Migration Guide: Amazon Aurora to MySQL HeatWave on Oracle Cloud Infrastructure (OCI)
**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

Purpose statement  
Disclaimer  
What is MySQL HeatWave  
Before you start  

I. Preparing your AWS environment  
  Section A: Prerequisites  
  Section B: Create an EC2 Instance and configure your SSH keys  
  Section C: Install MySQL Shell on your EC2 instance  

II. Exporting the Aurora database  
  Section D: In OCI, create an Object Storage Bucket  
  Section E: Add an API Key and save the Configuration File  
  Section F: Paste the Configuration File from Section E in your EC2 instance  
  Section G: Connect to your Amazon Aurora MySQL Server using MySQL Shell and run the util.dumpInstance() utility  

III. Importing the database  
  Section H: Navigate to the Object Storage bucket to confirm if the dump was successful  
  Section J: Create a Compute Instance and configure your SSH keys  
  Section K: Create a MySQL HeatWave System and import the dumped data using PAR URL  
  Section L: Connect to your Compute Instance and install MySQL Shell  
  Section M: Connect to your MySQL HeatWave System and verify if the data was imported successfully  

IV. Loading data into MySQL HeatWave  
  Section N: Create a HeatWave Cluster  
  Section O: Load data into the HeatWave Cluster  

V. Appendix  
  Section P: Create a new SSH Key pair and connect to your Compute Instance  
  Section Q: Connect to your Compute Instance using MySQL Shell and execute the util.loadDump() utility  
  Section R: Performing the util.dumpInstance() and util.loadDump() utility to and from a local filesystem
**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 is 6.5X faster than Amazon Redshift at half the cost, 7X faster than Snowflake at one-fifth the cost, and 1,400X faster than Amazon Aurora at half the cost. 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. Benchmarks demonstrate that, on average, HeatWave ML produces more accurate results than Amazon Redshift ML, trains models 25X faster at 1% of the cost, and scales as more nodes are added.

[Learn more about MySQL HeatWave](https://www.oracle.com/mysql/)

**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 should have an Amazon Aurora database instance installed. To migrate using the method that is shown in this guide, you will need a source Aurora MySQL instance based on MySQL 5.7 or later. For more information, see: [https://dev.mysql.com/doc/mysql-shell/8.0/en/mysql-shell-utilities-dump-instance-schema.html#mysql-shell-utilities-dump-opt-requirements](https://dev.mysql.com/doc/mysql-shell/8.0/en/mysql-shell-utilities-dump-instance-schema.html#mysql-shell-utilities-dump-opt-requirements)


4. You should have an account on Oracle Cloud Infrastructure (OCI) and be able to log in to it at [https://cloud.oracle.com/](https://cloud.oracle.com/).
   - If you do not have an account on OCI, you can sign-up for one at [https://www.oracle.com/mysql/free/](https://www.oracle.com/mysql/free/)
I. Preparing your AWS environment

Section A: Prerequisites

1. As mentioned, 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 later. For this guide, we have chosen an Amazon Aurora MySQL 5.7 (when applicable, you should always execute the commands shown in this guide as a root/admin user i.e., both for your AWS environment and OCI).

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

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

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

Let’s look at all the databases on our Amazon Aurora MySQL Server and the tables in the world database. We will be exporting the world database from Amazon Aurora MySQL to MySQL HeatWave on OCI.

```
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. When on “EC2 Dashboard” page, look for a “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, choose “Red Hat” either version “Linux 8 or 9”, here we have chosen 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.
9. When you click “Create new key pair”, a popup will appear asking you to “Create key pair”. Here, 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:
10. Next 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 one below which allows SSH from anywhere.
11. Once that is done, leave everything default and click “Launch instance”

12. Wait until your MySQL-EC2 “Instance state” is in “Running” before we can connect to it.

13. Once your EC2 instance is “Running”, open the Private SSH Key that we downloaded in Step 9 in a text editor of your choice.
14. Once you have opened your Private SSH Key in a text editor, copy the contents of the entire file as shown below:

15. 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.pem` inside your home directory. The guide uses the “nano” text editor, you can use a text editor of your own choice.
16. 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`
17. 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/user/id_rsa

18. 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
```

19. Now try to connect to the EC2 Instance we 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”.

20. We are now successfully connected to the EC2 instance.
**Section C: Install MySQL Shell on your EC2 instance**

21. 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 we created in Section B. It is now time to install MySQL Shell on the EC2 instance. We will be using 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/](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, do the following:

b. When prompted, click Run.
c. Follow the steps in the Setup Wizard.

**Installing MySQL Shell on Linux:**
Install MySQL Shell with this command:

a. `sudo yum install mysql-shell`

**Installing MySQL Shell on macOS:**
To install MySQL Shell on macOS, do the following:

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 as shown in the next step.

22. When you click “Download” as shown in Step 21, this page will come up. Right click on “No thanks, just start my download.” and select “Copy Link Address”
23. 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 install wget -y
$ wget https://dev.mysql.com/get/Downloads/MySQL-Shell/mysql-shell-8.0.31-1.el8.x86_64.rpm
```

```
[ec2-user@ip-****************]~$ sudo yum install wget -y
```

```
$ wget https://dev.mysql.com/get/Downloads/MySQL-Shell/mysql-shell-8.0.31-1.el8.x86_64.rpm
--2023-01-27 15:53:49--  https://dev.mysql.com/get/Downloads/MySQL-Shell/mysql-shell-8.0.31-1.el8.x86_64.rpm
Resolving dev.mysql.com (dev.mysql.com)... 96.7.17.219, 2600:1408:c400:1881::2e31, 2600:1408:c400:188c::2e31
Connecting to dev.mysql.com (dev.mysql.com)|96.7.17.219|:443... connected.
HTTP request sent, awaiting response... 302 Moved Temporarily
Location: https://cdn.mysql.com/Downloads/MySQL-Shell/mysql-shell-8.0.31-1.el8.x86_64.rpm [following]
--2023-01-27 15:53:49--  https://cdn.mysql.com/Downloads/MySQL-Shell/mysql-shell-8.0.31-1.el8.x86_64.rpm
Resolving cdn.mysql.com (cdn.mysql.com)... 23.56.12.246
Connecting to cdn.mysql.com (cdn.mysql.com)|23.56.12.246|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 30061380 (29M) [application/x-redhat-package-manager]
Saving to: 'mysql-shell-8.0.31-1.el8.x86_64.rpm'

2023-01-27 15:53:51 20.7M/s - 'mysql-shell-8.0.31-1.el8.x86_64.rpm' saved [30061380/30061380]

```
[ec2-user@ip-****************]~$
```

