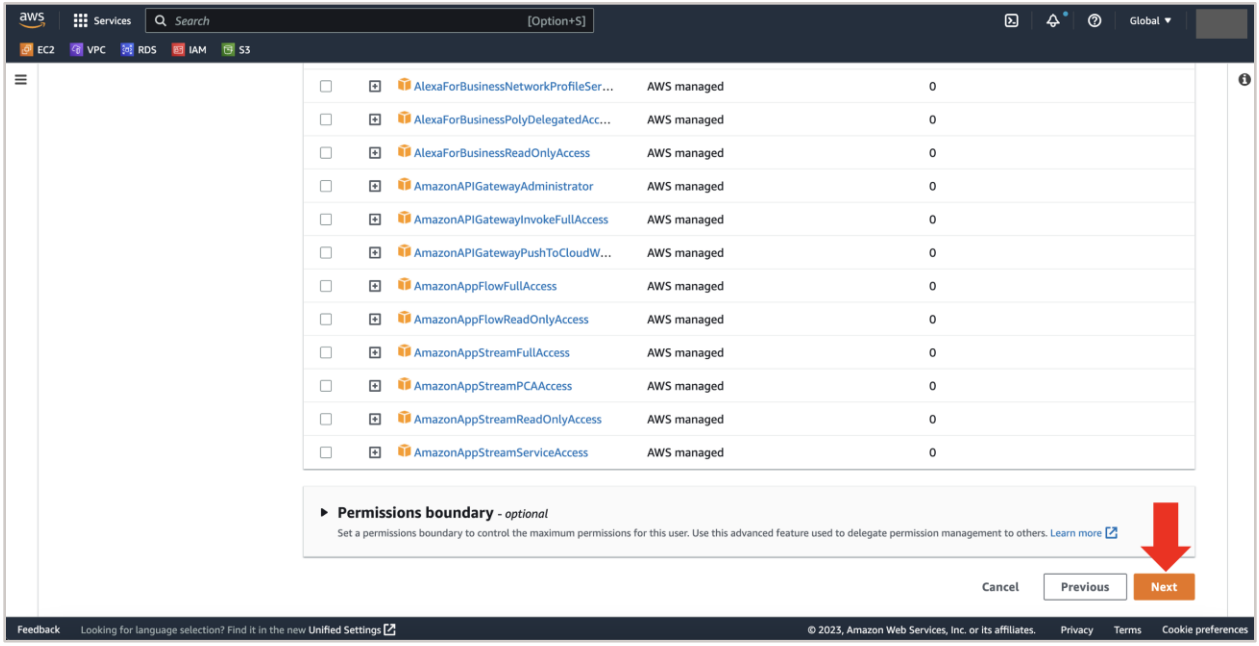
34. On Step 1 of ‘Create user’, enter a ‘User name’. Click “Next” afterwards.



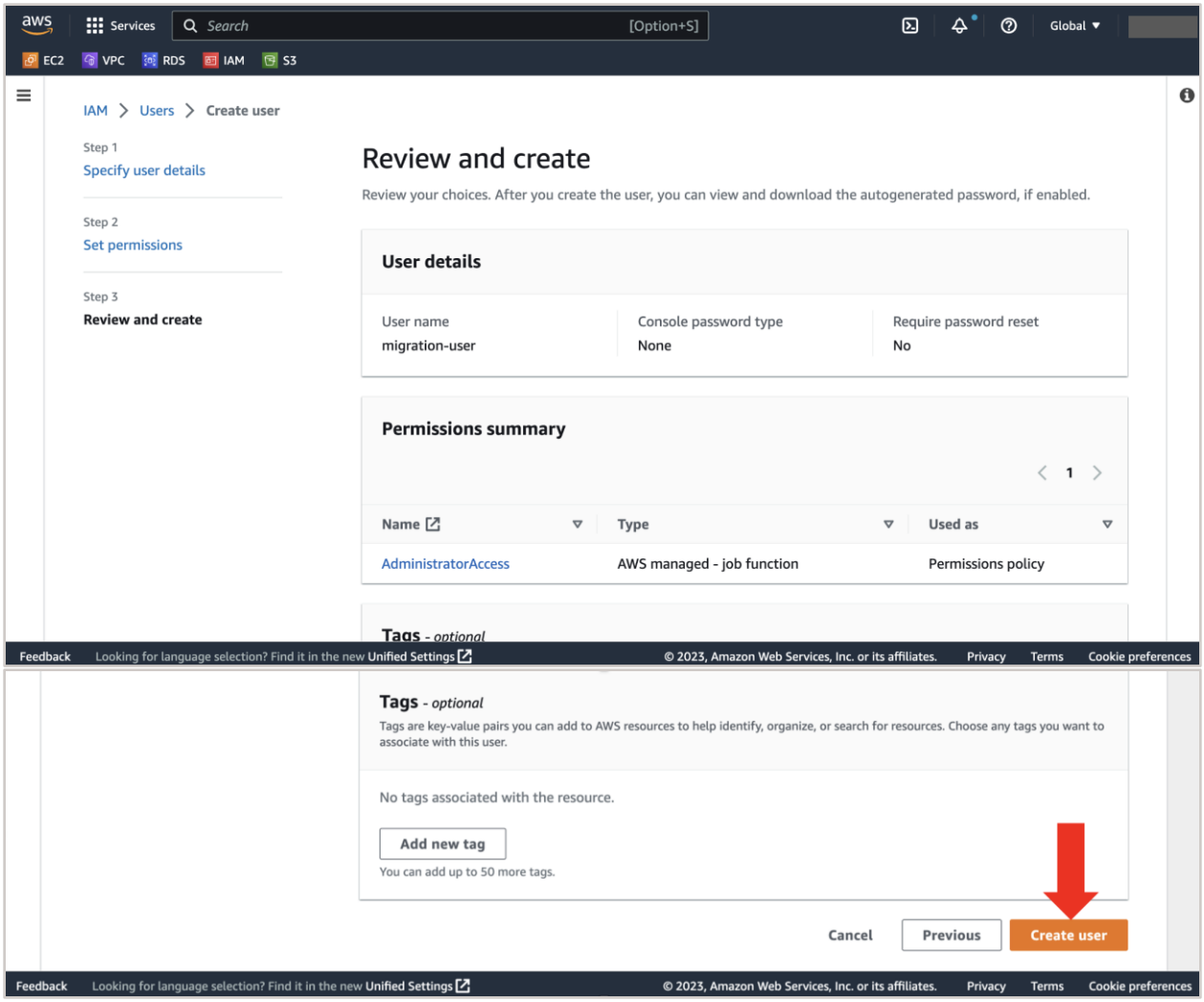
35. On Step 2, click “Attach policies directly” and select the ‘AdministratorAcess’ policy. Leave everything as it is and click “Next” afterwards



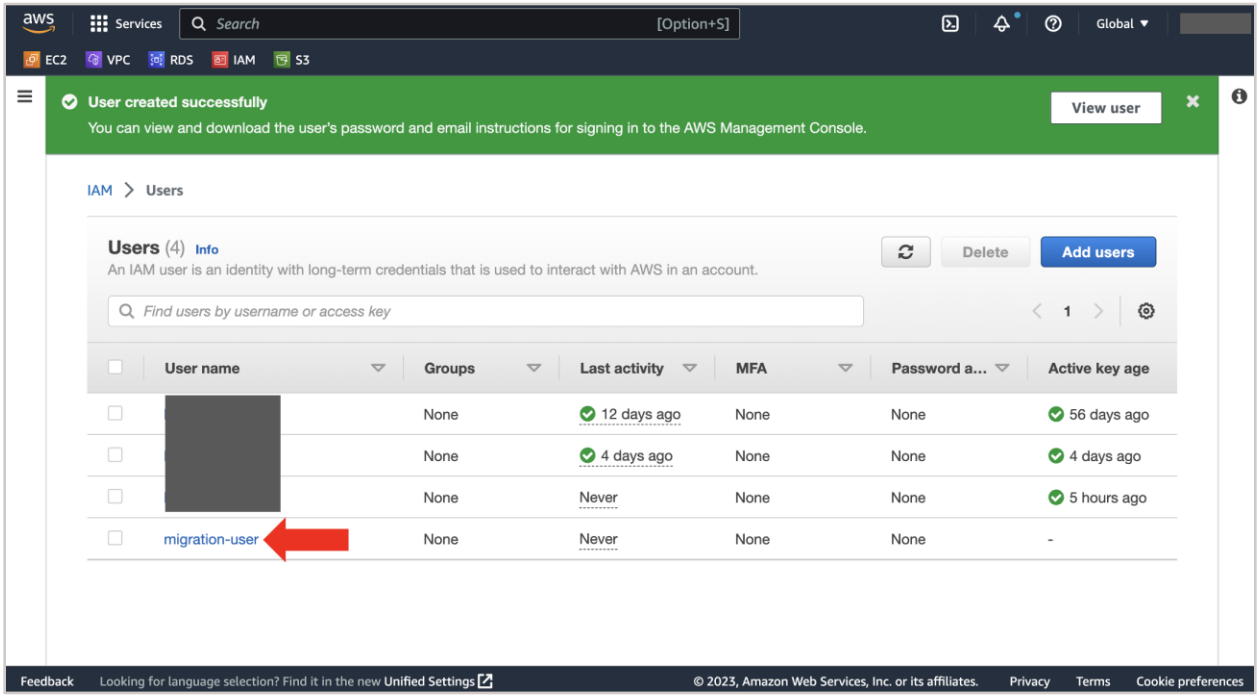




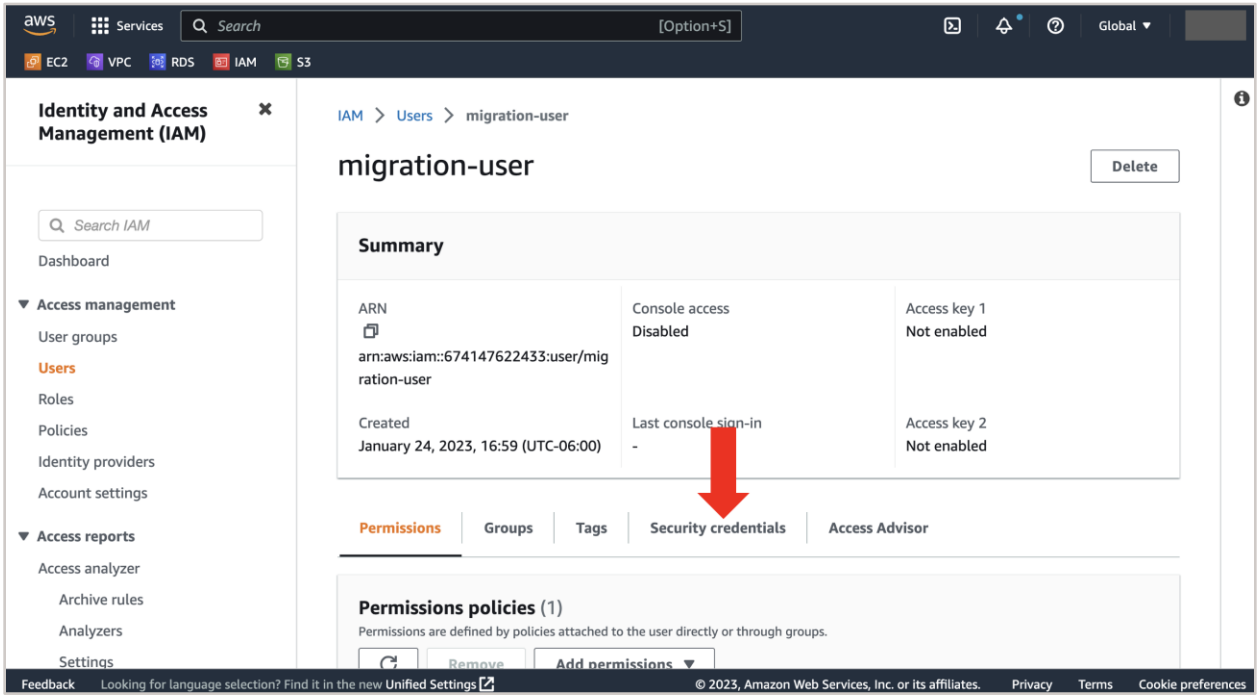
36. On Step 3, review all the information for accuracy. Click the “Create user” button afterwards.

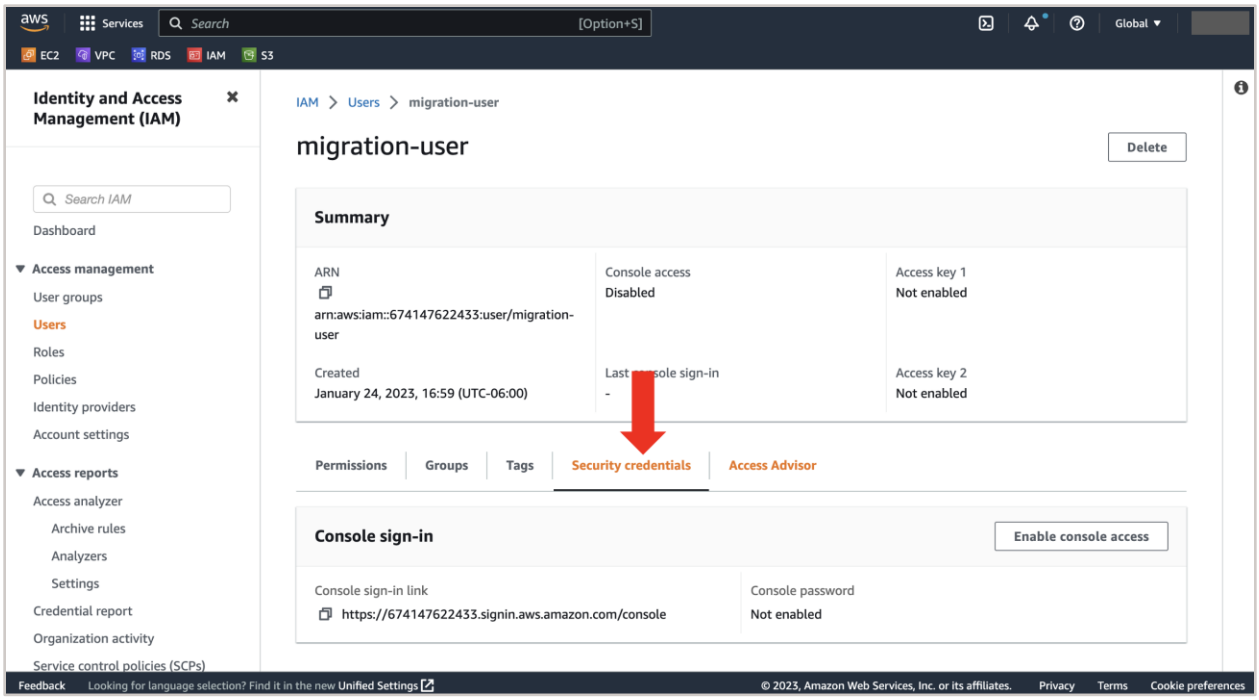


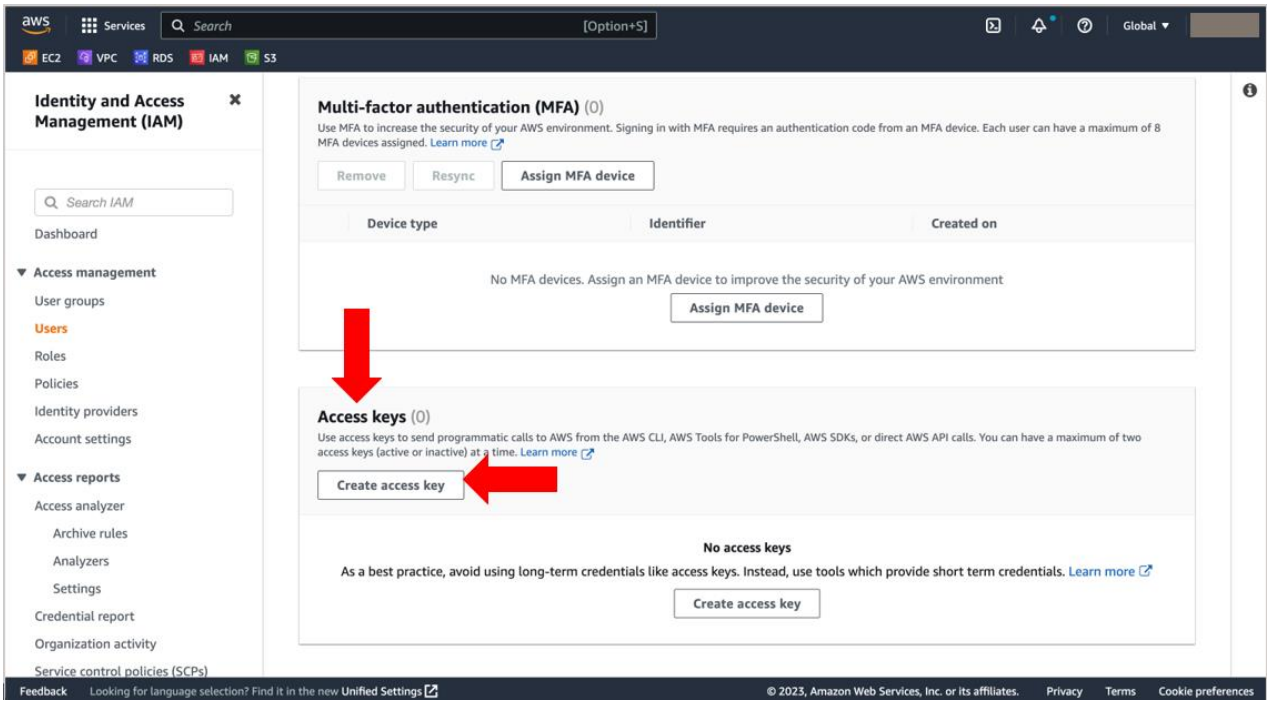
37. Once the User has been created, from the ‘Users’ page of IAM, click on the User we just created in the previous step



38. After your User page opens for the User that was just created, click on “Security credentials” and scroll down until you see “Access keys”