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

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

```
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 3a79bd29: NOKEY
error: Failed dependencies:
  libcrypto.so.1.1()(64bit) is needed by mysql-shell-8.0.31-1.el8.x86_64
  libcrypto.so.1.1(OPENVSSL_1_1_0)(64bit) is needed by mysql-shell-8.0.31-1.el8.x86_64
  libssl.so.1.1()(64bit) is needed by mysql-shell-8.0.31-1.el8.x86_64
  libssl.so.1.1(OPENVSSL_1_1_0)(64bit) is needed by mysql-shell-8.0.31-1.el8.x86_64
[ec2-user@ip-****************]~$
```

- Note: there were missing dependences when the rpm command was executed
25. To resolve the above dependency, we executed

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

Once all the required dependencies are installed, execute the same rpm command from Step 24

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

```
[ec2-user@ip-172-31-250-192 ~]$ 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 3a79bd29: NOKEY
Verifying...                                  [100%]
Preparing...                                  [100%]
Updating / installing...
  1:mysql-shell-8.0.31-1.el8  [100%]
[ec2-user@ip-172-31-250-192 ~]$
```

- Note: MySQL Shell was properly installed after all the dependencies were solved.
II. Exporting the Aurora database

Section D: In OCI, create an Object Storage Bucket

26. Once you have MySQL Shell installed on your EC2 instance and are ready to move data from your Amazon Aurora MySQL server to MySQL HeatWave on OCI, login to your Oracle Cloud account. https://cloud.oracle.com/

27. Click on the “Hamburger” menu and go to “Storage” > “Object Storage & Archive Storage” > “Buckets”
28. To create a bucket, click “Create Bucket”. In a later step, you will export your Amazon Aurora MySQL database to this bucket. Create all the resources shown in the guide in a dedicated “Compartment”, this guide uses the “oci labs” Compartment for the creation of all resources. If you don’t have a dedicated Compartment created, you can use your “root” Compartment for all your resources. For more information about Compartments, please visit: https://docs.oracle.com/en-us/iaas/Content/Identity/Tasks/managingcompartments.htm

29. A pop up will appear asking you to create your Bucket. Enter a “Bucket Name”. For this guide, we have chosen “Migration-Bucket”. Leave the “Default Storage Tier” as “Standard” and click “Create”
30. Once your Bucket is created and displayed on the “Buckets” page, click on your Bucket name and copy the “Namespace” value which can be found under the “General section” of “Bucket Information”

- Note: save the Bucket name as well as the Namespace value/string information into a notepad
Section E: Add an API Key and save the Configuration File

31. From the OCI Console, navigate to the top right corner where you will find a user-looking icon that says “Profile”. Click on the “Profile” icon.

Note: the above picture is how the menu will look like once you click on the “Profile” icon.

32. Once your “Profile” menu is expanded, click on “User settings”.

...
33. Once you click on ‘User settings’, you should land on a page like the below one “Identity” > “Users” > “User Details”
34. Scroll down when you are on the above ‘User Details’ page, until you see a “Resources” section on the left-hand side. Once the ‘Resources’ section is located, click on “API Keys”

![API Keys section](image)

35. Once you are on the ‘API Keys’ section, click on “Add API Key”

<table>
<thead>
<tr>
<th>Fingerprint</th>
<th>Created</th>
</tr>
</thead>
</table>
Note: when you click “Add API Key”, a popup will appear. On that popup:

1) Click “Generate API Key Pair”
2) Click “Download Private Key”
3) Click “Download Public Key”. Save those keys for later use, you cannot download the keys later
4) Click “Add” after you have downloaded both the Public and Private API Keys
36. Once you’ve clicked “Add” on the previous step, another popup will appear which will read “Configuration File Preview”. Select all the highlighted text under the “Configuration File Preview Read-only” section as shown in the image below and copy it. Save the copied content into a text document for later use.

- Note: click “Close” after you have saved the above “Configuration File Preview” into a notepad file. If you want to view the Configuration File again: navigate to this same API Keys page, select the appropriate ‘Fingerprint’ by looking at the ‘Date Created’. Click the three dots at the end of the selected row. Click ‘View Configuration file’. Look at the screenshot below:
Section F: Paste the Configuration File from Section E in your EC2 instance

37. Once you have created a bucket and added an API Key in OCI, go back to your AWS environment where Amazon Aurora and EC2 with MySQL Shell are installed. Connect to the EC2 instance and install nano on it. (You can use a text editor of your choice here)

```
$ ssh -i <path/to/your-private-ssh-key> ec2-user@<ec2-Public-DNS>
ec2-user $ sudo yum install nano -y
```

Register this system with Red Hat Insights: insights-client --register
Create an account or view all your systems at https://red.ht/insights-dashboard
Last login: Fri Jan 27 18:28:12 2023 from [edit]
[ec2-user@ip-172-31-1-232 ~]$ sudo yum install nano -y

```
Total Running transaction check
Running transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
Preparing : 1/1
  Installing : nano-5.6.1-5.el9.x86_64 1/1
Running scriptlet: nano-5.6.1-5.el9.x86_64 1/1
  Verifying : nano-5.6.1-5.el9.x86_64 1/1
Installed products updated.

Installed:
nano-5.6.1-5.el9.x86_64

Complete!
[ec2-user@ip-172-31-1-232 ~]$
```

38. After nano is installed on EC2, create a new directory called “.oci” inside your EC2 home directory. Next, go into the “.oci” directory and create a file called “config”. Paste the contents of the “Configuration File Preview” we copied in Step 36 into the newly created “config” file. The commands used to achieve this step for the guide, are listed below:

```
ec2-user $ mkdir ~/.oci
ec2-user $ cd .oci
ec2-user $ nano config
```

```
~]$ mkdir ~/.oci ~]
~]~]$ cd .oci .oci]
~]$ nano config
```
39. After the “config” file opens, paste the “Configuration File Preview” content from Step 36

```
[DEFAULT]
user=ocid1.user.oc1..aaa_
 tenancy=ocid1.tenancy
 region=us-ashburn-1
 key_file=<path to your private keyfile> # TODO
```

40. Once you have pasted the “Configuration File Preview” snippet into the “config” file, you will need to adjust the parameter where it says “key_file” with the file path to your own OCI Private API Key. Look at the example below:

```
[DEFAULT]
user=ocid1.user.oc1..aa_
 tenancy=ocid1.tenancy.oc1..aa_
 region=us-ashburn-1
 key_file=/home/ec2-user/privapikey.pem
```

- Note: If you need help uploading your Private API Key onto your EC2 instance, continue following the guide. (If you have already uploaded your API Key and have updated the “key_file” parameter for your “config” file, skip to the next Section)

41. After your new API Keys have successfully been added on OCI and you have created the “config” file on your EC2 instance, open your Private API Key in a text editor of your choice. The Private API Key will be the file without the word “public” in the file name. (You should have downloaded both the Private and Public API Keys in Step 35/36)

```
Name

oracleidentitycloudservice_2c5196-12-21-00-53.pem
oracleidentitycloudservice_2c5197-12-21-00-53_public.pem
```
42. Once you have opened your Private API Key in a text editor, copy the contents of the entire file as shown below:

43. After copying the contents, go back to your AWS EC2 instance where you have created the “config” file and have MySQL Shell installed. Create a new file there called `privapikey.pem`, for example. This guide used the “nano” text editor to create the `privapikey.pem` file on the EC2 instance. Choose a text editor of your own choice.

```
ec2-user $ cd
ec2-user $ nano privapikey.pem
```

```
[ec2-user@ip-]~]$ cd
[ec2-user@ip-]~]$ nano privapikey.pem
```
44. Once the `privapikey.pem` file opens up, paste the contents of the OCI Private API Key that we copied in Step 42, into this newly created `privapikey.pem` file. Save and close the file afterwards. If you are using nano,

- to paste the copied content: `command + V`
- to save the file: `control + O`
- to exit the file: `control + X`
45. After you have saved the Private API Key on your EC2 instance, grab the file path of the `privakkey.pem` and adjust the "key_file" parameter in the `.oci/config` file. To get the file path of your current working directory where you have the `privakkey.pem`, execute:
```
ec2-user $ ls
ec2-user $ pwd
```

![Image of file paths](image)

- Note: by looking at the above image, the **privakkey.pem** location for this guide will hence be `/home/ec2-user/privakkey.pem`. Go back to your `.oci` directory and adjust your `config` file accordingly.

```
[ec2-user@ip-]~]$ ls
[ec2-user@ip-]~]$ pwd
/home/ec2-user
[ec2-user@ip-]~]$ 
```

46. Save and close the `config` file after you have adjusted its "key_file" parameter.
Section G: Connect to your Amazon Aurora MySQL Server using MySQL Shell and run the `util.dumpInstance()` utility

47. 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>@<amazon-aurora-endpoint>
```

or

```
ec2-user $ mysqlsh -u <username> -h <amazon-aurora-endpoint> -P <portnumber> -p
```

Please provide the password for 'root@database-1.cluster-<clusterid>.us-east-1.rds.amazonaws.com': ********
Save password for 'root@database-1.cluster-<clusterid>.us-east-1.rds.amazonaws.com'? [Y]e
s/[N]o/Ne[try]er (default No):
MySQL Shell 8.0.31

Copyright (c) 2016, 2022, Oracle and/or its affiliates.
Oracle is a registered trademark of Oracle Corporation and/or its affiliates.
Other names may be trademarks of their respective owners.

Type '\help' or '\?' for help; '\quit' to exit.
Creating a session to 'root@database-1.cluster-<clusterid>.us-east-1.rds.amazonaws.com'
Fetching schema names for auto-completion... Press ^C to stop.
Your MySQL connection id is 22
Server version: 5.7.12 MySQL Community Server (GPL)
No default schema selected; type '\use <schema>' to set one.

```
MySQL database-1.cluster-<clusterid>.us-east-1.rds.amazonaws.com:JS>
```

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

48. 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`
49. Make sure you are in JavaScript mode by typing `\js` and execute the `dumpInstance` utility to export the dump data into the OCI Object Storage bucket.

MySQL JS> `\js`
MySQL JS> `util.dumpInstance("sampledump", {"osBucketName": "Migration-Bucket", "osNamespace": "idazzjlcqjz", "ocimds": "true", "compatibility": ["strip_restricted_grants", "strip_definers"], users: "true", dryRun:"true"})`

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 OCI
  b) S3-compatible buckets
  c) local filesystem


- 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 Service 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](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 the Object Storage bucket in OCI.
- Change the `osBucketName` and `osNamespace` to match with what you have when you created your bucket in OCI in an earlier step.
- 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:osp:sp::sntcmp=ow:osp:sp::](https://go.oracle.com/LP=132857?src1=ow:osp:sp::sntcmp=ow:osp:sp::)
- 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](https://dev.mysql.com/doc/mysql-shell/8.0/en/mysql-shell-utilities-dump-instance-schema.html)
50. Running the above Step 49 command may generate “Errors” regarding “table locks” (see Step 49 image). If you do encounter such a problem (if and only if) execute the same Step 49 command but this time add an additional option “consistent: false”

MySQL JS> util.dumpInstance("sampledump",{"osBucketName": "Migration-Bucket", "osNamespace": "idazzjlcjqzj", "ocimds": "true", "compatibility": ["strip_restricted_grants", "strip_definers"], 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 ptid_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- ast-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
   Check failed: Failed to load routine mysql.rds_import_binlog_ssl_material. The table mysql.proc is
   missing, corrupt, or contains bad data (internal code -6)
   More information:

2) Usage of db objects with names conflicting with new reserved keywords
   No issues found

3) Usage of utf8mb3 charset
   No issues found

4) Table names in the mysql schema conflicting with new tables in 8.0
   No issues found

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 SAGEMAKER, 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 / -5.39K rows), 0.00 rows/s, 0.00 B/s uncompressed, 0.00 B/s compressed
MySQL JS>util.dumpInstance("sampledump",{"osBucketName": "Migration-Bucket", "osNamespace": "idazzjlcjqzj", "ocimds": "true", "compatibility": ["strip_restricted_grants", "strip_definers"], users: "true", dryRun:"true", consistent: "false"})

• Note: the “table locks” error is not there anymore when consistent option is added
51. Once you have executed the command in Step 49/50 and did not see any additional errors or warnings, execute the same Step 49/50 command. Although, this time change the `dryRun` option to false.