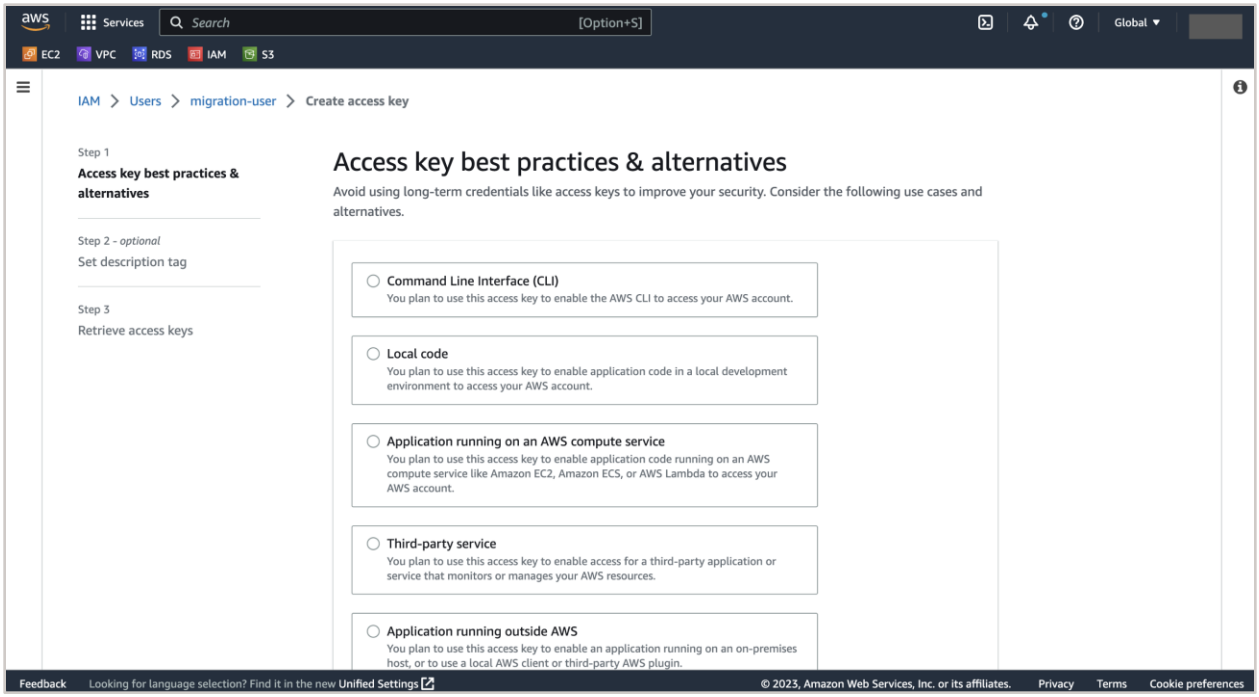


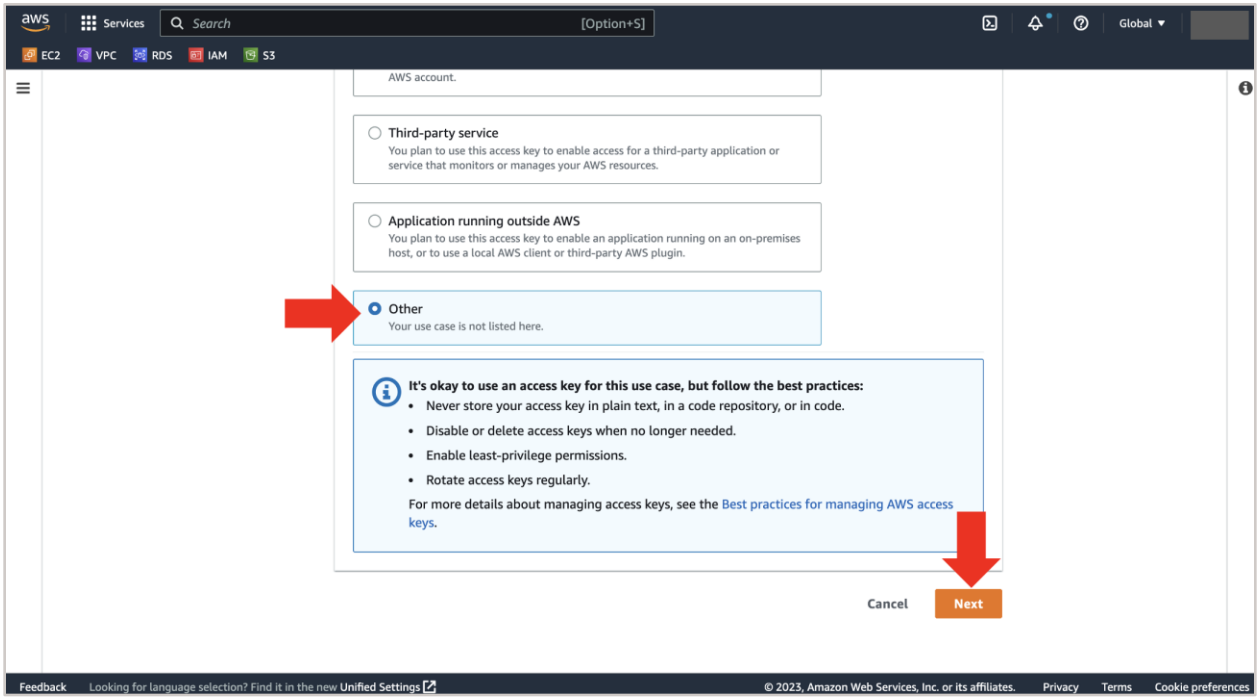




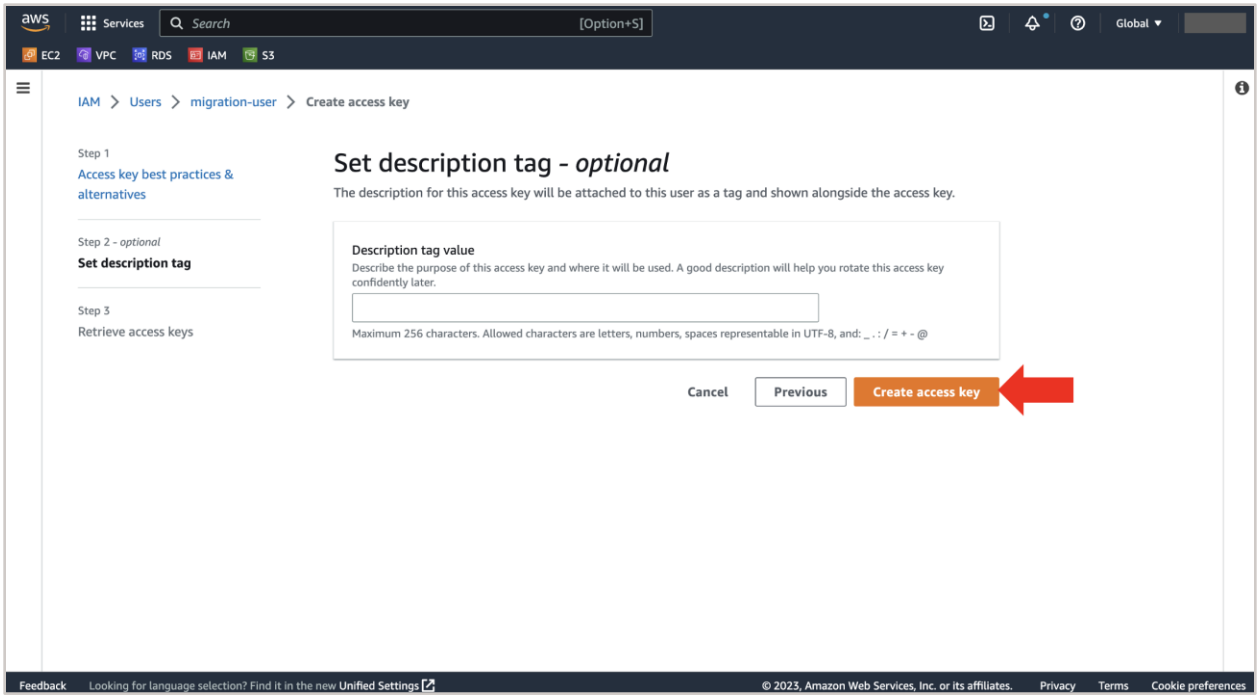
- Note: when you locate the 'Access keys' section, click on "Create access key"

39. On Step 1 of ‘Create access key’, locate and select “Other”. Click “Next” afterwards

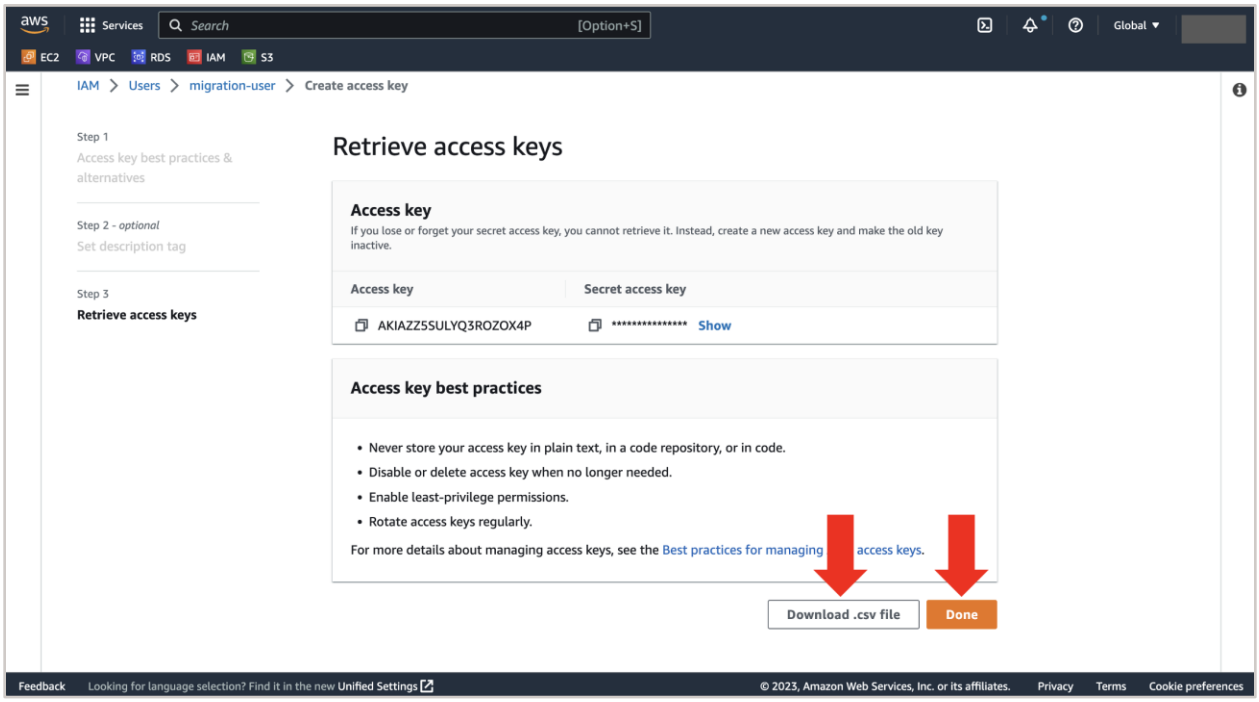




40. On Step 2, leave the values blank and click the “Create access key” button



41. On Step 3, your “Access key” will be created alongside the “Secret access key”. Save these two keys in a notepad for later use. Click “Download .csv file” to save the Access key and Secret access key in a .csv file. Click “Done” after downloading the .csv file



Section F: Create a credentials file in your EC2 instance

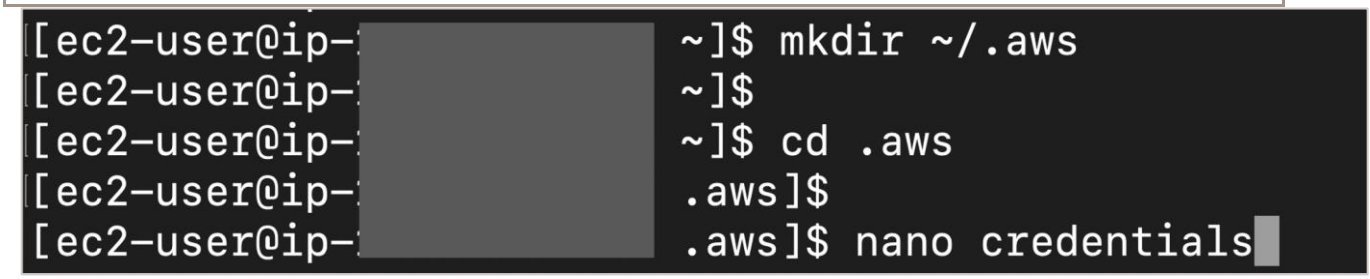
42. After creating the bucket and adding a user in AWS, go back to your EC2 instance where you have MySQL Shell installed.

    On the EC2 instance where MySQL Shell is installed, create a new directory called ".aws" inside your home directory. Next, go into the ".aws" directory and create a file called "credentials". After the file is created, copy and paste the below contents in that "credentials" file.

```
[default]
aws_access_key_id=
aws_secret_access_key=
region=
```

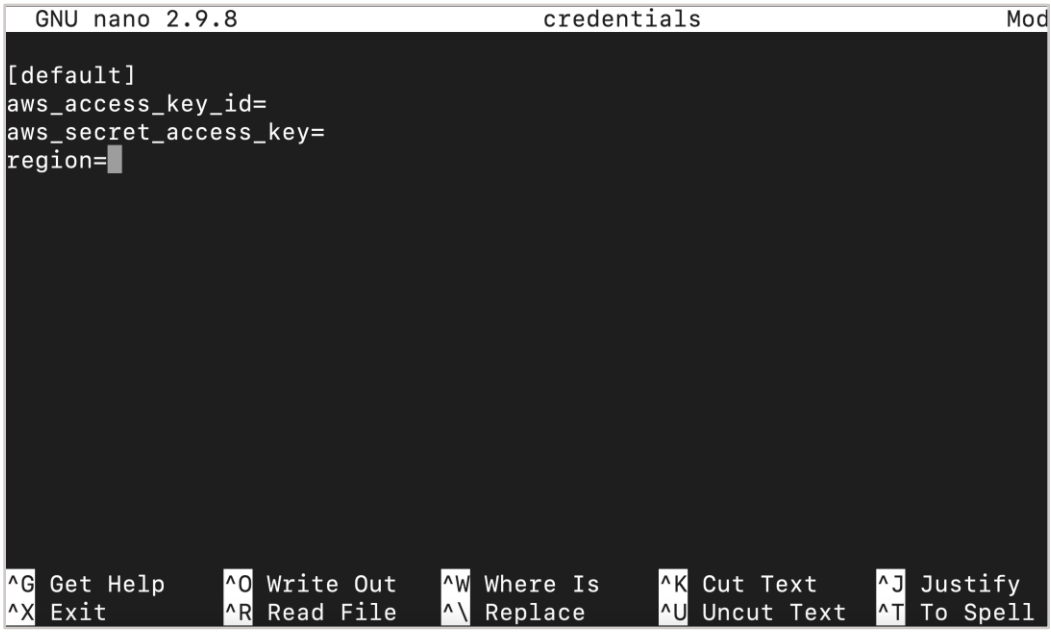
    The commands used to achieve this step for the guide are listed below:

```
ec2-user $ mkdir ~/.aws
ec2-user $ cd .aws
ec2-user $ nano credentials
```



- Note: to download nano, execute `sudo yum install nano -y`

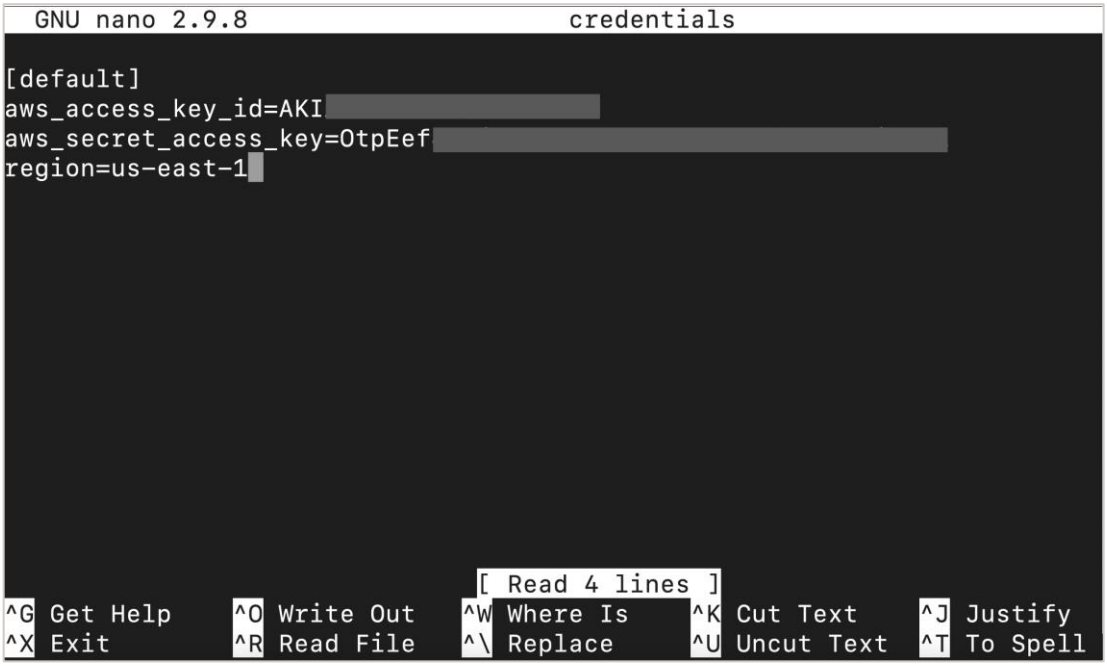
43. After pasting the "credentials" file contents from Step 42, below is how your "credentials" file should look like



44. Inside the ‘credentials’ file, for the “aws_access_key_id” and “aws_secret_access_key” fields, fill them using the .csv file we downloaded in Step 41. For “region”, since we have a Bucket created in us-east-1 and the MySQL HeatWave system that we will create in the later steps will also be in the same region, enter

```
us-east-1
```

After filling all the information for your credentials file, you should have something like this:



- Note: save the “credentials” file after filling all the missing fields. If you are using nano,
- to paste the copied content: `command + V`
- to save the file: `control + O`
- to exit the file: `control + X`

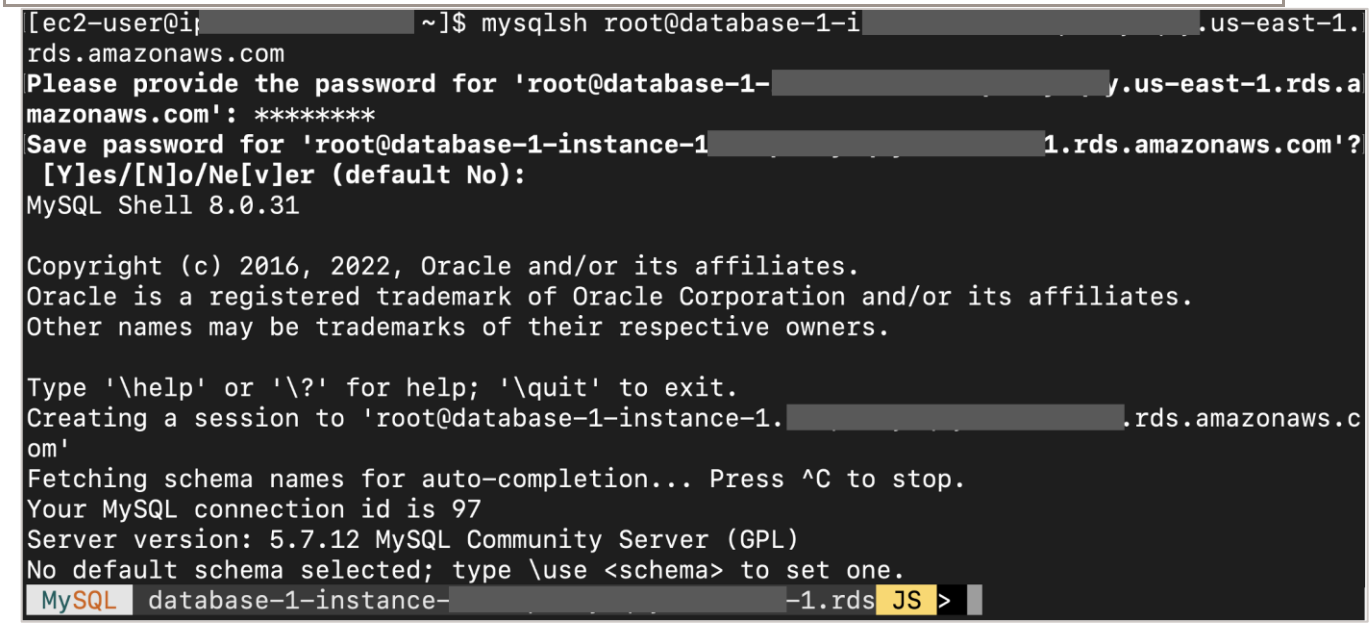
Section G: Connect to your Amazon Aurora MySQL Server using MySQL Shell and execute the util.dumpInstance() utility

45. Using MySQL Shell installed on your EC2 instance, connect to your Amazon Aurora MySQL Server by executing (account with Root privilege necessary):

```
ec2-user $ mysqlsh <username>@<localhost/ip>
```

or

```
ec2-user $ mysqlsh -u <username> -h <localhost/ip> -P <portnumber> -p
```



- Note: anytime you login using MySQL Shell, MySQL Shell will display the MySQL Shell version and MySQL Sever version currently being used. You can see this in the image above.

46. Once you are inside MySQL Shell, you can interact in three different modes. The default is JavaScript, the other ones you can choose from are SQL and Python. Once inside MySQL Shell:
    - to switch to JavaScript mode, execute: `\js`
    - to switch to SQL mode, execute: `\sql`
    - to switch to Python mode, execute: `\py`

47. Make sure you are in JavaScript mode by typing `\js` and execute the `dumpInstance` utility to export the dump data into AWS S3 Storage bucket.

```
MySQL JS> \js
```

```
MySQL JS> util.dumpInstance("sampledump",{s3bucketName: "heatwave-s3",
ocimds: "true", compatibility: ["strip_restricted_grants", "strip_definers",
"ignore_missing_pks"], users: "true", dryRun: "true"})
```

```
MySQL database-1.clust                                .rds.amazonaws JS > util.dumpInstance("sample
dump",{s3bucketName: "heatwave-s3", ocimds: "true", compatibility: ["strip_restricted_grants",
 "strip_definers", "ignore_missing_pks"], users: "true", dryRun: "true"})
dryRun enabled, no locks will be acquired and no files will be created.
NOTE: Backup lock is not supported in MySQL 5.7 and DDL changes will not be blocked. The dump
may fail with an error if schema changes are made while dumping.
Acquiring global read lock
WARNING: The current user lacks privileges to acquire a global read lock using 'FLUSH TABLES W
ITH READ LOCK'. Falling back to LOCK TABLES...
ERROR: The current user does not have required privileges to execute FLUSH TABLES WITH READ LO
CK.
    Backup lock is not supported in MySQL 5.7 and DDL changes cannot be blocked.
    The gtid_mode system variable is set to OFF or OFF_PERMISSIVE.
    The log_bin system variable is set to OFF or the current user does not have required privi
leges to execute SHOW MASTER STATUS.
The consistency of the dump cannot be guaranteed.
ERROR: Unable to acquire global read lock neither table read locks.
Global read lock has been released
Util.dumpInstance: While 'Initializing': Unable to lock tables: Consistency check has failed.
(MYSQLSH 52002)
MySQL database-1.cluster-                          s.amazonaws JS >
```

Note:

- The `util.dumpInstance()` utility will take a dump of all the databases except “mysql, sys, performance schema, and information schema”. The dump comprises of DDL files for the schema structure and tab-separated .tsv files containing the actual data. Additionally, you can also use `util.dumpSchemas()` or `util.dumpTables()` if you only want to dump specific schemas or tables. The three dump utilities can export the data into:
- a) Object Storage bucket in Oracle Cloud
- b) S3-compatible buckets
- c) local filesystem
- This guide showcases option b). For more information, refer: https://dev.mysql.com/doc/mysql-shell/8.0/en/mysql-shell-utilities-dump-instance-schema.html#mysql-shell-utilities-dump-opt-run
- The `dryRun` option runs the export command but does not generate any output export file. It displays information about what would be dumped with the specified set of options, and about the results of MySQL HeatWave compatibility checks (if the `ocimds` option is specified, which is required for this guide), but does not proceed with the dump. Setting this option enables you to list out all the compatibility issues before starting the dump. The default is false. You can read more about the utility options at https://dev.mysql.com/doc/mysql-shell/8.0/en/mysql-shell-utilities-dump-instance-schema.html#mysql-shell-utilities-dump-opt-control
- In the command above, `sampledump` is the prefix under which all the exported dump files will be stored in S3 Storage bucket in AWS.
- Change the `s3bucketName` to match with what you have when you created your bucket in AWS in Step 30.
- Setting the `ocimds: true` option ensures compatibility of the export dump with MySQL HeatWave.

- Primary keys are required on every table for using MySQL HeatWave.
- If you can't seem to solve an error during the `dryRun`, contact a MySQL Solution Engineer for guidance: https://go.oracle.com/LP=132857?src1=:ow:o:s:po:::&intcmp=:ow:o:s:po:::
- To understand the `dumpInstance()`, `dumpSchemas()`, or `dumpTables()` utility in more detail, refer to this website: https://dev.mysql.com/doc/mysql-shell/8.0/en/mysql-shell-utilities-dump-instance-schema.html

48. Running the commands in Step 47 may generate "Errors" regarding "table locks" (see the image in Step 47). If (and only if) you do encounter such an error, execute the same command but add an additional option "consistent: false"

```
MySQL JS> util.dumpInstance("sampledump",{s3bucketName: "heatwave-s3",
ocimds: "true", compatibility: ["strip_restricted_grants", "strip_definers",
"ignore_missing_pks"], users: "true", dryRun: "true", consistent: "false"})
```

```
 MySQL  database-1.c                                  s-east-1.rds  JS > util.dumpInstance("sampl
edump",{s3bucketName: "heatwave-s3", ocimds: "true", compatibility: ["strip_restric
ted_grants", "strip_definers", "ignore_missing_pks"], users: "true", dryRun: "true"
, consistent: "false"})
dryRun enabled, no locks will be acquired and no files will be created.
Initializing - done
1 out of 5 schemas will be dumped and within them 3 tables, 0 views.
2 out of 3 users will be dumped.
Gathering information - done
WARNING: The dumped value of gtid_executed is not guaranteed to be consistent
Checking for compatibility with MySQL Database Service 8.0.31
NOTE: MySQL Server 5.7 detected, please consider upgrading to 8.0 first.
Checking for potential upgrade issues.
The MySQL server at
database-1.cluster-c                                 .rds.amazonaws.com:3306, version
5.7.12 - MySQL Community Server (GPL), will now be checked for compatibility
issues for upgrade to MySQL 8.0.31...

1) MySQL 8.0 syntax check for routine-like objects
```

```
16) Check for invalid table names and schema names used in 5.7
  No issues found

Errors:   0
Warnings: 1
Notices:  0

NOTE: No fatal errors were found that would prevent an upgrade, but some potential
issues were detected. Please ensure that the reported issues are not significant be
fore upgrading.
NOTE: User 'rdsadmin'@'localhost' had restricted privileges (CREATE TABLESPACE, FIL
E, RELOAD, SHUTDOWN, SUPER) removed
NOTE: User 'root'@'%' had restricted privileges (INVOKE COMPREHEND, INVOKE LAMBDA,
INVOKE SAGEMAKER, LOAD FROM S3, RELOAD, SELECT INTO S3) removed
Compatibility issues with MySQL Database Service 8.0.31 were found and repaired. Pl
ease review the changes made before loading them.
Validating MDS compatibility - done
Writing global DDL files
Writing users DDL
Writing DDL - done
Starting data dump
0% (0 rows / ~5.30K rows), 0.00 rows/s, 0.00 B/s uncompressed, 0.00 B/s compressed
 MySQL  database-                                    .us-east-1.rds  JS >
```

49. Once you have executed the command in Step 47/48 and did not see any errors or warnings, execute the same Step 47/48 command. Although, this time change the dryRun option to false

```
MySQL JS> util.dumpInstance("sampledump",{s3bucketName: "heatwave-s3",
ocimds: "true", compatibility: ["strip_restricted_grants", "strip_definers",
"ignore_missing_pks"], users: "true", dryRun: "false", consistent: "false"})
```

```
 MySQL  database-1.c                                  s-east-1.rds JS > util.dumpInstance("sampl
edump",{s3bucketName: "heatwave-s3", ocimds: "true", compatibility: ["strip_restric
ted_grants", "strip_definers", "ignore_missing_pks"], users: "true", dryRun: "false
", consistent: "false"})
Initializing - done
1 out of 5 schemas will be dumped and within them 3 tables, 0 views.
2 out of 3 users will be dumped.
Gathering information - done
WARNING: The dumped value of gtid_executed is not guaranteed to be consistent
Checking for compatibility with MySQL Database Service 8.0.31
NOTE: MySQL Server 5.7 detected, please consider upgrading to 8.0 first.
Checking for potential upgrade issues.
The MySQL server at
database-1.cl                                  -east-1.rds.amazonaws.com:3306, version
5.7.12 - MySQL Community Server (GPL), will now be checked for compatibility
issues for upgrade to MySQL 8.0.31...

1) MySQL 8.0 syntax check for routine-like objects
```

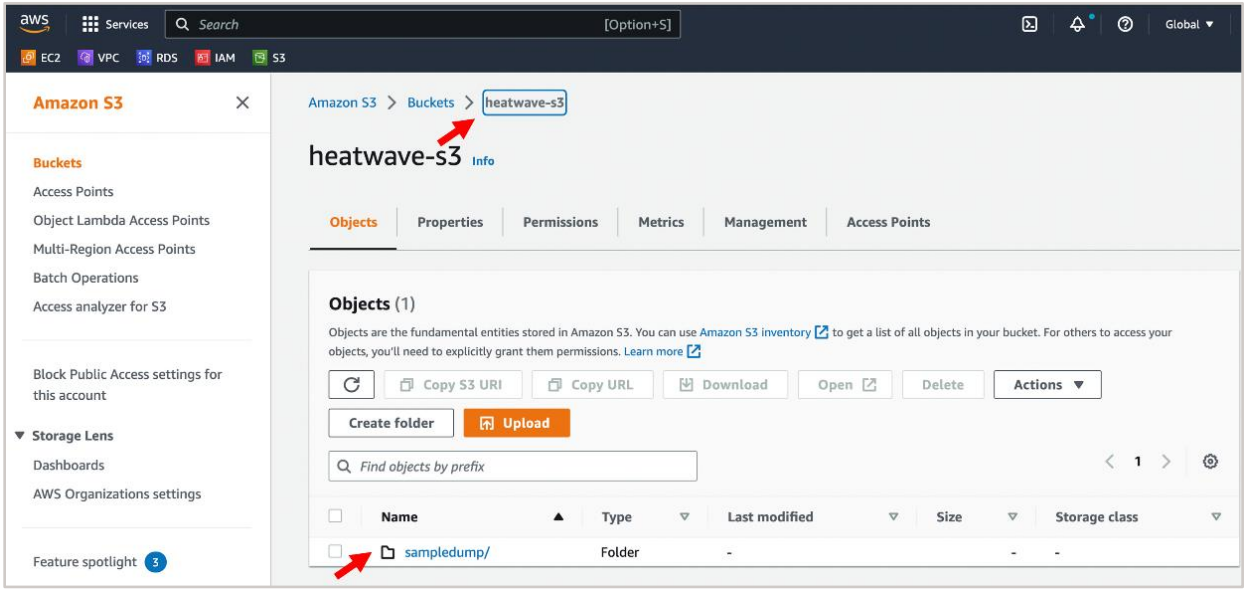
```
Validating MDS compatibility - done
Writing global DDL files
Writing users DDL
Running data dump using 4 threads.
NOTE: Progress information uses estimated values and may not be accurate.
Writing schema metadata - done
Writing DDL - done
Writing table metadata - done
Starting data dump
3 thds dumping - 75% (4.00K rows / ~5.30K rows), 0.00 rows/s, 0.00 B/s uncompressed

100% (5.30K rows / ~5.30K rows), 0.00 rows/s, 0.00 B/s uncompressed, 0.00 B/s compr
essed

Dump duration: 00:00:00s

Total duration: 00:00:01s

Schemas dumped: 1

Tables dumped: 3

Uncompressed data size: 194.62 KB

Compressed data size: 91.71 KB

Compression ratio: 2.1

Rows written: 5302

Bytes written: 91.71 KB

Average uncompressed throughput: 194.62 KB/s

Average compressed throughput: 91.71 KB/s
 MySQL  database-1.                                  t-1.rds JS >
```

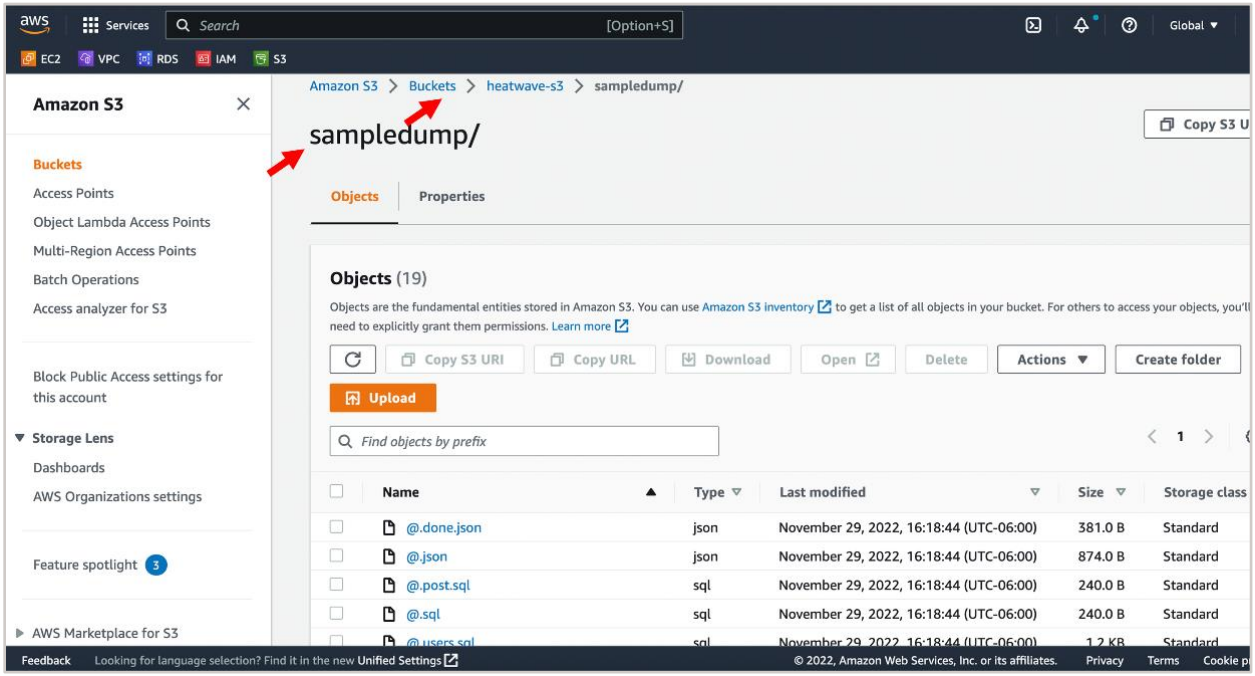
- Note: once the dump process is complete, MySQL Shell will display a summary of the dump process like the one shown above.