```javascript
MySQL JS> util.dumpInstance("sampledump", {"osBucketName": "Migration-Bucket", "osNamespace": "idazzjlcqzj", "ocimds": "true", "compatibility": [{"strip_restricted_grants", "strip_definers"}], 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.cluster.us-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
   Check failed: Failed to load routine mysql.rds_import_binlog_ssl_material. The table mysql.proc is missing, corrupt, or contains bad data (internal code -6)

```
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 (INVOCATION COMPREHEND, INVOCATION LAMBDA, INVOCATION SAGEMAKER, 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
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
100% (6.39K 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: 00:00:01s
Schemas dumped: 1
Tables dumped: 3
Uncompressed data size: 194.62 KB
Compressed data size: 91.71 KB
Compression ratio: 2.1
Rows written: 5382
Bytes written: 91.71 KB
Average uncompressed throughput: 194.62 KB/s
Average compressed throughput: 91.71 KB/s
```

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

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

52. Go back to your OCI Object Storage bucket created in Step 30 and locate the dump files under the `sampledump` prefix.
Section I: Create a VCN and adjust the Security List

53. From the OCI Console, click on the “Hamburger” menu, but this time go to “Networking” > “Virtual Cloud Networks”

54. Once on the “Virtual Cloud Networks” page, click on “Start VCN Wizard”

- Note: a popup will appear saying “Start VCN Wizard”. Choose the “Create VCN with Internet Connectivity” option and click “Start VCN Wizard”
55. Give a name to your VCN, we have chosen Migration-VCN for this guide. Leave everything default, and click “Next”

56. You will be on “Review and Create” page after the previous step. Review your VCN information and click “Create”

57. Once all the Resources for your VCN have been created, click on “View Virtual Cloud Network”
58. You will land on the “Virtual Cloud Network Details” page. From here, locate your “Private Subnet” under the “Subnets” section.
59. Click on the “Private Subnet” from the above screen. A “Subnet Details” page will open up that will say “Private Subnet-<VCN-Name>”.

60. Scroll down on the above page until you see a “Security Lists” section. Click on “Security List for Private Subnet-<VCN-Name>”
61. On the Security List for Private Subnet page, click “Add Ingress Rules”

62. A popup will appear, for “Source CIDR” enter 0.0.0.0/0
Leave the IP Protocol as TCP, and Source Port Range as blank. For the “Destination Port Range” enter 3306,33060
For Description, you can write “MySQL Port Access”. Click “Add Ingress Rules” afterwards (Note: it is not recommended to use 0.0.0.0/0 Source CIDR for your production system)
63. Your Ingress Rules should be like those on the below image

<table>
<thead>
<tr>
<th>Stateless</th>
<th>Source</th>
<th>IP Protocol</th>
<th>Source Port Range</th>
<th>Destination Port Range</th>
<th>Type and Code</th>
<th>Allows</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>10.0.0.0/16</td>
<td>TCP</td>
<td>All</td>
<td>22</td>
<td>TCP traffic for port 22 (SSH) Remote Login Protocol</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No</td>
<td>0.0.0.0/0</td>
<td>ICMP</td>
<td>3, 4</td>
<td>3, 4</td>
<td>ICMP traffic for Destination Unreachable and Don’t Fragment was set</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No</td>
<td>10.0.0.0/16</td>
<td>ICMP</td>
<td>3</td>
<td>3</td>
<td>ICMP traffic for Destination Unreachable</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No</td>
<td>0.0.0.0/0</td>
<td>TCP</td>
<td>All</td>
<td>3306</td>
<td>TCP traffic for port 3306 (MySQL Port Access)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No</td>
<td>0.0.0.0/0</td>
<td>TCP</td>
<td>All</td>
<td>3306</td>
<td>TCP traffic for port 3306 (MySQL Port Access)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Section J: Create a Compute Instance and configure your SSH keys

64. From the OCI Console, click on the “Hamburger” menu again, and navigate to “Compute” > “Instances”

65. Click on “Create instance” when on the “Instances” page

66. Give your Compute Instance (VM) a name. We have chosen “MySQL-Compute” for this guide
67. Scroll down until you see an “Image and shape” section. Under “Image” make sure “Oracle Linux 8” is selected and for the “Shape”, select one that suits your needs. (Standard.E4.Flex recommended. This Compute Instance will be used to connect to the MySQL HeatWave instance and later to demonstrate MySQL Shell loadDump). Remember: The dumpInstance and loadDump speed depends on the number of OCPUs/vCPUS and the amount of the data that is being migrated.

68. Once the Image and Shape are selected, scroll down until you see the “Networking” section. Make sure you have the right Compartment selected and that the VCN we created earlier is selected (the guide used “Migration-VCN”). Also make sure that “Public Subnet” is selected under the Subnet option.
69. Lastly, scroll down until you see the “Add SSH keys” section. You can either generate a pair of new SSH keys, upload your own public key file, or paste it. The guide uses “Paste public keys” options since we already had an SSH Key pair downloaded.

- Note: after you have created/added your SSH keys, click “Create”. Refer to Section P of this guide if you need instructions on how to generate a new SSH key pair and then use that, to login into your Compute Instance.

70. Your Compute will soon be in an “ACTIVE” state afterwards

- Note: you can SSH into the Compute Instance by executing

```
ssh -i <path/to/you-private-ssh-key> opc@<compute-public-ip-address>
```

```
mac ~ % ssh -i .ssh/id_rsa opc@1.2.3.4
```
Section K: Create a MySQL HeatWave System and import the dumped data using PAR URL

71. Once the all the dump files have successfully been exported to the Object Storage Bucket in OCI, click on the “Hamburger” menu and go to “Databases” > “MySQL”

72. Once on the MySQL DB Systems page, click “Create DB System”
73. A new page will appear. Here, give a name to your MySQL DB System and select the “HeatWave” option.

- Note: after selecting “HeatWave”, create your Administrator credentials that will be used to manage the MySQL HeatWave database.
74. After creating the Administrator credentials, configure your Networking. Choose the VCN we created earlier (this guide uses Migration-VCN) and make sure the “Private Subnet” is selected.

75. For “Configure Placement” leave it as default. Afterwards, configure your DB hardware by choosing an appropriate HeatWave DB Shape. For the Data Storage Size, it is the size of your database. Be sure to make the size large enough for future growth.
76. Next, choose a “Backup Plan”

77. Once your Backup Plan is configured, lastly scroll down until you see “Show advanced options”. Click on it to expand
- Note: from the above screen, go to the “Data Import” tab.

78. On the “Data Import” tab, click on ‘Click here to create a PAR URL for an existing bucket’.

- Note: this will open a pop up which will look like the below one:
79. From the “Create PAR for existing bucket” screen, under “Select a bucket in <compartment-name>”, choose the bucket we created in Step 30. Specify an appropriate PAR expiration time once the bucket is selected.

- Note: once the bucket is selected, make sure the right Prefix is also selected if you had one, under “Configure prefix”. The prefix we used was `sampledump`.

Note: to learn what Pre-Authenticated Request (PAR) is, refer to: [https://docs.oracle.com/en-us/iaas/Content/Object/Tasks/usingpreauthenticatedrequests.htm](https://docs.oracle.com/en-us/iaas/Content/Object/Tasks/usingpreauthenticatedrequests.htm)
Note: click “Create and set PAR URL” once you have chosen the appropriate bucket, prefix, and expiration time for the PAR.

80. Once you have finished Step 79, the “PAR Source URL” field will automatically be populated by the URL. Click “Create” afterwards.