III. Importing the database

Section H: Navigate to the S3 Storage bucket to confirm if the dump was successful

50. Once the export dump operation has completed, go back to your AWS S3 Storage bucket created in Step 30 and locate the dump files under the `sampledump` prefix

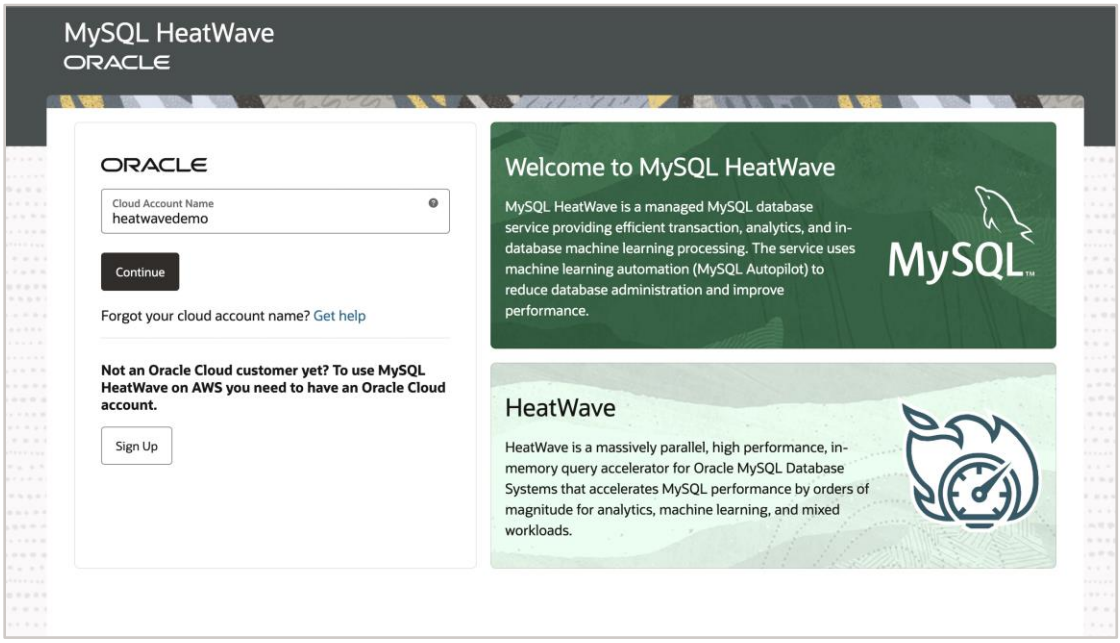




Section I: Create a MySQL HeatWave System

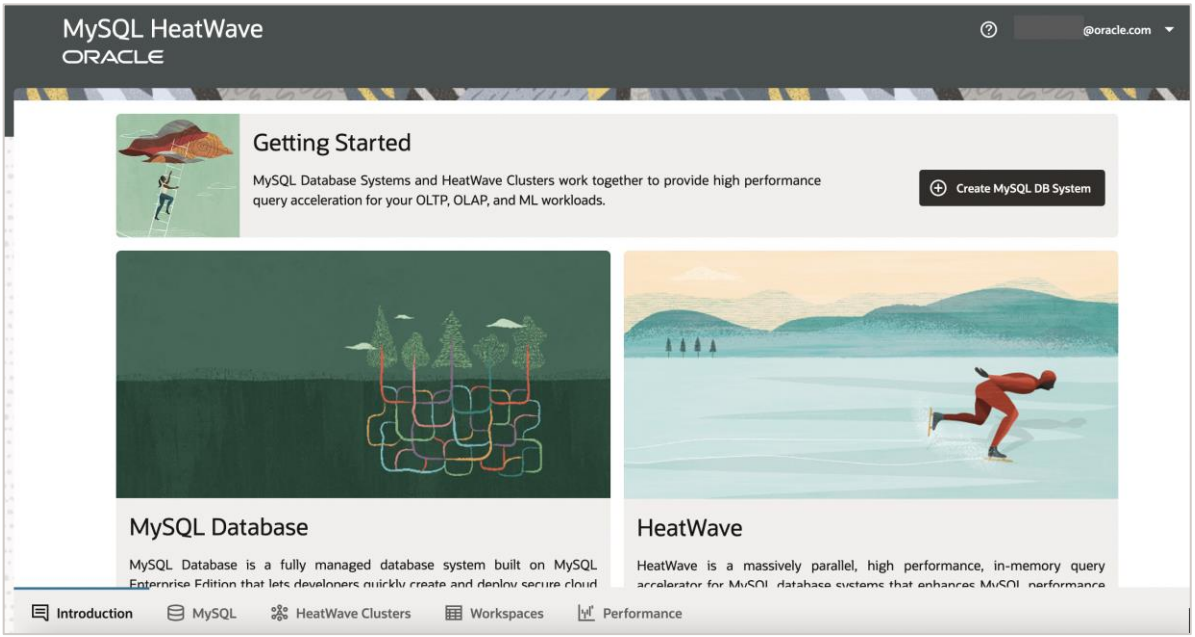
51. After completing all the above Steps, navigate to “cloud.mysql.com” to provision your MySQL HeatWave on AWS instance (assuming you have enabled MySQL HeatWave on AWS from OCI)

```
cloud.mysql.com
```

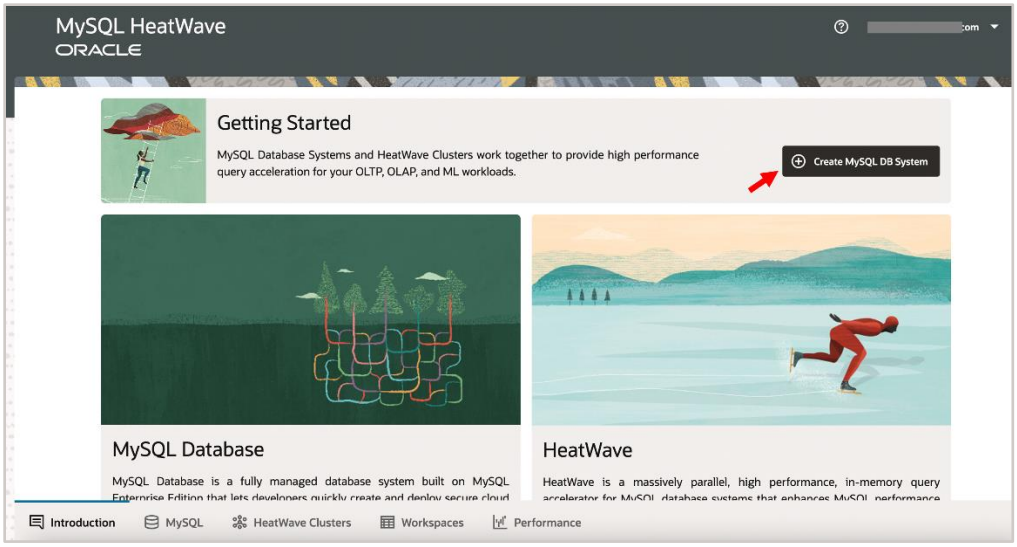


- Note: on the above page, enter your OCI Account Name and click “Continue”. Afterwards, you will be prompted to enter your Oracle Cloud ‘User Name’ and ‘Password’.

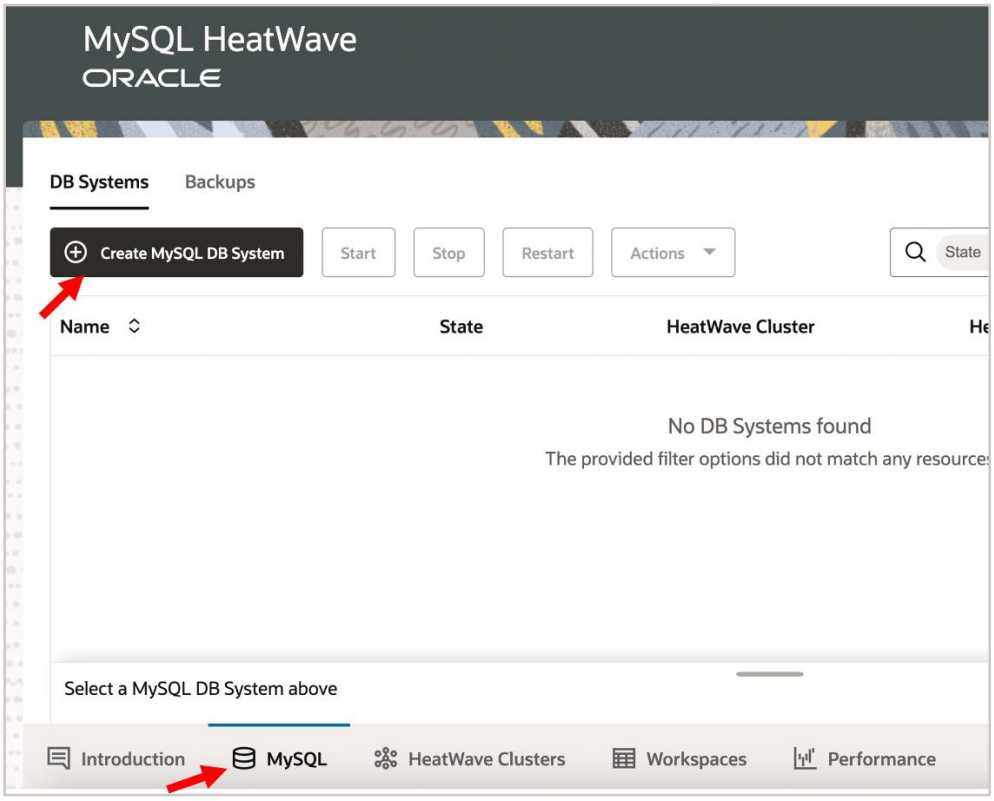
52. Once you are logged in, this is what the home screen looks like: the MySQL HeatWave on AWS Console



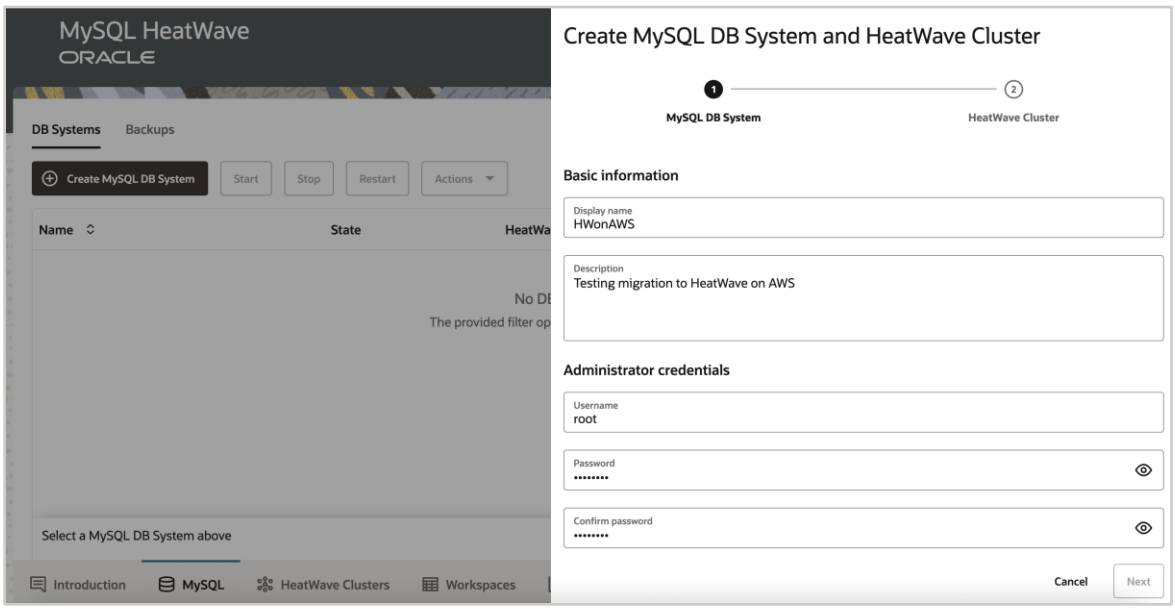
53. Click the “Create MySQL DB System” button



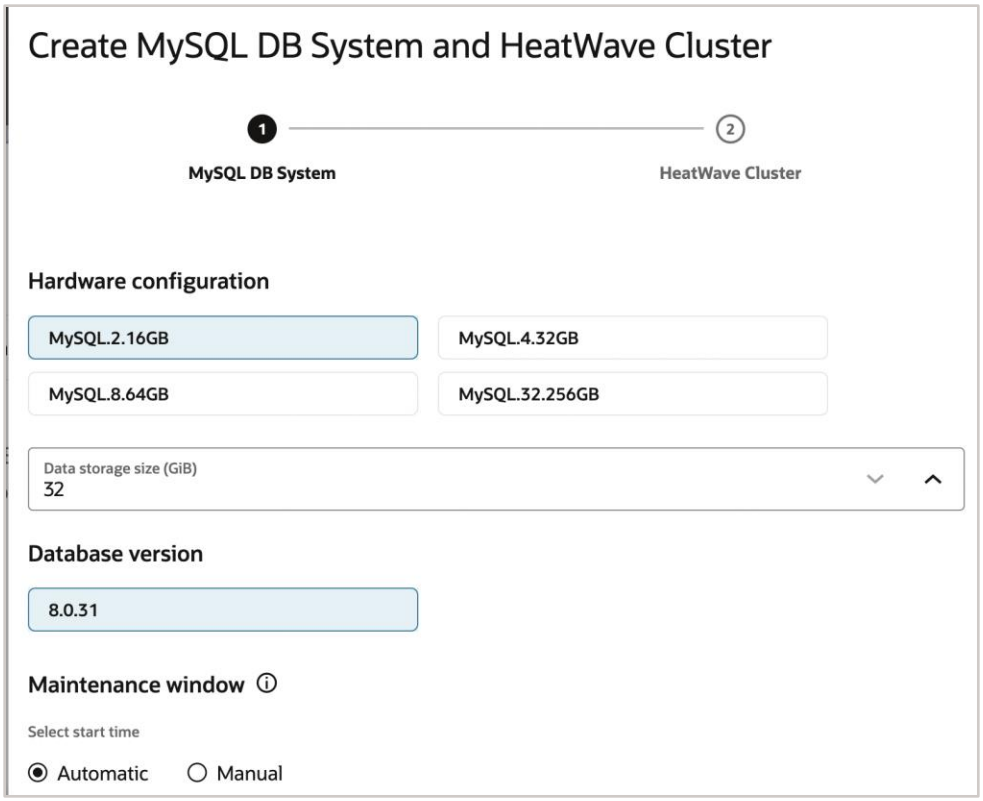
- Note: You can also perform the same action by clicking the ‘MySQL’ tab at the bottom of the page and then clicking the ‘Create MySQL DB System’ button



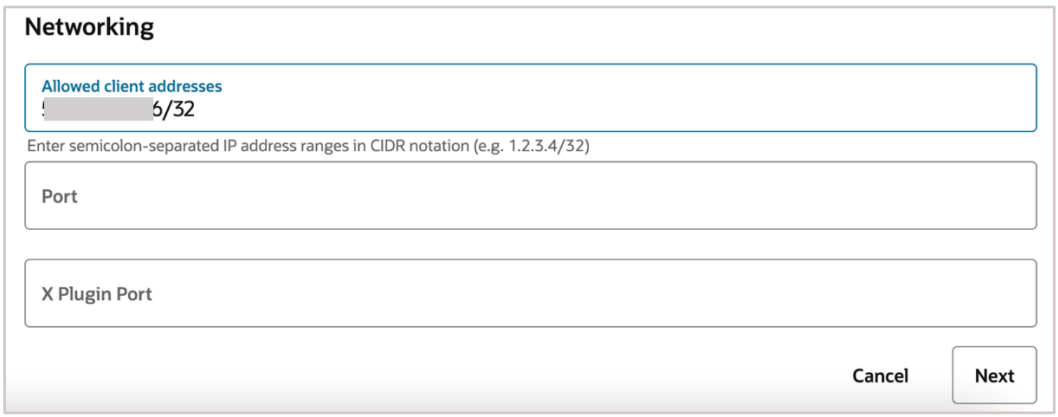
54. After clicking on “Create MySQL DB System”, enter a name for your MySQL DB system. Then, create an Admin ‘Username’ and ‘Password’.



55. Scroll down and choose the appropriate ‘Hardware configuration’ and ‘Data storage size’. The minimum storage size you can select is 32 GB. The maximum storage size is 65 TB. For your InnoDB storage, if it is greater than 1 TB, we recommend you switch to the 32.256GB shape. Leave the ‘Maintenance window’ and ‘Availability zone’ as-is.

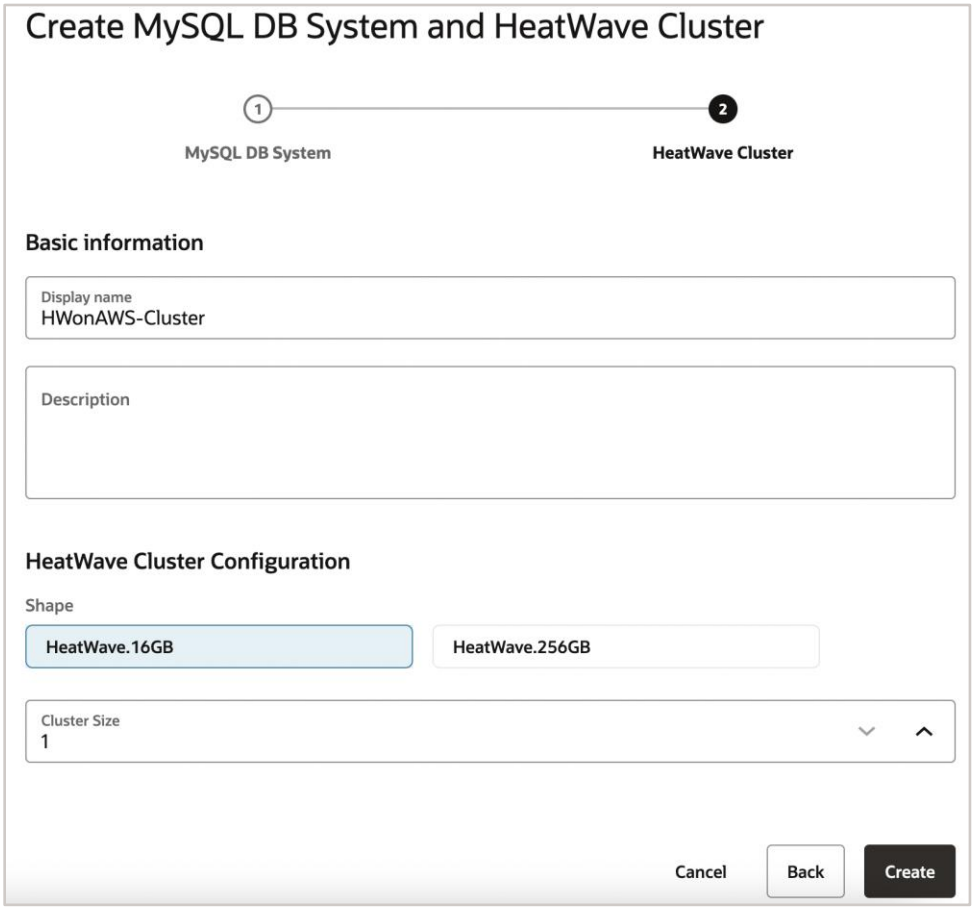


56. Under ‘Networking’ and ‘Allowed client addresses’, enter the Public IP address of your EC2 Compute Instance that we created in the earlier step, followed by a ‘/32’.

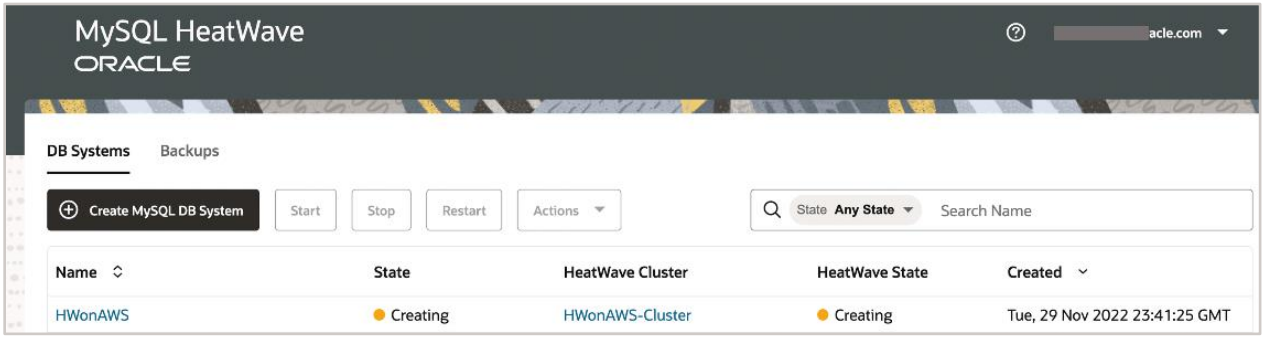


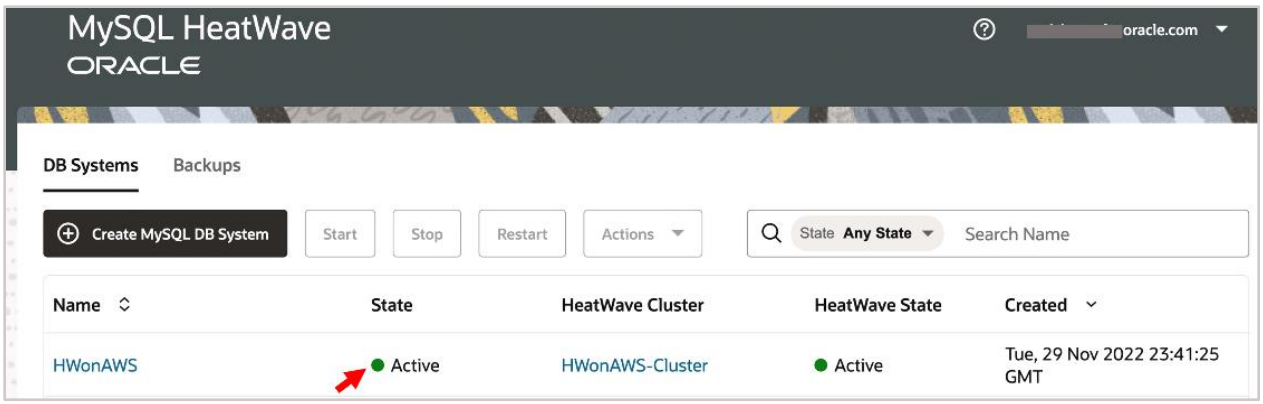
- Note: click “Next” after you have entered at least one client address under the ‘Allowed client addresses’

57. After clicking Next, you will be taken to Page 2 where you will create a HeatWave Cluster. Name your HeatWave Cluster whatever you want and chose the appropriate “HeatWave Cluster Configuration”. For the “Shape”, you can either choose a Cluster Node of 16 GB (can handle ~25 GB of data) or a Cluster Node of 256 GB (can handle ~400 GB of data). The “Cluster Size” can go from 1 to 128. Here we will use the “HWonAWS-Cluster” name, 16GB Shape and Cluster Size of 1

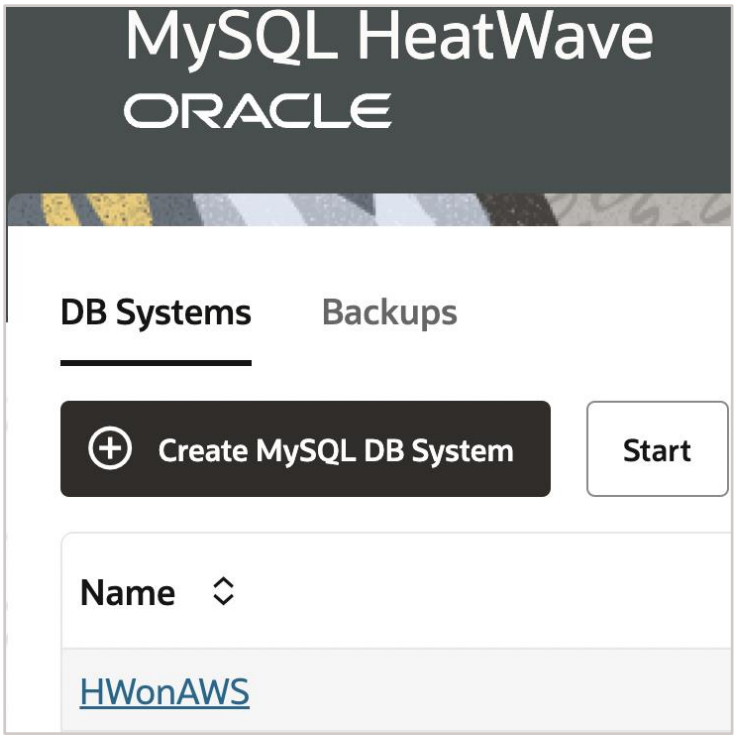


58. Click “Create” as shown in the above image, once you are done with everything. After a few minutes, your MySQL HeatWave System will be created and will be in an “Active” State

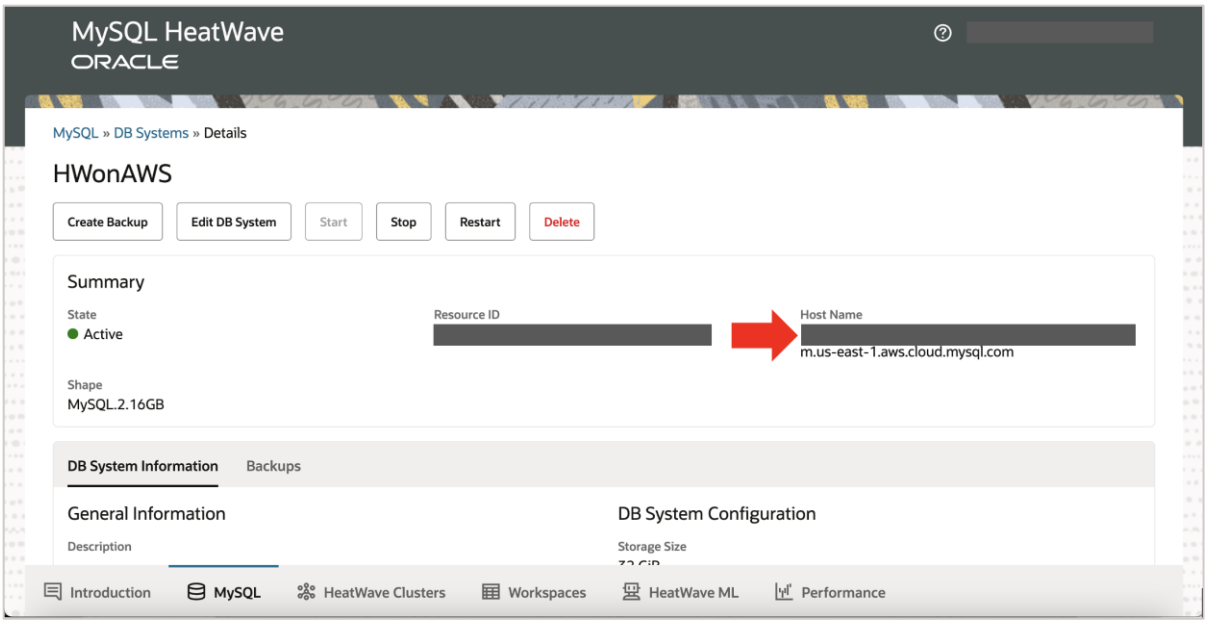




59. Once the System is created, click on the “Name” of your system. This will take us to the “DB Systems Details” page where we can view a variety of information regarding your MySQL HeatWave System.



60. You will then be taken to the “DB Systems Details” page



- Note: here, copy the “Host Name” for later use

Section J: Import the dumped data using the util.loadDump() utility

61. After noting down the Host Name, log back into your EC2 instance where we have the 'credentials' file and MySQL Shell installed. Using MySQL Shell, log in to your MySQL HeatWave instance (the EC2 instance whose IP you entered in 'Allowed Client Addresses' in Step 56)

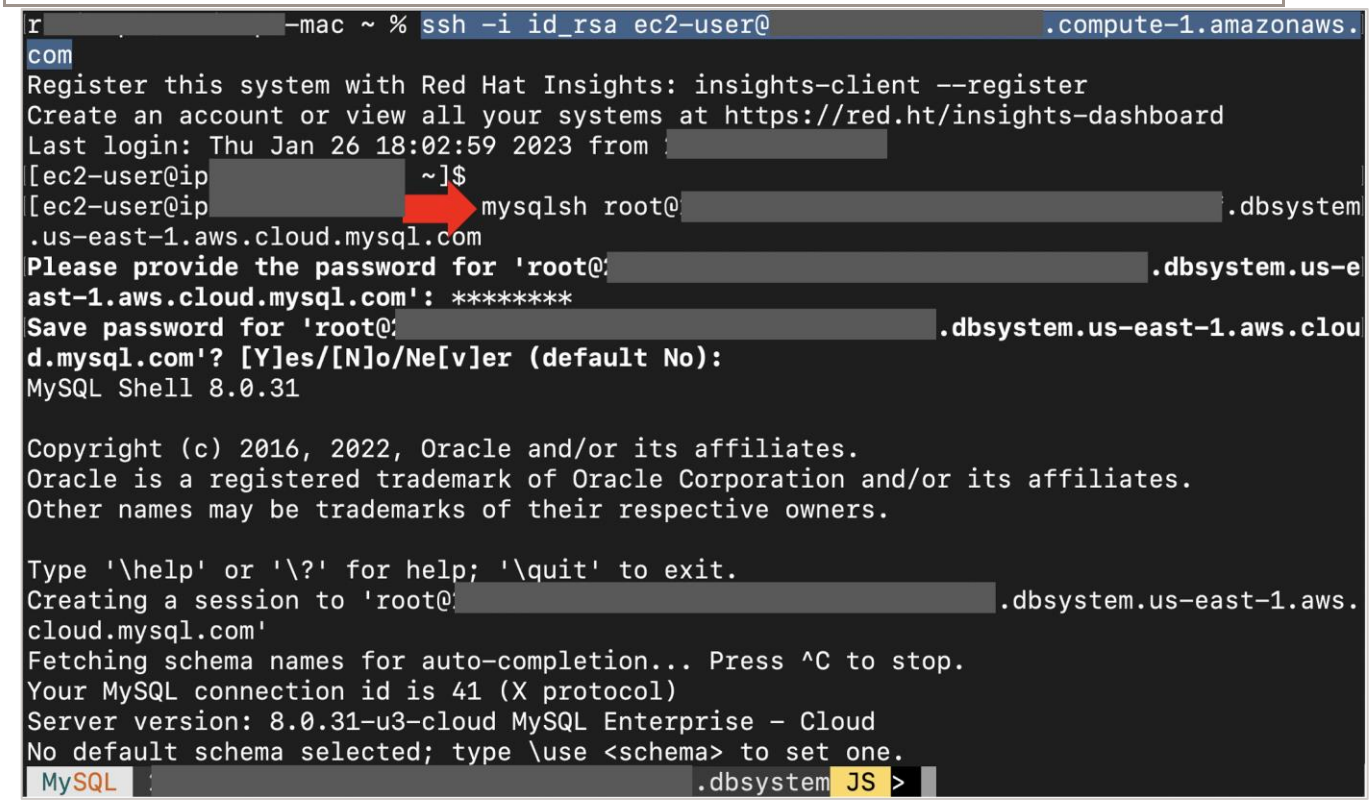
```
ssh -i <path/to/you-private-ssh-key> ec2-user@<ec2-Public-DNS>
```

then

```
$ mysqlsh <username>@<hostname>
```

or

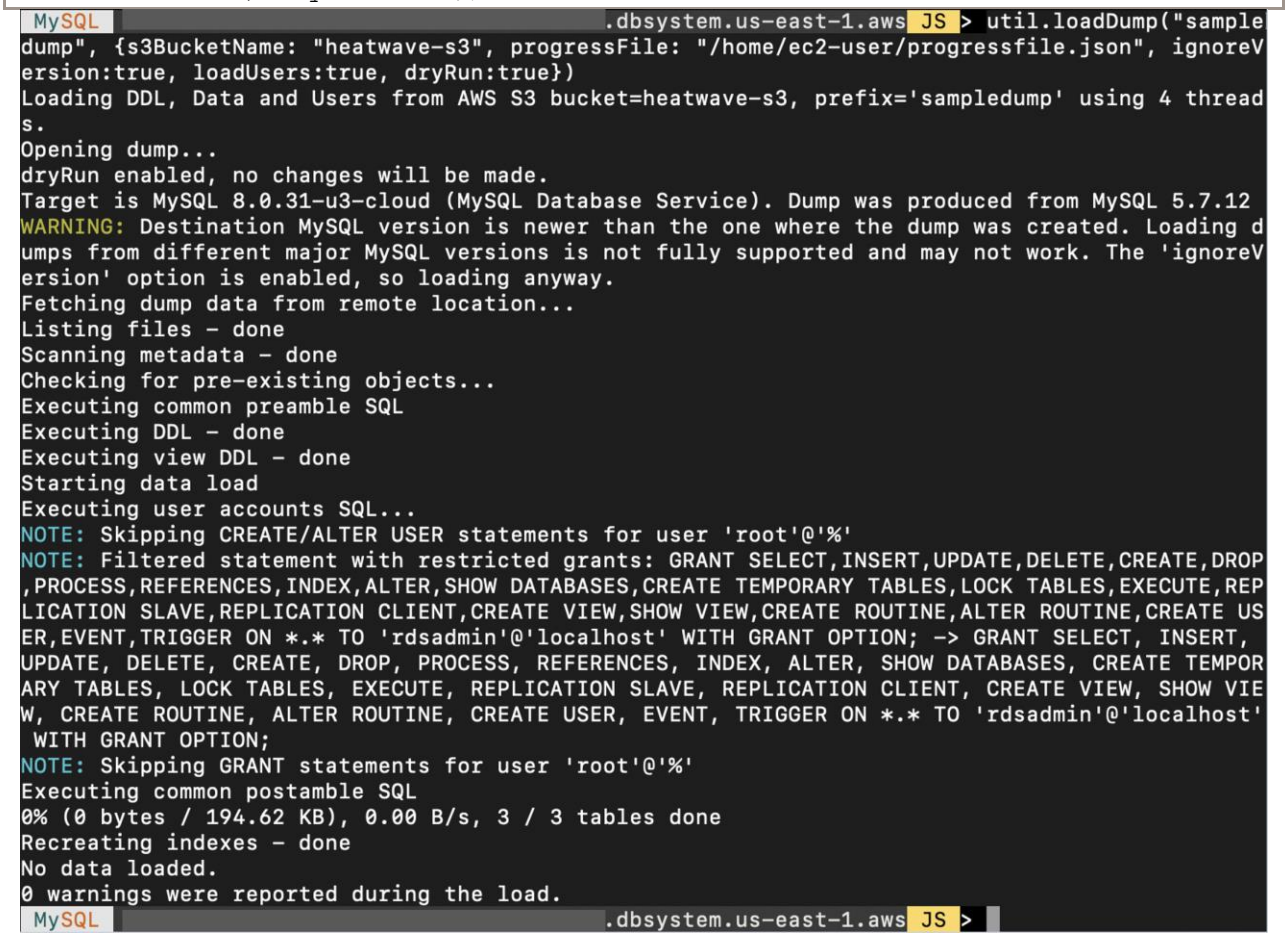
```
$ mysqlsh -u <username> -h <hostname> -P <portnumber> -p
```



62. Now that you are logged in to the MySQL HeatWave on AWS System, it is time to load our Amazon Aurora MySQL Server data from S3 into this newly created MySQL HeatWave System. Make sure you are in the JavaScript mode of MySQL Shell by executing `\js` and then execute the MySQL Shell Load command

```
MySQL JS> \js
```

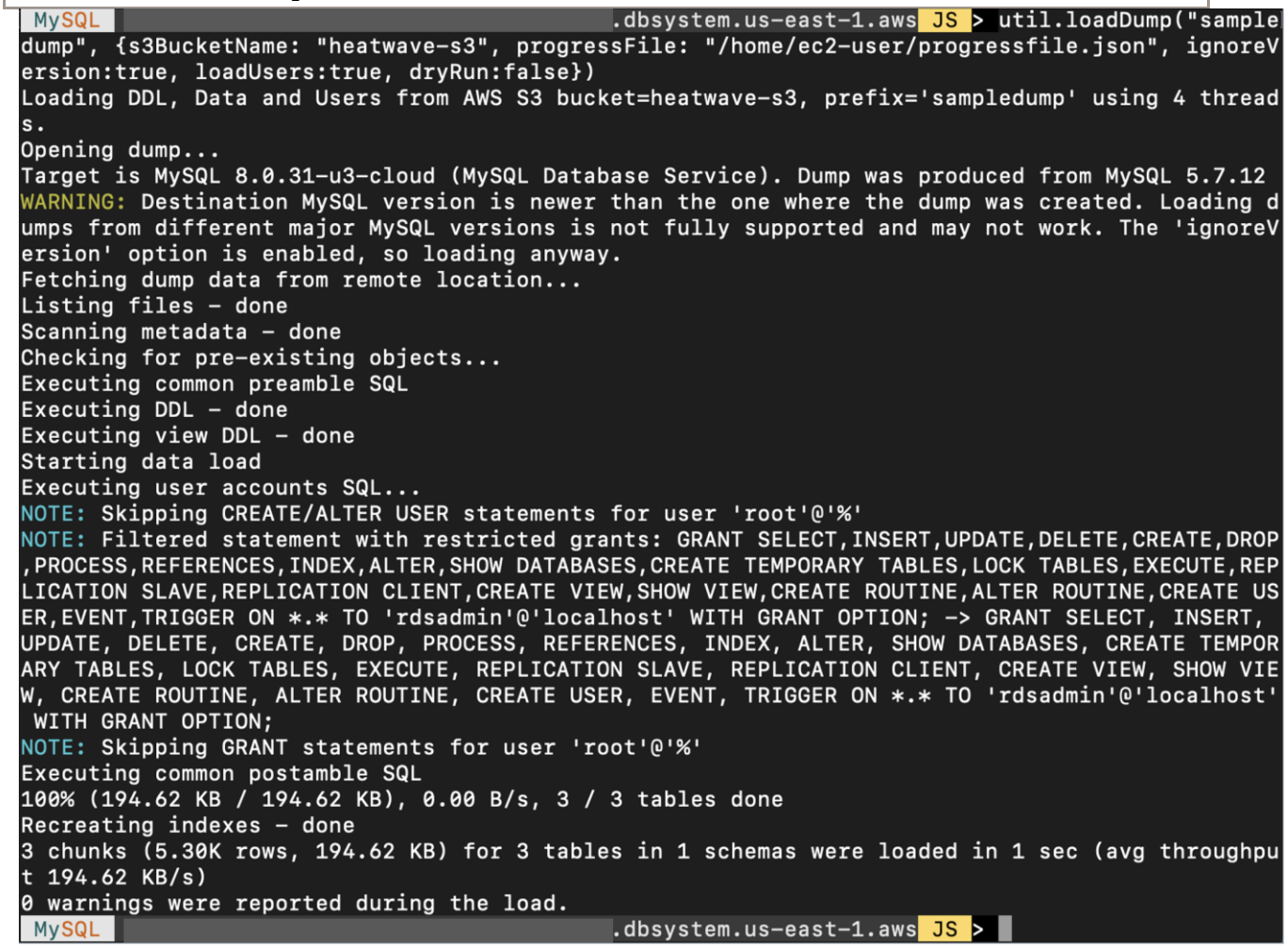
```
MySQL JS> util.loadDump("sampledump", {s3BucketName: "heatwave-s3",
progressFile: "/home/ec2-user/progressfile.json", ignoreVersion:true,
loadUsers:true, dryRun:true})
```



- Note:
  - The `util.loadDump()` utility will use the DDL files and tab-separated .tsv data files to set up the server instance or schema in the target MySQL instance, then load the data. For more information, refer to: https://dev.mysql.com/doc/mysql-shell/8.0/en/mysql-shell-utilities-load-dump.html
  - Change the prefix and `s3BucketName` to match with what you have.