Note: using the “Data Import” “PAR Source URL” option, your data stored in OCI Object Storage bucket will automatically start loading while your MySQL DB System is creating. Once the MySQL DB System is “ACTIVE”, the data that was imported using the PAR URL will also be present, which we will see in subsequent steps.
81. Once you click Create in Step 80, your MySQL DB System will start “CREATING”

![MySQL DB System in Creating State]

- Note: your MySQL DB System State will change from CREATING TO “ACTIVE” once the DB System is ready.

![MySQL DB System in Active State]

82. Once your MySQL DB System is ACTIVE, a “Private IP Address” will be allocated to it, find it and copy it. You can find this Private IP under “DB System Information” > “Endpoint” section on the “DB System Details” page.

![MySQL DB System in Active State with Private IP Address]
• Note: you can navigate to the “DB System Details” page by going to the “Hamburger” menu in OCI. “Databases” > “MySQL” > “DB Systems”. Click on the name of your MySQL DB System to open the “DB System Details” page.

Note: copy the Private IP Address. You can now login to your MySQL DB System using MySQL Shell. Execute

```
$ mysqlsh <username>@<private-mysql-ip>
```

or

```
$ mysqlsh -u <username> -h <private-mysql-ip> -P <portnumber> -p
```
Section L: Connect to your Compute Instance and install MySQL Shell

83. Next, SSH into the Compute Instance we created in Step 70.

```
ssh -i <path/to/you-private-ssh-key> opc@<compute-public-ip-address>
```

84. Install MySQL Shell on the Compute Instance after SSHing into it. On the terminal execute

```
$ sudo yum install mysql-shell -y
```

```
[opc@mysql-compute ~]$ sudo yum install mysql-shell -y
```

```
Oracle Linux 8 Application Stream (x86_64)       64 kB/s  |   3.9 kB   00:00
Oracle Linux 8 Application Stream (x86_64) 2.2 MB/s |  42 MB    00:18
Oracle Linux 8 Addons (x86_64) 190 kB/s |  3.0 kB    00:00
Latest Unbreakable Enterprise Kernel Release 6 181 kB/s |  3.0 kB    00:00
Latest Unbreakable Enterprise Kernel Release 6 61 MB/s |  60 MB    00:00
```

- Note: once MySQL Shell is successfully installed, you should see a “Complete!” message on your terminal as the below one:

```
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
Preparing:
  python39-setuptools-wheel-50.3.2-4.module+el8.6.0+20364 1/1
Installing:
  python39-pip-wheel-20.2.4-7.module+el8.6.0+20625+ee813d 1/4
  python39-libs-3.9.13-2.module+el8.7.0+20879+a85b87b0.x86_64 2/4
  mysql-shell-8.0.31-1.el8.x86_64 3/4
  mysql-shell-8.0.31-1.el8.x86_64 4/4
Running scriptlet: mysql-shell-8.0.31-1.el8.x86_64 4/4
Verifying:
  mysql-shell-8.0.31-1.el8.x86_64 1/4
  python39-libs-3.9.13-2.module+el8.7.0+20879+a85b87b0.x86_64 2/4
  python39-pip-wheel-20.2.4-7.module+el8.6.0+20625+ee813d 3/4
  python39-pip-wheel-20.2.4-7.module+el8.6.0+20625+ee813d 4/4
Installed:
  mysql-shell-8.0.31-1.el8.x86_64
  python39-libs-3.9.13-2.module+el8.7.0+20879+a85b87b0.x86_64
  python39-pip-wheel-20.2.4-7.module+el8.6.0+20625+ee813d
  python39-setuptools-wheel-50.3.2-4.module+el8.5.0+20364+c7fe1181.noarch
Complete!
```
Section M: Connect to your MySQL HeatWave System and verify if the data was imported successfully

85. Once MySQL Shell is installed on the Compute Instance, verify if your data was successfully imported into the MySQL HeatWave system. Using MySQL Shell, login to MySQL HeatWave via the Private IP

```
$ mysqlsh <username>@<private-mysql-ip>
```

or

```
$ mysqlsh -u <username> -h <private-mysql-ip> -P <portnumber> -p
```

86. Once logged in, switch to ‘SQL’ mode and execute SHOW SCHEMAS

```
MySQL JS> \sql
MySQL SQL> SHOW SCHEMAS;
```
IV. Loading data into MySQL HeatWave

Section N: Create a HeatWave Cluster

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

87. From your OCI Home Page, click the ‘Hamburger’ menu and go to “Databases” > “MySQL”. Once on the MySQL page, click on the name of your MySQL HeatWave System.

- Note: clicking on the name of your MySQL System will open the “DB System Details” page.
88. Once on “DB System Details” page, click on “More actions” and select “Add HeatWave cluster”

89. After finishing Step 88, a new page will open that will say “Add HeatWave cluster”. On this page, click on “Estimate node”

- Note: Based on your database size, the next steps will estimate the size of the HeatWave cluster that you will need to load the data into memory and run queries. The number of nodes in MySQL HeatWave on OCI can scale up to 64. Each node can handle approximately 800 GB of data.
90. Once you click on “Estimate node”, an “Estimate node” screen will appear that will look like the below:

![Estimate node](image1)

91. Here, click on “Generate estimate”. This will show you a list of all the databases that you have in your MySQL HeatWave system. Afterwards, you can select what tables and databases you want to load in-memory, from the list of databases that will appear after clicking “Generate estimate”.

![Generate estimate](image2)

- Note: once you click on “Generate estimate”, it may take several minutes to display your schema information.
92. This is what our screen looked like after hitting “Generate estimate”. It pulled up all the database schema that we currently have in MySQL.

93. From the above screen, you can either select the entire database/s or select individual tables that you want to load in-memory.
94. After you are done selecting the tables/databases you want to load in-memory, on that same screen, scroll down

- Note: instead of loading the whole database, for this guide we will only load the two tables (city and countrylanguage)

- Note: The end of that page shows us a summary of how many nodes will be required depending on the data we have selected (in our case as you can see, only 1 node was needed and 6 MB of memory will be used)
95. On the same page, there is a section that says “Load command”. Simply copy that line of code and keep it aside. In the upcoming steps we will take the copied command and execute it inside our MySQL HeatWave system, which will automatically load the selected data into memory using parallelism.

- Note: after copying the command, click “Apply estimated node”. This will change the number of nodes required to load the data that you have selected (depending on the data size).

- Note: after your node count has been updated, click “Add HeatWave Cluster” to finish adding the HeatWave cluster.
96. After clicking “Add HeatWave cluster”, you can see the status of the Cluster change to “Creating” on the “DB System Details” page.

- Note: to track how far along the Cluster creation process is, simply scroll down on that same page and look for “Work Requests” under the Resources section on the left.