63. Once you have executed the command in Step 62 and did not see any errors or warnings, execute the same Step 62 command. Although, this time change the dryRun option to false

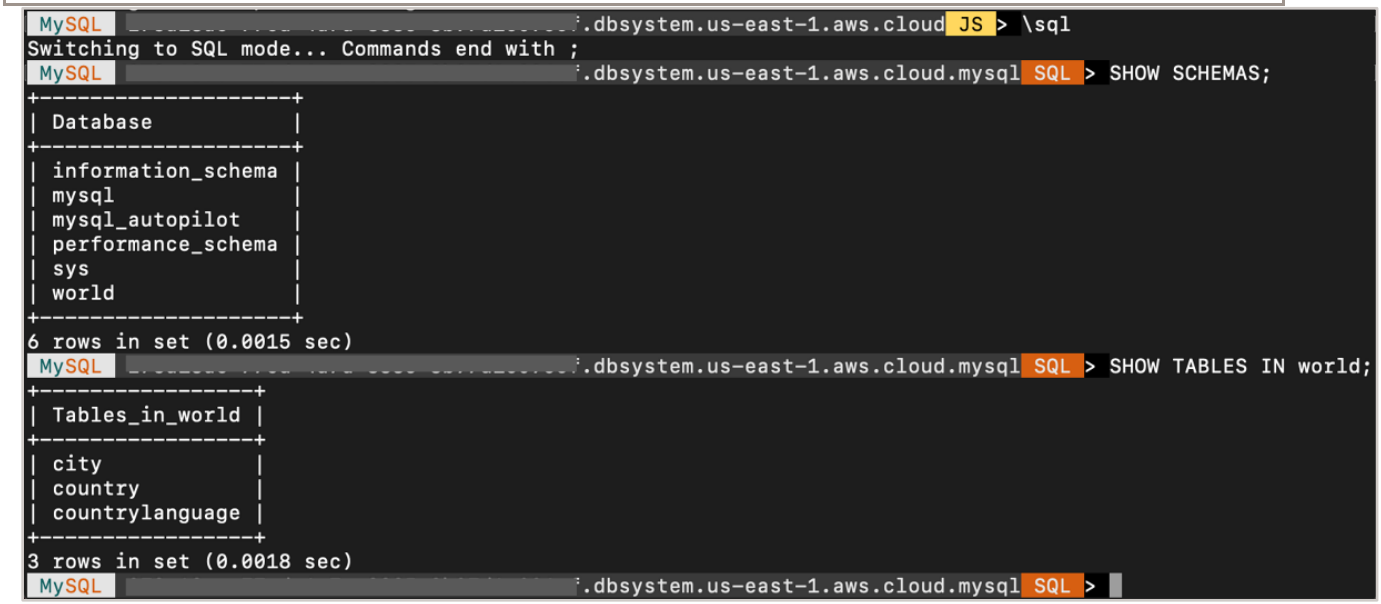
```
MySQL JS> util.loadDump("sampledump", {s3BucketName: "heatwave-s3",
progressFile: "/home/ec2-user/progressfile.json", ignoreVersion:true,
loadUsers:true, dryRun:false})
```



- Note: once the load process is complete, MySQL Shell will display a summary of the dump process like the one shown in the image above.

64. After your import command has completed successfully in the previous step, you can verify the schemas and tables imported by running the following commands in `\sql` mode of MySQL Shell:

```
MySQL JS> \sql
MySQL SQL> SHOW SCHEMAS;
MySQL SQL> SHOW TABLES IN world;
```

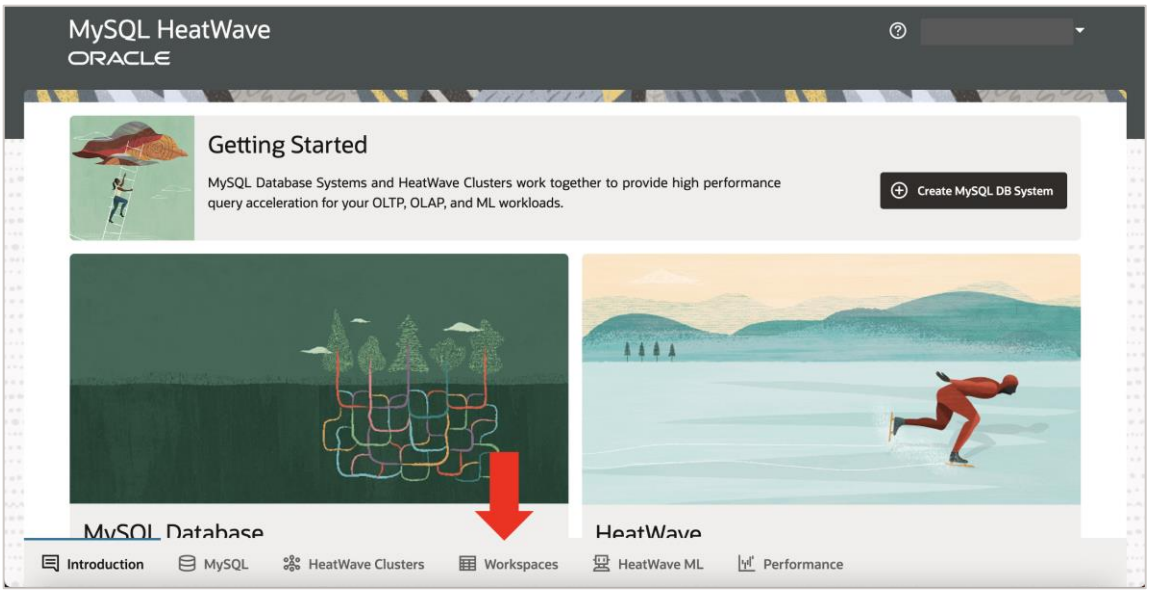


IV. Loading data into MySQL HeatWave

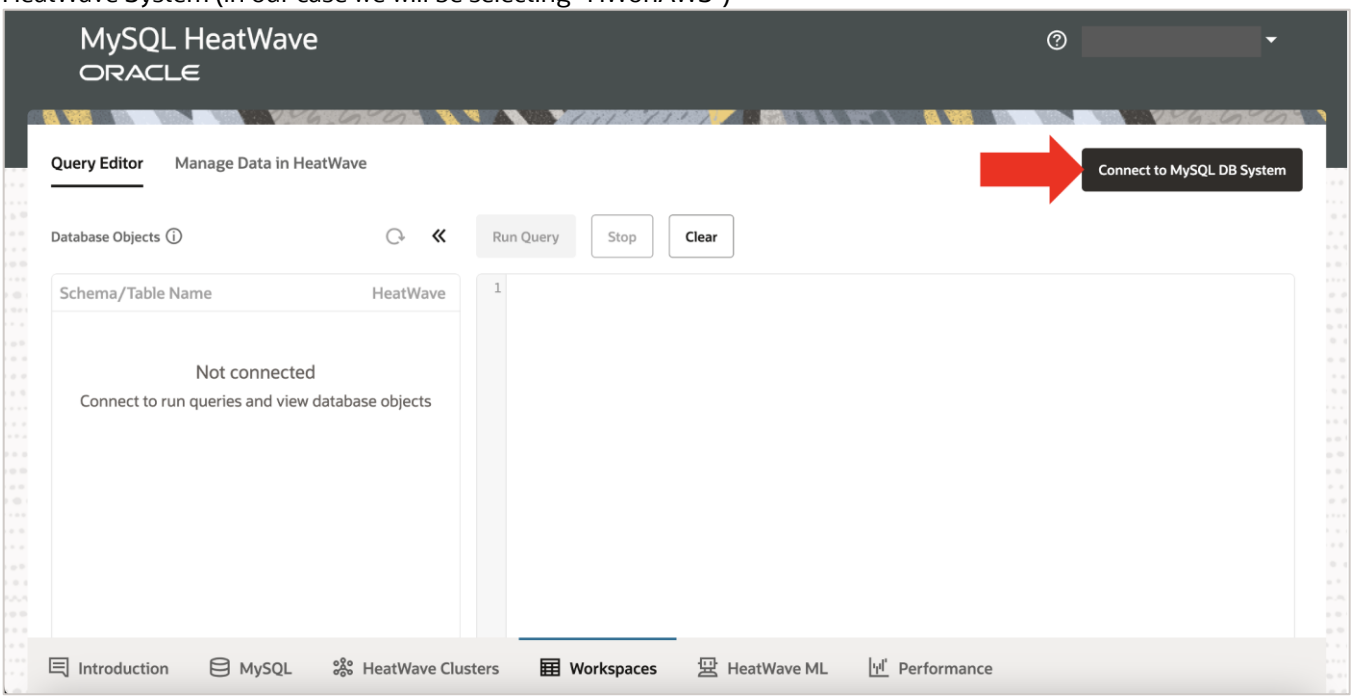
Section K: Load data into the HeatWave Cluster

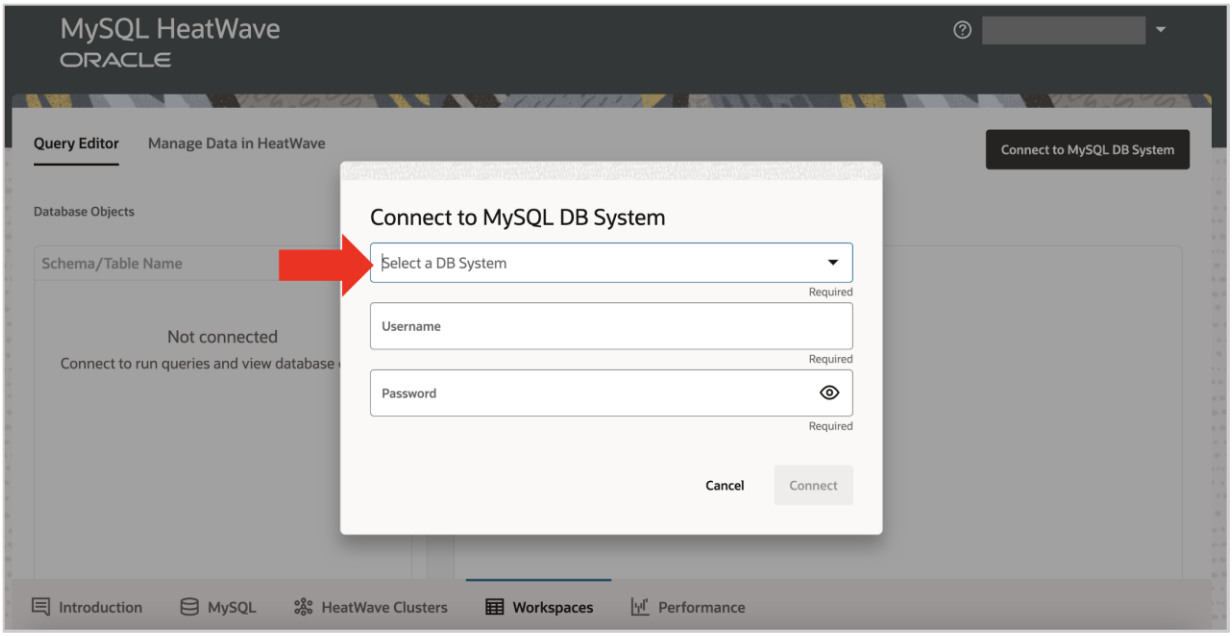
To make use of MySQL HeatWave's in-memory query engine and query acceleration capabilities, you need to attach a HeatWave cluster to your MySQL database.

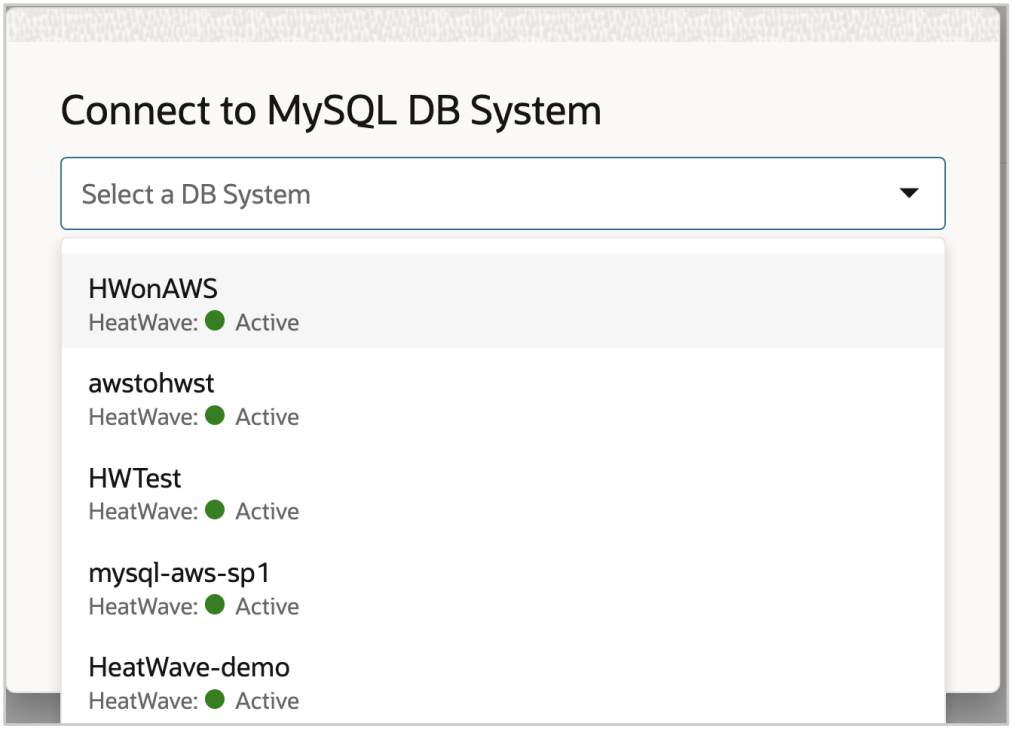
65. Login to your MySQL HeatWave on AWS Console and navigate to the 'Workspaces' tab



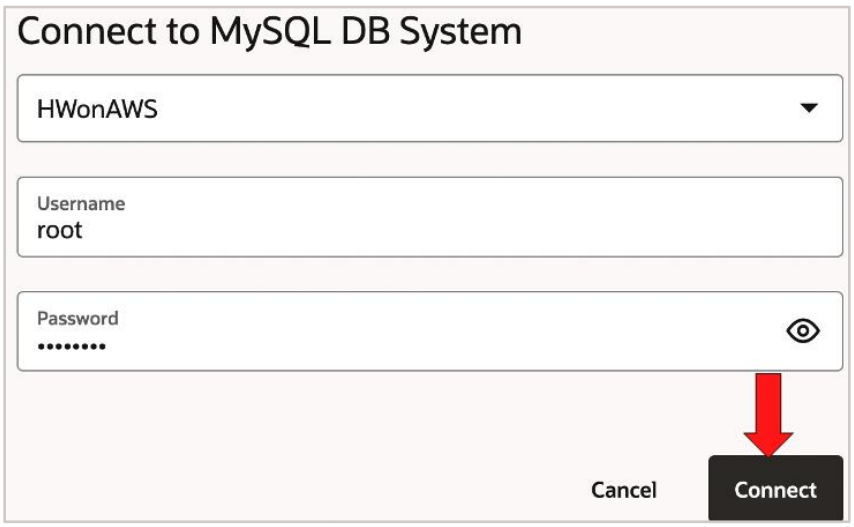
66. From the 'Workspaces' tab, click the "Connect to MySQL DB System' button and then, select your MySQL HeatWave System (in our case we will be selecting "HWonAWS")



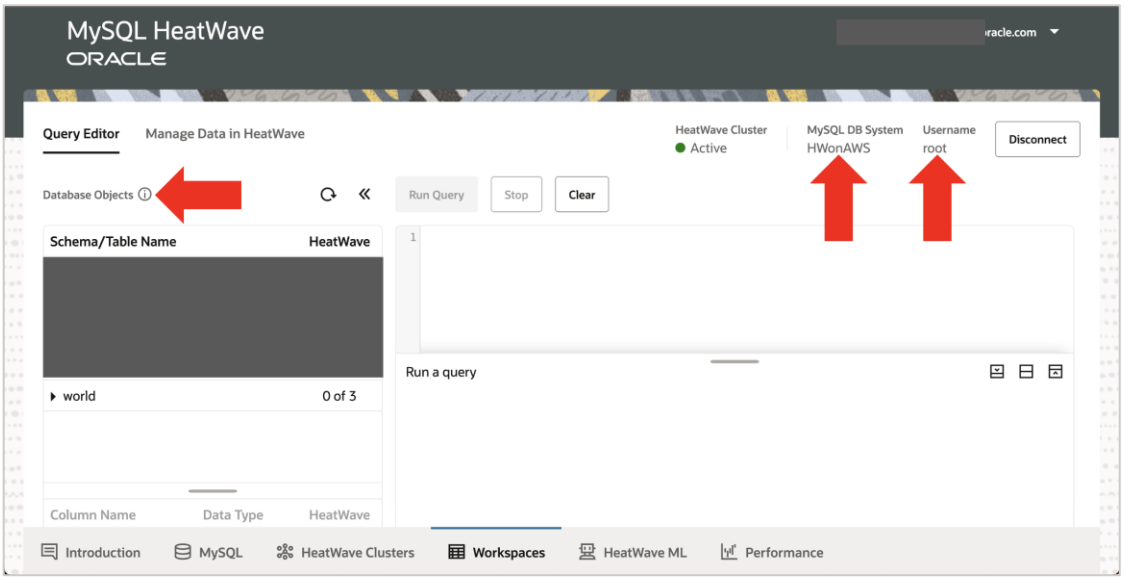




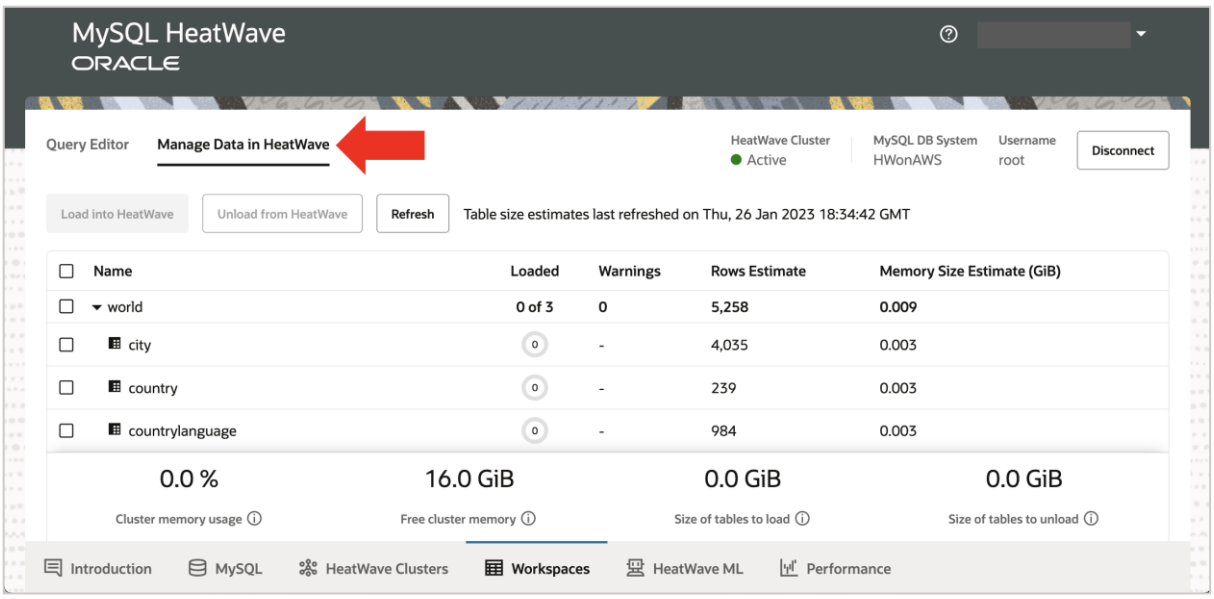
67. After selecting the appropriate MySQL DB System, enter the DB Username and Password. Click “Connect” afterwards