97. Once the HeatWave Cluster is created and “ACTIVE”, this is what it should look like.

- Note: notice the “Cluster state: ACTIVE” status.
Section O: Load data into the HeatWave Cluster

98. Now it is time to login back into our MySQL HeatWave system, and load the data into memory. Login to your Compute instance with MySQL Shell installed to connect to MySQL HeatWave.

```bash
ssh -i <path/to/you-private-ssh-key> opc@<compute-public-ip-address>
```

then,

```bash
$ mysqlsh <username>@<private-mysql-ip>
```

or

```bash
$ mysqlsh -u <username> -h <private-mysql-ip> -P <portnumber> -p
```

99. Once logged in, change the MySQL Shell mode to \sql and execute the command we copied in Step 95 to load the data in-memory.

```sql
MySQL JS> \sql
MySQL SQL> CALL sys.heatwave_load(JSON_ARRAY('world'), JSON_OBJECT('exclude_list', JSON_ARRAY('world.country')));
```

(replace the load command with what you have)
100. Once you invoke that command, MySQL HeatWave will automatically load all your data without any user intervention. This is what the screen looks like:

```
MySQL> :33860+ sql sh SQL> CALL sys.heatwave_load(JSON_ARRAY('world'), JSON_OBJECT('exclude_list',null));

INITIALIZING HEATWAVE AUTO PARALLEL LOAD

Version: 1.39
Load Mode: normal
Load Policy: disable_unsupported_columns
Output Mode: normal

6 rows in set (0.5811 sec)

OFFLOAD ANALYSIS

 Verifying input schemas: 1
 User excluded items: 1

<table>
<thead>
<tr>
<th>SCHEMA</th>
<th>OFFLOADABLE TABLES</th>
<th>OFFLOADABLE COLUMNS</th>
<th>SUMMARY OF ISSUES</th>
</tr>
</thead>
<tbody>
<tr>
<td>'world'</td>
<td>2</td>
<td>9</td>
<td>1 table(s) are excluded by user</td>
</tr>
</tbody>
</table>

10 rows in set (0.5811 sec)
```

- Note: on the above image you can see that HeatWave tells us that 2 tables will be loaded in-memory and we have excluded one of the tables.

```
CAPACITY ESTIMATION

Default load pool for tables: TRANSACTIONAL
Default encoding for string columns: VARLEN (unless specified in the schema)
Estimating memory footprint for 1 schema(s)

<table>
<thead>
<tr>
<th>SCHEMA NAME</th>
<th>TOTAL OFFLOADABLE TABLES</th>
<th>TOTAL HEATWAVE MODE FOOTPRINT</th>
<th>ESTIMATED MySQL NODE FOOTPRINT</th>
<th>TOTAL STRING COLUMNS</th>
<th>DICTIONARY ENCODED COLUMNS</th>
<th>VARLEN ENCODED COLUMNS</th>
<th>ESTIMATED LOAD TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>'world'</td>
<td>2</td>
<td>6.11 MB</td>
<td>704.00 KIB</td>
<td>5</td>
<td>0</td>
<td>5</td>
<td>1.00 s</td>
</tr>
</tbody>
</table>

Sufficient MySQL host memory available to load all tables.
Sufficient HeatWave cluster memory available to load all tables.

13 rows in set (0.5811 sec)
```

- Note: on that same screen, HeatWave also tells us how much memory will be consumed after the data has loaded, alongside some other useful information.
Note: at the end, HeatWave breaks down each table that was loaded in-memory telling us if there were any warnings during the load process, how many threads were used, etc. Once all the data is loaded, HeatWave gives us a “Load Summary” where you can see it took 528 ms to load 2 tables which consisted of 9 columns.

101. You now have a complete MySQL HeatWave cluster.

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

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

Section P: Create a new SSH Key pair and connect to your Compute Instance

103. In OCI, go to the “Identity” > “Users” > “User Details” page and create your Compute Instance (refer to Section I of this guide on how to get to that page). Give your Compute Instance (VM) a name. We have chosen “MySQL-Compute” for this guide

104. Scroll down until you see an “Image and shape” section. Under “Image” make sure “Oracle Linux 8” is selected and for the “Shape”, select one that suits your needs. (Standard.E4.Flex recommended. This Compute Instance will be used to connect to the MySQL HeatWave instance and later to demonstrate MySQL Shell loadDump). Remember: The dumpInstance and loadDump speed depends on the number of OCPUs/vCPUS and the amount of the data that is being migrated)
105. Once the Image and Shape are selected, scroll down until you see the “Networking” section. Make sure you have the right Compartment selected and that the VCN we created earlier is selected (the guides used “Migration-VCN”). Also make sure that the “Public Subnet” is selected under the Subnet option.

106. Lastly, scroll down until you see “Add SSH keys” section. Make sure “Generate a key pair for me” is selected and download both the “private key” and “public key”
107. Click “Create” once you have your SSH keys downloaded. Your Compute will then start PROVISIONING and will be in a “RUNNING” state within a few minutes.

108. Once your Compute is RUNNING, open the private SSH key in a text editor of your choice. The private SSH key will be the one with the “.key” extension at the end.
109. Once you have opened your private SSH Key in a text editor, copy the contents of the entire file as shown below:

```
-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEAMrQ7JALGKLMq13LsF4T/LvKaR1U0F
...snip...
-----END RSA PRIVATE KEY-----
```

110. After copying the contents, go back to your terminal window from which you will be able to SSH into the newly created Compute Instance. For this guide, we have chosen an Oracle Linux machine for SSHing into the “MySQL-Compute” Compute Instance.

```
[opc@oraclelinux8 ~]$ $ cd
[opc@oraclelinux8 ~]$ $ mkdir .ssh
[opc@oraclelinux8 ~]$ $ cd .ssh
```

111. On your terminal, inside your home directory, create a new “.ssh” directory. Once the directory is created, go inside that directory. If you already have a .ssh directory, just go inside that directory.

```
oraclelinux8 $ cd
oraclelinux8 $ mkdir .ssh
oraclelinux8 $ cd .ssh
```

```
[opc@oraclelinux8 ~]$ cd
[opc@oraclelinux8 ~]$ $ mkdir .ssh
[opc@oraclelinux8 ~]$ $ cd .ssh
[opc@oraclelinux8 .ssh]$ $ ...
```
112. Once inside the "ssh" directory, create a new file called `id_rsa` and paste the contents of the private SSH key into the newly created `id_rsa` file. The guide uses the "nano" text editor, use a text editor of your own choice.

```
oraclelinux8 $ nano id_rsa
```