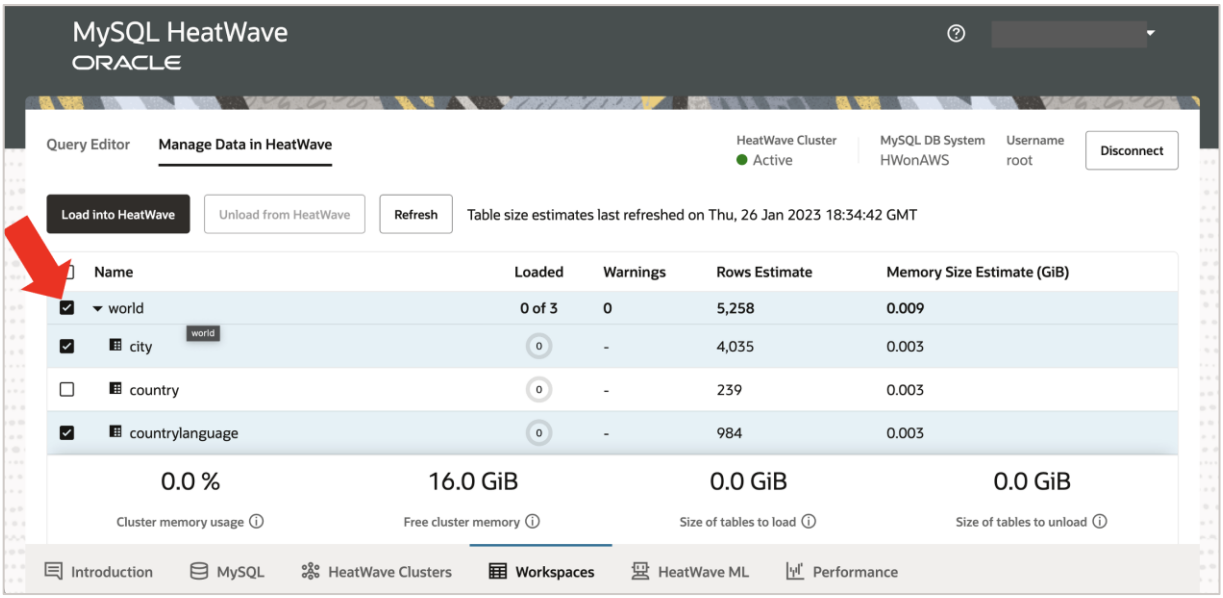
68. Once you connect, your MySQL DB System name alongside the username that was used to connect will be displayed on the top right. You can also view all the “Schemas” that are currently in the MySQL InnoDB Storage engine, under the “Database Objects”



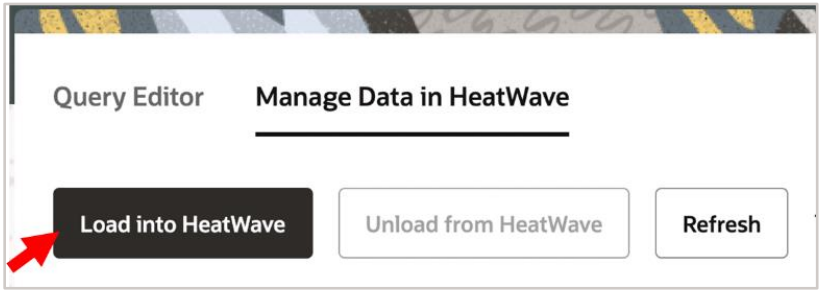
69. On the ‘Workspaces’ tab, switch to “Manage Data in HeatWave” from ‘Query Editor’



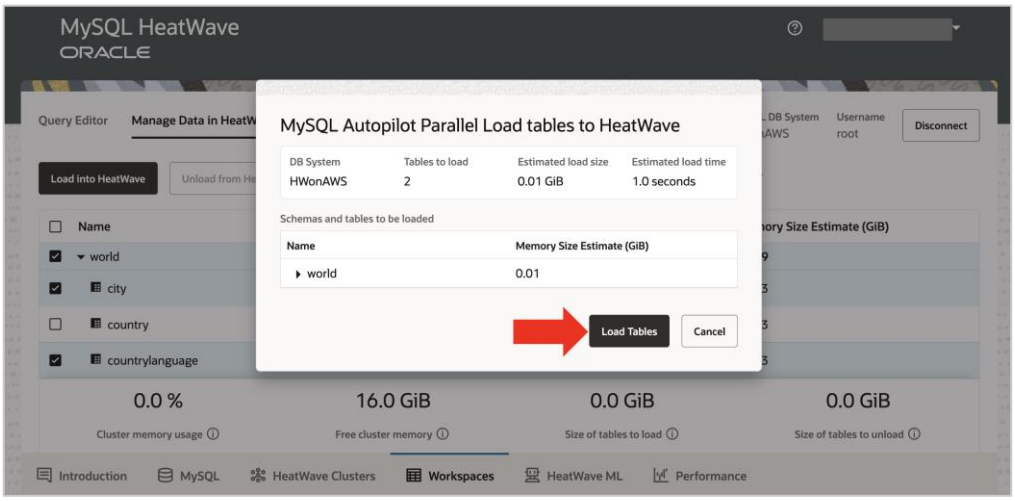
70. This screen will show a list of the schemas and tables that are loaded in the MySQL DB System. From this screen, you can select the schemas and tables to load into MySQL HeatWave’s in-memory engine. Select the databases/tables you want to load by checking the box next to the appropriate database(s)/table(s). (For this guide, instead of loading the whole “world” database, we will only load the “city” table and “countrylanguage” table)



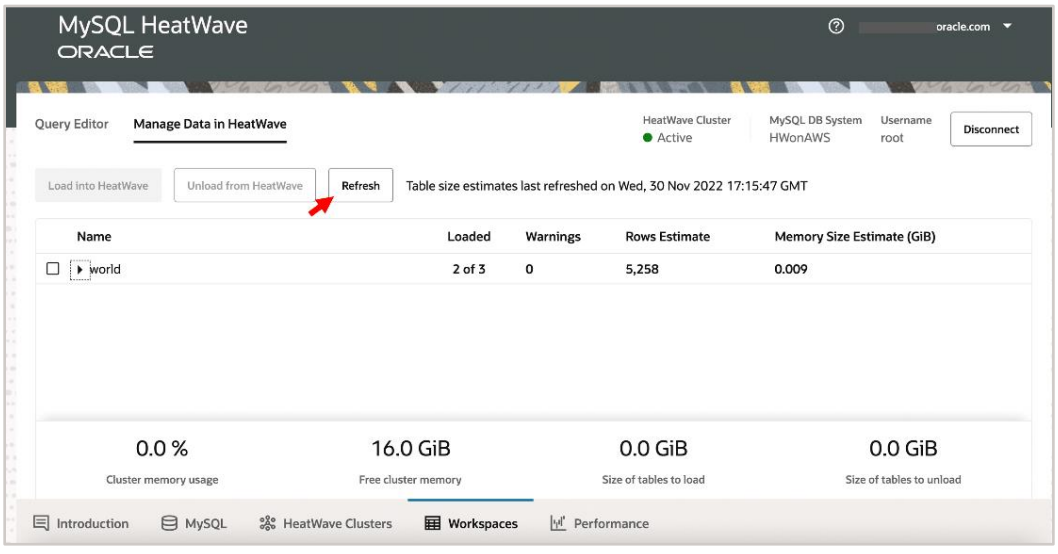
71. After you have selected all the tables you want to load into HeatWave, click the ‘Load into HeatWave’ button on the top left.



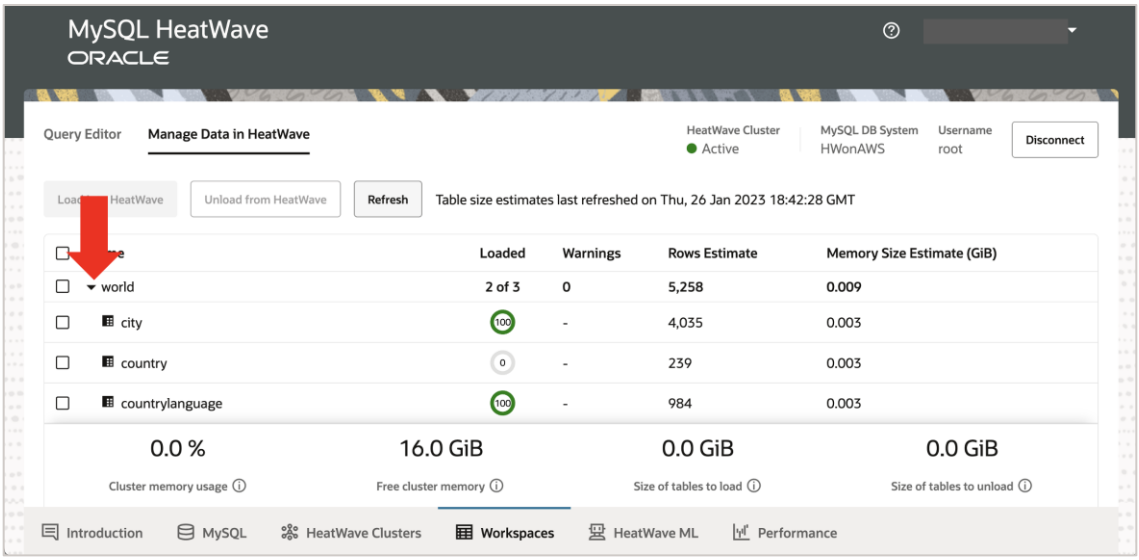
72. After you’ve clicked the ‘Load into HeatWave’ button, a popup will appear, which will show you information about the tables that will be loaded and how much memory HeatWave will consume. The estimated time required to load the tables into memory will also be displayed. Click “Load Tables” when the below popup appears.

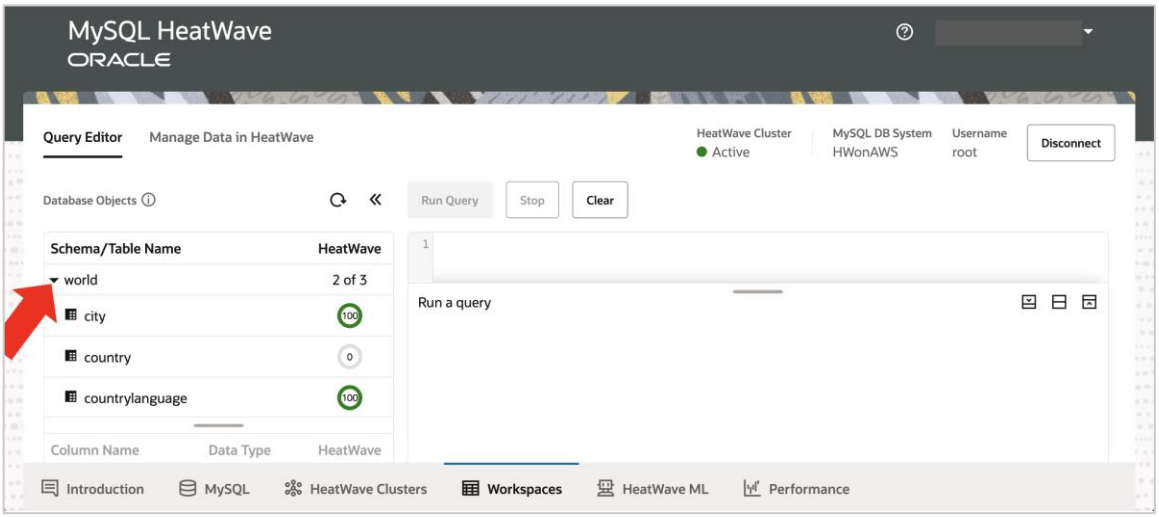


73. You can click the ‘Refresh’ button to view the progress of how much data has been loaded into HeatWave. Depending on the size of your data, it may take a few minutes to complete the load.



74. To confirm if your data is 100% loaded, expand the schema by clicking the tiny arrow next to the Schema name from either the “Query Editor” or “Manage Data in HeatWave” on the ‘Workspaces’ tab.





75. You now have a complete MySQL HeatWave cluster.

76. Congratulations, you’ve now successfully migrated your data from Amazon Aurora MySQL to MySQL HeatWave on AWS!

    To learn more about using HeatWave, please visit our documentation.

V. Appendix

Section L: Performing the util.dumpInstance()and util.loadDump() utility to and from a local filesystem

77. For relatively small databases, you can create the dump files on your local system. Although, you need to transfer them to the AWS EC2 instance using the copy utility of your choice, depending on the operating system you chose for your EC2 instance. (MySQL Shell must be installed on the systems from where you intend to run the util.dumpInstance() and util.loadDump() utility, setting up the `credentials` file is not required here)

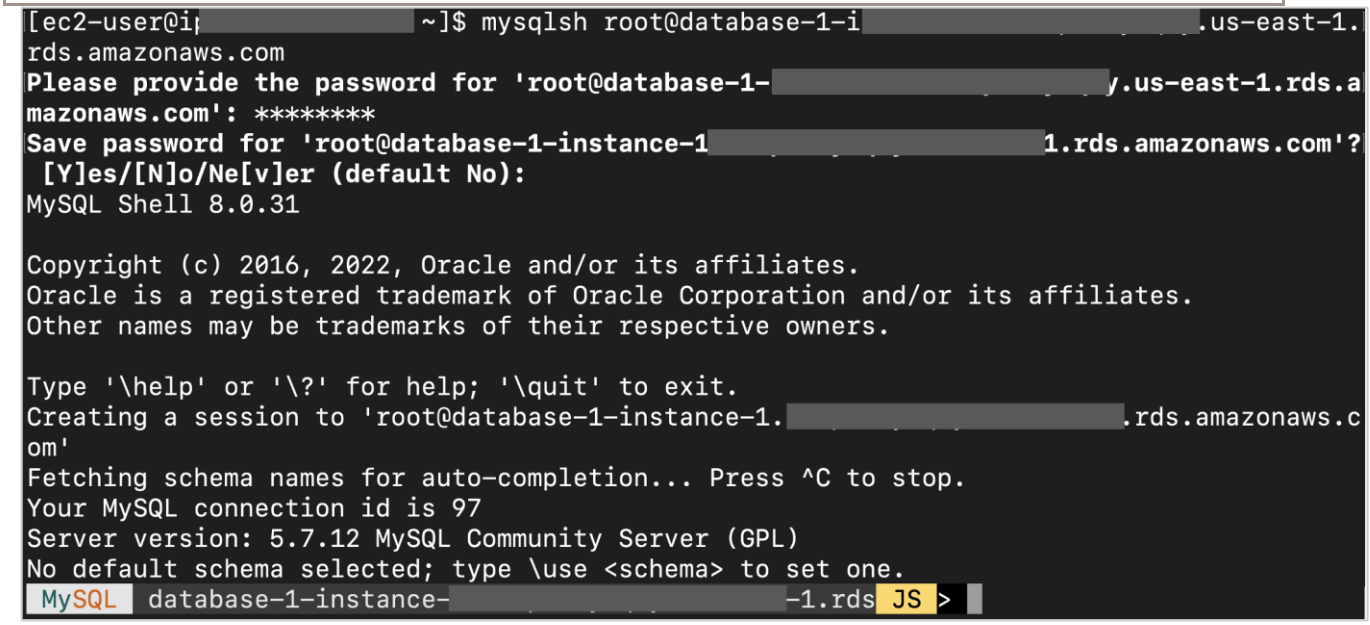
78. In this Section, we will showcase how to perform the dumpInstance() utility from the Amazon Aurora MySQL instance into a local filesystem. The local filesystem used for the dumpInstance() in this guide is the AWS EC2 instance that was shown in Step 11.