113. 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`
114. After you have saved the private SSH Key on your terminal window, 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:

```
oraclelinux8 $ ls
oraclelinux8 $ pwd

[opc@oraclelinux8 .ssh]$ ls
id_rsa
[opc@oraclelinux8 .ssh]$ pwd
/home/opc/.ssh
```

Note: by looking at the above image, the id_rsa location for this guide will hence be /home/opc/.ssh/id_rsa

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

```
oraclelinux8 $ chmod 400 id_rsa

[opc@oraclelinux8 .ssh]$ chmod 400 id_rsa
```

116. Now try to connect to the Compute Instance we created earlier by executing the following from your terminal window where you have the SSH keys

```
ssh -i <path/to/you-private-ssh-key> opc@<compute-public-ip-address>

[opc@oraclelinux8 ~]$ ssh -i .ssh/id_rsa opc@150.136.224.43
Activate the web console with: systemctl enable --now cockpit.socket

[opc@mysql-compute ~]$
```
Section Q: Connect to your Compute Instance using MySQL Shell and execute the util.loadDump() utility

117. This guide showcased how you can import your data from an Object Storage bucket into MySQL HeatWave on OCI using PAR URL, which will import the data while the MySQL HeatWave System is creating. But if you want to load your data after the MySQL HeatWave System is created, the alternate method of loading the data into your MySQL HeatWave database is to use MySQL Shell loadDump() utility.

To use MySQL Shell dump utility, we will need to first drop the “world” database we imported using the PAR URL in the earlier step. Login to your Compute instance with MySQL Shell installed to connect to MySQL HeatWave

```
ssh -i <path/to/you-private-ssh-key> opc@<compute-public-ip-address>
```

then,

```
mysql-compute $ mysqlsh <username>@<private-mysql-ip>
```

or

```
mysql-compute $ mysqlsh -u <username> -h <private-mysql-ip> -p <portnumber> -p
```

118. Once you are connected to MySQL HeatWave, perform a DROP DATABASE and exit out of MySQL Shell

```
MySQL JS> \sql
MySQL SQL> DROP DATABASE world;
MySQL SQL> \q
```
119. After exiting MySQL HeatWave and MySQL Shell, from the command line of your OCI Compute Instance verify that you are in your home directory. Create a directory named `.oci`. Next, go into the `.oci` directory and create a file called `config`. Paste the contents of the “Configuration File Preview” we copied in Step 36 into the newly created `config` file. The commands used to achieve this step for the guide, are listed below:

```
mysql-compute $ mkdir ~/.oci
mysql-compute $ cd .oci
mysql-compute $ nano config
```

120. After the “config” file opens, paste the “Configuration File Preview” content from Step 36

```
[DEFAULT]
user=ocid1.user.oc1..aaa
tenancy=ocid1.tenancy.oc1..aaa
region=us-ashburn-1
key_file=<path to your private keyfile>  # TODO
```

121. Once you have pasted the “Configuration File Preview” snippet into the “config” file, you will need to adjust the parameter where it says “key_file” with the file path to your own OCI Private API Key. Look at an example below:

```
[DEFAULT]
user=ocid1.user.oc1..aaa
tenancy=ocid1.tenancy.oc1..aaa
region=us-ashburn-1
key_file=/home/opc/privakey.pem
```

- Note: If you need help uploading your Private API Key onto your Compute Instance, the steps are as follows
122. Open your Private API Key that we downloaded in Step 35 in a text editor of your choice. The Private API Key will be the file without the word “public” in the file name.

123. Once you have opened your Private API Key in a text editor, copy the contents of the entire file like shown below:

124. After copying the contents, go back to your Compute Instance on OCI where you have created the “config” file and have MySQL Shell installed. Create a new file there called privapikey.pem, for example. This guide used the “nano” text editor to create the privapikey.pem file on the Compute Instance. Choose a text editor of your own choice.

```
mysql-compute $ cd
mysql-compute $ nano privapikey.pem
```

```
[opc@mysql-compute ~]$ nano privapikey.pem
```
125. Once the `privapikey.pem` file opens, paste the contents of the OCI Private API Key that we copied in Step 123, into this newly created `privapikey.pem` file. Save and close the file afterwards. If you are using nano,

- to paste the copied content: `command + V`
- to save the file: `control + O`
- to exit the file: `control + X`

126. After you have saved the Private API Key on your OCI Compute Instance, grab the file path of the `privapikey.pem` and adjust the “`key_file`” parameter in the `.oci/config` file. To get the file path of your current working directory where you have the `privapikey.pem`, execute:

```
mysql-compute $ ls
mysql-compute $ pwd
```

```
[opc@mysql-compute ~]$ ls
privapikey.pem
[opc@mysql-compute ~]$ pwd
/home/opc
[opc@mysql-compute ~]$ 
```

- Note: by looking at the above image, the `privapikey.pem` location for this guide will hence be `/home/opc/privapikey.pem`. Adjust your config file accordingly.
After completing the above steps, you are now ready to load the dump data from the Object Storage Bucket into MySQL HeatWave using MySQL Shell. Log back into your MySQL HeatWave system using the Compute Instance where we have created the “config” file and have MySQL Shell installed.

```
mysql-compute $ mysqlsh <username>@<private-mysql-ip>
```

or

```
mysql-compute $ mysqlsh -u <username> -h <private-mysql-ip> -P <portnumber> -p
```
It is now time to load our sample database “world”, which we dumped from our Amazon Aurora MySQL to OCI Object Storage. Inside MySQL Shell, make sure you are in the JavaScript mode of MySQL Shell by executing `\js` and then, execute the `loaddump` utility to import the dumped data from Oracle Cloud Object Storage bucket into MySQL HeatWave.

```sql
MySQL SQL> \js
```

```sql
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: [https://dev.mysql.com/doc/mysql-shell/8.0/en/mysql-shell-utilities-load-dump.html](https://dev.mysql.com/doc/mysql-shell/8.0/en/mysql-shell-utilities-load-dump.html)
• Change the prefix, `osBucketName` and `osNamespace` to match with what you have.
```
129. Once you have executed the command in Step 128 and did not see any errors or warnings, execute the same Step 128 command. Although, this time change the dryRun option to false.