79. Connect to your Amazon Aurora MySQL Server using MySQL Shell

```
ec2-user $ mysqlsh <username>@<localhost/ip>
```

or

```
ec2-user $ mysqlsh -u <username> -h <localhost/ip> -P <portnumber> -p
```



80. Make sure you are in JavaScript mode by typing `\js` and execute the `dumpInstance` utility to export the dump data into your local filesystem

```
MySQL JS> \js
```

```
MySQL JS> util.dumpInstance("/home/ec2-user/sampledump", {"ocimds": "true",
"compatibility": ["strip_restricted_grants", "strip_definers"], users:
"true", dryRun:"true", consistent: "false"})
```

```
 MySQL  database-1.(                          east-1.rds.amazonaws JS > util.dumpInstance("/home/
ec2-user/sampledump", {"ocimds": "true", "compatibility": ["strip_restricted_grants", "strip_d
efiners"], users: "true", dryRun:"true", consistent: "false"})
dryRun enabled, no locks will be acquired and no files will be created.
Initializing - done
0 out of 4 schemas will be dumped and within them 0 tables, 0 views.
2 out of 3 users will be dumped.
Gathering information - done
WARNING: The dumped value of gtid_executed is not guaranteed to be consistent
Checking for compatibility with MySQL Database Service 8.0.31
NOTE: MySQL Server 5.7 detected, please consider upgrading to 8.0 first.
Checking for potential upgrade issues.
The MySQL server at
database-1.                          east-1.rds.amazonaws.com:3306, version
5.7.12 - MySQL Community Server (GPL), will now be checked for compatibility
issues for upgrade to MySQL 8.0.31...

1) MySQL 8.0 syntax check for routine-like objects
```

```
16) Check for invalid table names and schema names used in 5.7
  No issues found

Errors:   0
Warnings: 1
Notices:  0

NOTE: No fatal errors were found that would prevent an upgrade, but some potential issues were
 detected. Please ensure that the reported issues are not significant before upgrading.
NOTE: User 'rdsadmin'@'localhost' had restricted privileges (CREATE TABLESPACE, FILE, RELOAD,
SHUTDOWN, SUPER) removed
NOTE: User 'root'@'%' had restricted privileges (INVOKE COMPREHEND, INVOKE LAMBDA, INVOKE SAGE
MAKER, LOAD FROM S3, RELOAD, SELECT INTO S3) removed
Compatibility issues with MySQL Database Service 8.0.31 were found and repaired. Please review
 the changes made before loading them.
Validating MDS compatibility - done
Writing global DDL files
Writing users DDL
Writing DDL - done
Starting data dump
0% (0 rows / ~5.30K rows), 0.00 rows/s, 0.00 B/s uncompressed, 0.00 B/s compressed
 MySQL  database-1.                          -east-1.rds world JS >
```

Note:

- `dumpInstance` SYNTAX: `util.dumpInstance(outputUrl[, options])`
- `/home/opc/sampledump` is the `outputUrl`. Here, you can specify an absolute path or a path relative to the current working directory for your local filesystem.
- `sampledump` is the directory under which all the exported dump files will be stored in EC2. The `sampledump` directory must not exist or if it does, the directory should be empty
- Add the `consistent: false` option, if and only if, your `dump` utility produces "Errors" regarding "table locks" (MySQLSH 52002: See Steps 47/48 for more information)
- The `util.dumpInstance()` utility will take a dump of all the databases except "mysql, sys, performance schema, and information schema". The dump comprises of DDL files for the schema

structure and tab-separated .tsv files containing the actual data. Additionally, you can also use `util.dumpSchemas()` or `util.dumpTables()` if you only want to dump specific schemas or tables. The three dump utilities can export the data into:

- a) Object Storage bucket in Oracle Cloud
- b) S3-compatible buckets
- c) local filesystem
- This Section showcases option c). For more information, refer: https://dev.mysql.com/doc/mysql-shell/8.0/en/mysql-shell-utilities-dump-instance-schema.html#mysql-shell-utilities-dump-opt-run
- The `dryRun` option runs the export command but does not generate any output export file. It displays information about what would be dumped with the specified set of options, and about the results of MySQL HeatWave compatibility checks (if the `ocimds` option is specified, which is required for this guide), but does not proceed with the dump. Setting this option enables you to list out all the compatibility issues before starting the dump. The default is false. You can read more about the utility options at https://dev.mysql.com/doc/mysql-shell/8.0/en/mysql-shell-utilities-dump-instance-schema.html#mysql-shell-utilities-dump-opt-control
- Setting the `ocimds: true` option ensures compatibility of the export dump with MySQL HeatWave.
- Primary keys are required on every table for using MySQL HeatWave.
- If you can't seem to solve an error during the `dryRun`, contact a MySQL Solution Engineer for guidance: https://go.oracle.com/LP=132857?src1=:ow:o:s:po:::&intcmp=:ow:o:s:po:::
- To understand the `dumpInstance()`, `dumpSchemas()`, or `dumpTables()` utility in more detail, refer to this website: https://dev.mysql.com/doc/mysql-shell/8.0/en/mysql-shell-utilities-dump-instance-schema.html

81. Once you have executed the command in Step 80 and did not see any additional errors or warnings, execute the same Step 80 command. Although, this time change the `dryRun` option to false

```
MySQL JS> util.dumpInstance("/home/ec2-user/sampledump", {"ocimds": "true",
"compatibility": ["strip_restricted_grants", "strip_definers"], users:
"true", dryRun:"false", consistent: "false"})
```

```
 MySQL  database-                          us-east-1.rds  world  JS > util.dumpInstance("/home/ec2
-user/sampledump", {"ocimds": "true", "compatibility": ["strip_restricted_grants", "strip_defi
ners"], users: "true", dryRun:"false", consistent: "false"})
Initializing - done
1 out of 5 schemas will be dumped and within them 3 tables, 0 views.
2 out of 3 users will be dumped.
Gathering information - done
WARNING: The dumped value of gtid_executed is not guaranteed to be consistent
Checking for compatibility with MySQL Database Service 8.0.31
NOTE: MySQL Server 5.7 detected, please consider upgrading to 8.0 first.
Checking for potential upgrade issues.
The MySQL server at
database-1.                          -east-1.rds.amazonaws.com:3306, version
5.7.12 - MySQL Community Server (GPL), will now be checked for compatibility
issues for upgrade to MySQL 8.0.31...

1) MySQL 8.0 syntax check for routine-like objects
```

```
NOTE: Progress information uses estimated values and may not be accurate.
Writing schema metadata - done
Writing DDL - done
Writing table metadata - done
Starting data dump
100% (5.30K rows / ~5.30K rows), 0.00 rows/s, 0.00 B/s uncompressed, 0.00 B/s compressed
Dump duration: 00:00:00s
Total duration: 00:00:00s
Schemas dumped: 1
Tables dumped: 3
Uncompressed data size: 194.62 KB
Compressed data size: 91.71 KB
Compression ratio: 2.1
Rows written: 5302
Bytes written: 91.71 KB
Average uncompressed throughput: 194.62 KB/s
Average compressed throughput: 91.71 KB/s
 MySQL  database-1                          us-east-1.rds  world  JS >
```

- Note: once the dump process is complete, MySQL Shell will display a summary of the dump process like the one shown in the above image.

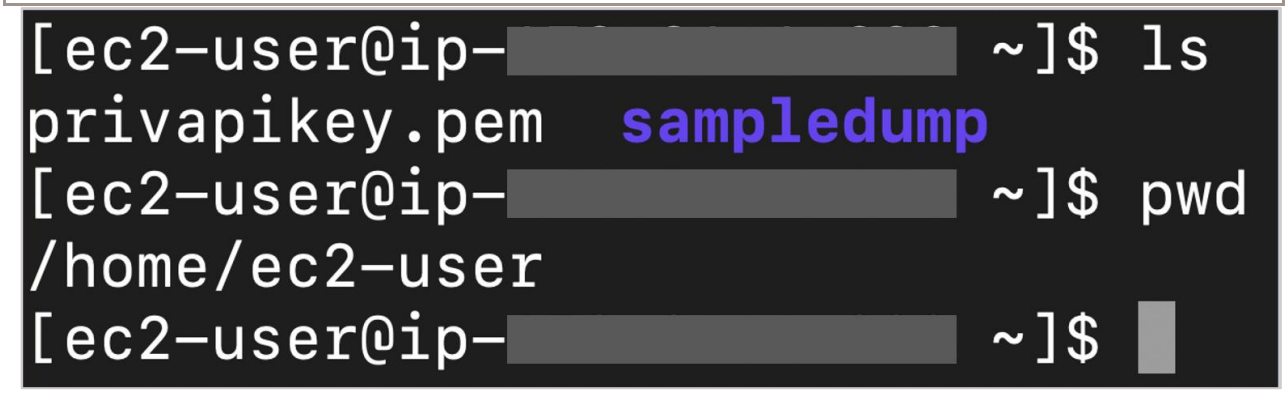
82. Go back to your local filesystem and locate the dump files under the `sampledump` directory, to confirm if the dump was successful (in our case, the EC2 instance).

```
[ec2-user@ip                ~]$
[ec2-user@ip                ~]$ ls
mysql-shell-8.0.31-1.el8.x86_64.rpm  privapikey.pem  sampledump  world-db  world-db.zip
[ec2-user@ip-               ~]$
[ec2-user@ip-               ~]$ cd sampledump
[ec2-user@ip-               sampledump]$
[ec2-user@ip-               sampledump]$ ls
@.done.json                  world@city.json                    world@countrylanguage.json
@.json                       world@city.sql                     world@countrylanguage.sql
@.post.sql                   world@country@@0.tsv.zst           world@country.sql
@.sql                        world@country@@0.tsv.zst.idx       world.json
@.users.sql                  world@country.json                 world.sql
world@city@@0.tsv.zst        world@countrylanguage@@0.tsv.zst
world@city@@0.tsv.zst.idx    world@countrylanguage@@0.tsv.zst.idx
[ec2-user@ip-               sampledump]$
```

83. Now, transfer the `sampledump` directory to the AWS EC2 instance using the copy utility of your choice, depending on the operating system you chose for your EC2 instance. One way to do this is to use the `scp` command.

84. After you have copied over your `sampledump` directory to the AWS EC2 instance, login to that EC2 instance and retrieve the path to the `sampledump` directory.

```
ssh -i <path/to/you-private-ssh-key> ec2-user@<ec2-Public-DNS>
```



- Note: by looking at the above image, the `sampledump` directory location for this guide will hence be `/home/ec2-user/sampledump`

85. Make sure you are logged in to that EC2 instance, and then login to your MySQL HeatWave instance using MySQL Shell to load those dump files.

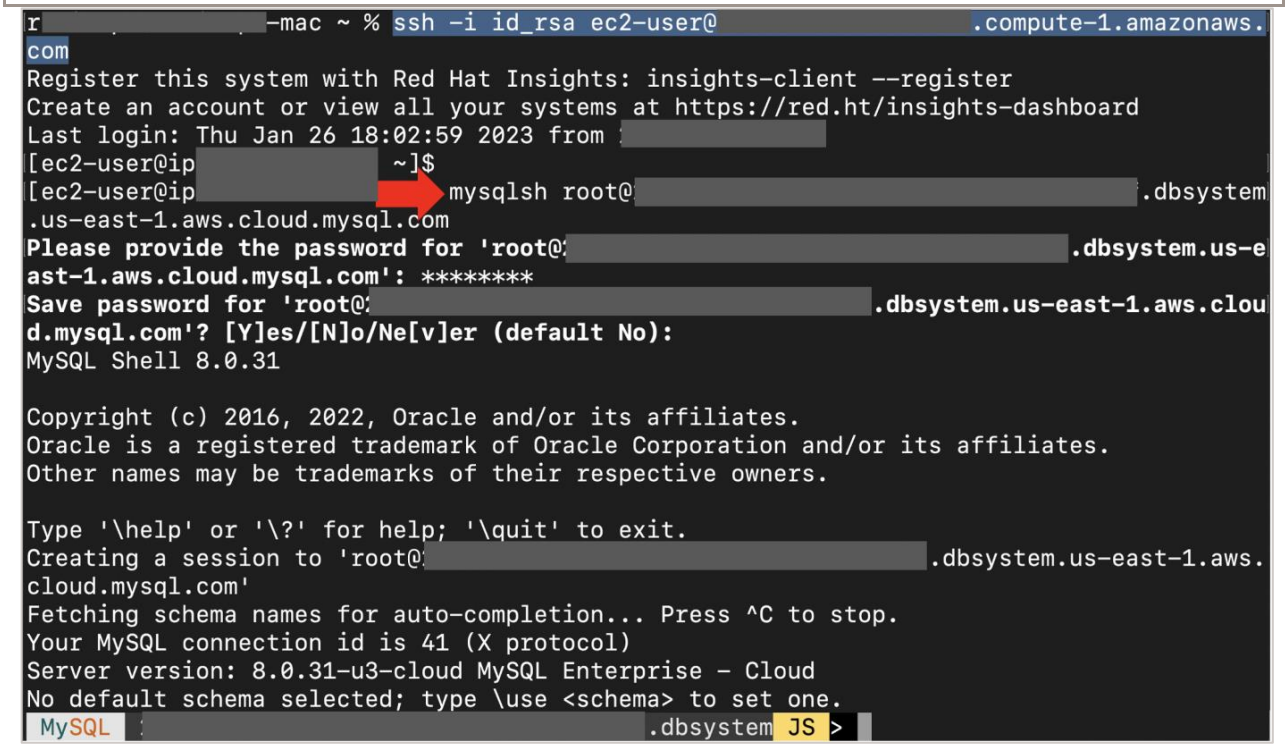
```
ssh -i <path/to/you-private-ssh-key> ec2-user@<ec2-Public-DNS>
```

then:

```
$ mysqlsh <username>@<hostname>
```

or

```
$ mysqlsh -u <username> -h <hostname> -P <portnumber> -p
```



86. It is now time to load our sample database “world”, that was dumped from our Amazon Aurora MySQL instance to the local filesystem, which we later transferred to the AWS EC2 instance using the copy utility of your choice. Inside MySQL Shell, make sure you are in JavaScript mode of MySQL Shell by executing `\js` and then, execute the `loaddump` utility to import the dumped data from AWS EC2 instance into MySQL HeatWave.

```
MySQL SQL> \js
```

```
MySQL JS> util.loadDump("/home/ec2-user/sampledump", {progressFile:
"/home/ec2-user/progressfile.json", ignoreVersion: "true", loadUsers: "true",
dryRun: "true"})
```

```
MySQL                                   .dbsystem.us-east-1.aws JS > util.loadDump("/home/
ec2-user/sampledump", {progressFile: "/home/ec2-user/progressfile.json", ignoreVersion: "true"
, loadUsers: "true", dryRun: "true"})
Loading DDL, Data and Users from '/home/ec2-user/sampledump' using 4 threads.
Opening dump...
dryRun enabled, no changes will be made.
Target is MySQL 8.0.31-u3-cloud (MySQL Database Service). Dump was produced from MySQL 5.7.12
WARNING: Destination MySQL version is newer than the one where the dump was created. Loading d
umps from different major MySQL versions is not fully supported and may not work. The 'ignoreV
ersion' option is enabled, so loading anyway.
Scanning metadata - done
Checking for pre-existing objects...
Executing common preamble SQL
Executing DDL - done
Executing view DDL - done
Starting data load
Executing user accounts SQL...
NOTE: Skipping CREATE/ALTER USER statements for user 'root'@'%'
NOTE: Filtered statement with restricted grants: GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP
,PROCESS,REFERENCES,INDEX,ALTER,SHOW DATABASES,CREATE TEMPORARY TABLES,LOCK TABLES,EXECUTE,REP
LICATION SLAVE,REPLICATION CLIENT,CREATE VIEW,SHOW VIEW,CREATE ROUTINE,ALTER ROUTINE,CREATE US
ER,EVENT,TRIGGER ON *.* TO 'rdsadmin'@'localhost' WITH GRANT OPTION; -> GRANT SELECT, INSERT,
UPDATE, DELETE, CREATE, DROP, PROCESS, REFERENCES, INDEX, ALTER, SHOW DATABASES, CREATE TEMPOR
ARY TABLES, LOCK TABLES, EXECUTE, REPLICATION SLAVE, REPLICATION CLIENT, CREATE VIEW, SHOW VIE
W, CREATE ROUTINE, ALTER ROUTINE, CREATE USER, EVENT, TRIGGER ON *.* TO 'rdsadmin'@'localhost'
 WITH GRANT OPTION;
NOTE: Skipping GRANT statements for user 'root'@'%'
Executing common postamble SQL
0% (0 bytes / 194.62 KB), 0.00 B/s, 3 / 3 tables done
Recreating indexes - done
No data loaded.
0 warnings were reported during the load.
MySQL                                   .dbsystem.us-east-1.aws JS >
```

- Note:
  - `loadDump` SYNTAX: `util.loadDump(url[, options])`
  - `/home/opc/sampledump` is the `url`. Here, you can specify the path to a local directory containing the dump files
  - The `util.loadDump()` utility will use the DDL files and tab-separated .tsv data files to set up the server instance or schema in the target MySQL instance, then loads the data. For more information, refer to: https://dev.mysql.com/doc/mysql-shell/8.0/en/mysql-shell-utilities-load-dump.html
  - Change the filesystem path to match with what you have.

87. Once you have executed the command in Step 86 and did not see any errors or warnings, execute the same Step 86 command. Although, this time change the dryRun option to false

```
MySQL JS> util.loadDump("/home/ec2-user/sampledump", {progressFile:
"/home/ec2-user/progressfile.json", ignoreVersion: "true", loadUsers: "true",
dryRun: "false"})
```

```
 MySQL                                .dbsystem.us-east-1.aws JS > util.loadDump("/home/
ec2-user/sampledump", {progressFile: "/home/ec2-user/progressfile.json", ignoreVersion: "true"
, loadUsers: "true", dryRun: "false"})
Loading DDL, Data and Users from '/home/ec2-user/sampledump' using 4 threads.
Opening dump...
Target is MySQL 8.0.31-u3-cloud (MySQL Database Service). Dump was produced from MySQL 5.7.12
WARNING: Destination MySQL version is newer than the one where the dump was created. Loading d
umps from different major MySQL versions is not fully supported and may not work. The 'ignoreV
ersion' option is enabled, so loading anyway.
Scanning metadata - done
Checking for pre-existing objects...
Executing common preamble SQL
Executing DDL - done
Executing view DDL - done
Starting data load
Executing user accounts SQL...
NOTE: Skipping CREATE/ALTER USER statements for user 'root'@'%'
NOTE: Filtered statement with restricted grants: GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP
,PROCESS,REFERENCES,INDEX,ALTER,SHOW DATABASES,CREATE TEMPORARY TABLES,LOCK TABLES,EXECUTE,REP
LICATION SLAVE,REPLICATION CLIENT,CREATE VIEW,SHOW VIEW,CREATE ROUTINE,ALTER ROUTINE,CREATE US
ER,EVENT,TRIGGER ON *.* TO 'rdsadmin'@'localhost' WITH GRANT OPTION; -> GRANT SELECT, INSERT,
UPDATE, DELETE, CREATE, DROP, PROCESS, REFERENCES, INDEX, ALTER, SHOW DATABASES, CREATE TEMPOR
ARY TABLES, LOCK TABLES, EXECUTE, REPLICATION SLAVE, REPLICATION CLIENT, CREATE VIEW, SHOW VIE
W, CREATE ROUTINE, ALTER ROUTINE, CREATE USER, EVENT, TRIGGER ON *.* TO 'rdsadmin'@'localhost'
 WITH GRANT OPTION;
NOTE: Skipping GRANT statements for user 'root'@'%'
Executing common postamble SQL
100% (194.62 KB / 194.62 KB), 0.00 B/s, 3 / 3 tables done
Recreating indexes - done
3 chunks (5.30K rows, 194.62 KB) for 3 tables in 1 schemas were loaded in 0 sec (avg throughpu
t 194.62 KB/s)
0 warnings were reported during the load.
 MySQL                                .dbsystem.us-east-1.aws JS >
```

- Note: once the load process is complete, MySQL Shell will display a summary of the dump process like the one shown in the image above.

88. After your import command has completed successfully in the previous step, you can verify the schemas and tables imported by running the following commands in `\sql` mode:

```
MySQL JS> \sql
MySQL SQL> SHOW SCHEMAS;
MySQL SQL> SHOW TABLES IN world;
```

```
 MySQL                                .dbsystem.us-east-1.aws.cloud JS > \sql
Switching to SQL mode... Commands end with ;
 MySQL                                .dbsystem.us-east-1.aws.cloud.mysql SQL > SHOW SCHEMAS;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| mysql_autopilot    |
| performance_schema |
| sys                |
| world              |
+--------------------+
6 rows in set (0.0015 sec)
 MySQL                                .dbsystem.us-east-1.aws.cloud.mysql SQL > SHOW TABLES IN world;
+-----------------+
| Tables_in_world |
+-----------------+
| city            |
| country         |
| countrylanguage |
+-----------------+
3 rows in set (0.0018 sec)
 MySQL                                .dbsystem.us-east-1.aws.cloud.mysql SQL >
```

Connect with us

Call +**1.800.ORACLE1** or visit **oracle.com**. Outside North America, find your local office at: **oracle.com/contact**.

blogs.oracle.com facebook.com/oracle twitter.com/oracle