Loading DDL, Data and Users from OCI ObjectStorage bucket=Migration-Bucket, prefix='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 dumps from different major MySQL versions is not fully supported and may not work. The 'ignoreVersion' option is enabled, so loading anyway.
Fetching dump data from remote location...
Listing files - done
Scanning metadata - done
Checking for pre-existing objects... Executing common preambles 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, REPLICATION SLAVE, REPLICATION CLIENT, CREATE VIEW, SHOW VIEW, CREATE ROUTINE, ALTER ROUTINE, CREATE USER, EVENT, TRIGGER ON ** TO 'rdsadmin'@'localhost' WITH GRANT OPTION; -> GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, PROCESS, REFERENCES, INDEX, ALTER, SHOW DATABASES, CREATE TEMPORARY TABLES, LOCK TABLES, EXECUTE, REPLICATION SLAVE, REPLICATION CLIENT, CREATE VIEW, SHOW VIEW, 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
188KB (194.62 KB / 194.62 KB), 0.00 B/s, 3 / 3 tables done
Recreating indexes - done
3 chunks (5.38K rows, 194.62 KB) for 3 tables in 1 schemas were loaded in 1 sec (avg throughput 194.62 KB/s)
0 warnings were reported during the load.

MySQL 33860+ ssi 3S -

- Note: once the load process is complete, MySQL Shell will display a summary of the dump process like the one shown in the above image.
130. 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:

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

![Database output image](image-url)
Section R: Performing the util.dumpInstance() and util.loadDump() utility to and from a local filesystem

131. For relatively small databases, you can create the dump files on your local system. Although, you need to transfer them to the OCI Compute instance using the copy utility of your choice, depending on the operating system you chose for your Compute 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 config file is not required here)

132. 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 EC2 instance that we had provisioned earlier.

133. Connect to your Amazon Aurora MySQL instance using MySQL Shell.

```
ec2-user $ mysqlsh <username>@<amazon-aurora-endpoint>
```

or

```
ec2-user $ mysqlsh -u <username> -h <amazon-aurora-endpoint> -P <portnumber> -p
```

```
Please provide the password for 'root@database-1.cluster.us-east-1.rds.amazonaws.com': *******
```

```
Save password for 'root@database-1.cluster.us-east-1.rds.amazonaws.com'? [Y]/n/O/[N]o/[N]e[v]er (default No):
```

```
MySQL Shell 8.0.31
```

Copyright (c) 2016, 2022, Oracle and/or its affiliates.
Oracle is a registered trademark of Oracle Corporation and/or its affiliates.
Other names may be trademarks of their respective owners.

Type '\help' or '\?' for help; '\quit' to exit.
Creating a session to 'root@database-1.cluster.us-east-1.rds.amazonaws.com'
Fetching schema names for auto-completion... Press ^C to stop.
Your MySQL connection id is 22
Server version: 5.7.12 MySQL Community Server (GPL)
No default schema selected; type '\use <schema>' to set one.
```
MySQL database-1.cluster-us-east-1.rds.amazonaws JS >
```
134. Make sure you are in JavaScript mode by typing `\js` and execute the `dumpInstance` utility to export the dump data into your local filesystem

```javascript
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"})
```

**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-cbsp6cxyrq6y.us-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 (INVOKING COMPREHEND, INVOKING LAMBDA, INVOKING 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 proceeding.

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

**Note:**
- `/home/ec2-user/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
- Use the `consistent: false` option, if and only if, your `dump` utility produces “Errors” regarding “table locks” (MySQLSH 52002: See Steps 49/50 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 Database Service/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
135. Once you have executed the command in Step 134 and did not see any additional errors or warnings, execute the same Step 134 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-1.cluster-0.us-east-1.rds world JS > util.dumpInstance("/home/ec2-user/sampledump", {"ocimds": "true", "compatibility": ["strip_restricted_grants", "strip_definers"], 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 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-0.us-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

MySQL database-1.cluster-0.us-east-1.rds world JS > util.dumpInstance("/home/ec2-user/sampledump", {"ocimds": "true", "compatibility": ["strip_restricted_grants", "strip_definers"], 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 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-0.us-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: once the dump process is complete, MySQL Shell will display a summary of the dump process like the one shown in the image above.

136. 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 ~]$ ls
[ec2-user@ip ~]$ cd sampledump
[ec2-user@ip sampledump]$ ls
@.done.json  world@city.json  world@countrylanguage.json
testapikey.pem  world@city.sql  world@countrylanguage.sql
world@city@@0.tsv.zst  world@country@@0.tsv.zst  world@country@@0.tsv.zst.idx
world@countrylanguage@@0.tsv.zst  world@countrylanguage@@0.tsv.zst.idx
world@country@@0.tsv.zst.idx
[ec2-user@ip sampledump]$ ls
```

85  Migration Guide: Amazon Aurora to MySQL HeatWave on Oracle Cloud Infrastructure (OCI)
Copyright © 2023, Oracle and/or its affiliates. Public
137. Now, transfer the `sampledump` directory to the OCI Compute instance using the copy utility of your choice, depending on the operating system you chose for your Compute instance. One way to do this is to use the `scp` command.

138. After you have copied over your `sampledump` directory to the Oracle Cloud Infrastructure Compute instance, login to that Compute instance and retrieve the path to the `sampledump` directory.

```
ssh -i <path/to/you-private-ssh-key> opc@<compute-public-ip-address>
```

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

139. Login to that Compute 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> opc@<compute-public-ip-address>
then,
mysql-compute $ mysqlsh <username>@<private-mysql-ip>
or
mysql-compute $ mysqlsh -u <username> -h <private-mysql-ip> -P <portnumber> -p
```
140. It is now time to load our sample database "world", that was dumped from our Amazon Aurora MySQL to the local filesystem in EC2, which we later transferred to the OCI Compute 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 Oracle Cloud Compute instance into MySQL HeatWave.

```sql
MySQL SQL> \js
```

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 loads the data. For more information, refer to: [https://dev.mysql.com/doc/mysql-shell/8.0/en/mysql-shell-utilities-load-dump.html](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.
141. Once you have executed the command in Step 140 and did not see any errors or warnings, execute the same Step 140 command. Although, this time change the dryRun option to false.

```sql
```

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

- 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.
142. 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;
```

```
Switching to SQL mode... Commands end with ;
Patching global names for auto-completion... Press ^C to stop.
MySQL
```

```
<table>
<thead>
<tr>
<th>Database</th>
</tr>
</thead>
<tbody>
<tr>
<td>information_schema</td>
</tr>
<tr>
<td>mysql</td>
</tr>
<tr>
<td>performance_schema</td>
</tr>
<tr>
<td>sys</td>
</tr>
<tr>
<td>world</td>
</tr>
</tbody>
</table>

```

```
MySQL SQL> SHOW SCHEMAS;
```

```
5 rows in set (0.0010 sec)
```

```
MySQL SQL> SHOW TABLES IN world;
```

```
<table>
<thead>
<tr>
<th>Tables_in_world</th>
</tr>
</thead>
<tbody>
<tr>
<td>city</td>
</tr>
<tr>
<td>country</td>
</tr>
<tr>
<td>countrylanguage</td>
</tr>
</tbody>
</table>
```

3 rows in set (0.0015 sec)