Migration Guide: MariaDB to MySQL HeatWave on Oracle Cloud Infrastructure (OCI)
Before you start:

• You must have an account on Oracle Cloud Infrastructure (OCI).
• Some OCI knowledge is preferred.
• This migration document only covers how to migrate your database from MariaDB to MySQL HeatWave (HW) on OCI. Before performing the migration, you should have considered downtime (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), application compatibility, current database metrics (CPU, storage size, RAM, max number of concurrent users, backups, binary logs expiration, number of replicas if any, etc.), desired database metrics, networking, security, user testing, etc.
• The migration method shown in this guide works for MariaDB 5.5 and MariaDB 10.3 to 10.11. It could work with more recent versions but has not been tested.
• When following the guide, you should always execute the commands/steps shown as an admin/root user wherever applicable.
  o On OCI you must have the ability to create and manage resources.
  o For your MariaDB instance, use an admin/root user.
• You do not need to make any configuration changes to your on-premises MariaDB for this migration unless it’s explicitly stipulated.
• If you have replication configured in your current MariaDB environment, you can perform the migration steps shown in this guide from either your source or replica instance.
• The Overview section of this migration guide contains all the steps that are needed to complete the database migration from on-premises MariaDB to MySQL HW on OCI.
• In the Walkthrough section of this guide, we will apply the information provided in the Overview section and give you a simple step-by-step guide. In this step-by-step guide, we will have an on-premises MariaDB instance with some sample data pre-loaded and will migrate it over to MySQL HW on OCI. This will help you follow and better visualize the process/information provided in the Overview section.
• You can use the Walkthrough section’s step-by-step guide as a reference for your migration from on-premises MariaDB to MySQL HW. When following the guide, make changes along the way to your on-premises and OCI environment accordingly or as required. Since each user following the step-by-step guide will have their environments configured differently, we cannot provide an ideal example that works for everyone.
Overview:

Following are the required steps to migrate data from MariaDB to MySQL HW on OCI:

I) Have an Oracle Cloud Infrastructure (OCI) account.

OCI Sign in/Sign up page: https://cloud.oracle.com

II) Set up a VPN connection from OCI to on-premises.

[A VPN connection will allow you to bridge your on-premises network with the OCI VCN. The VPN connection will allow your MariaDB to connect to MySQL HW on OCI and it also ensures that your data in transit is encrypted while it is being migrated.]


III) On OCI, create a MySQL HW instance.

[You can create either a Standalone or High Availability MySQL HW instance. Both options are fully-managed.]


IV) Install MySQL Shell 8.2.1 (or above) on an on-premises instance that can connect to your MariaDB.

[MySQL Shell will be used to copy DDL and data from MariaDB to MySQL HW on OCI. You must download MySQL Shell 8.2.1 or above.]

Download MySQL Shell: https://dev.mysql.com/downloads/shell/

V) Connect to the MariaDB using MySQL Shell. And perform some compatibility checks.

MariaDB is no longer a drop-in replacement for MySQL and some checks are mandatory before the migration.

VI) Afterwards, execute the MySQL Shell `util.copyInstance()` utility to export all schemas (including users, indexes, routines, triggers) from MariaDB to the MySQL HW on OCI.

[The dump created by MySQL Shell's instance copy utility comprises DDL files specifying the schema structure, and tab-separated .tsv files containing the data.]

VI) (Optional) On OCI, use the Cloud Shell to verify whether the data was migrated successfully from MariaDB to MySQL HW on OCI.

[Cloud Shell is a web browser-based terminal accessible from the Oracle Cloud Console.]
OCI Cloud Shell: https://docs.oracle.com/en-us/iaas/Content/API/Concepts/cloudshellintro.htm

VII) (Optional) On OCI, if the HeatWave option was enabled during MySQL HW DB creation, add the HW Cluster and load data from MySQL InnoDB storage into the HW Cluster using automation.

[Attaching the HeatWave in-memory Cluster combines transactions, analytics, and machine learning services into one MySQL Database.]
Add a HeatWave Cluster: https://docs.oracle.com/en-us/iaas/mysql-database/doc/adding-heatwave-
Walkthrough:

I) Have an Oracle Cloud Infrastructure (OCI) account.
OCI Sign in/Sign up page: https://cloud.oracle.com

II) Set up a VPN connection from OCI to on-premises.
Note: this guide uses OpenVPN Access Server which lets you connect your MariaDB with MySQL HW on OCI. You cannot use OpenVPN Access Server to connect entire sites or networks to an Oracle VCN; in that scenario, it is recommended to use Site-to-site VPN or FastConnect.

1. Below is the MariaDB instance version and the sample database (“world”) that will be migrated for this guide. The sample world database consists of 3 tables.

```
SQL> select @@version;
+-----------------------+
| @@version             |
| 11.1.2-MariaDB        |
+-----------------------+
1 row in set (0.0002 sec)

MySQL 127.0.0.1:3306 2023-10-29 21:57:01

SQL> show schemas;
+-----------------------+
| Database               |
| information_schema    |
| mysql                 |
| performance_schema    |
| sys                   |
| test                  |
| world                 |
+-----------------------+
6 rows in set (0.0010 sec)

MySQL 127.0.0.1:3306 2023-10-29 21:57:11

SQL> show tables in world;
+-----------------------+
| Tables_in_world       |
| city                  |
| country               |
| countrylanguage      |
+-----------------------+
3 rows in set (0.0003 sec)
```

2. Log in to OCI and create a VCN. Open the navigation menu, click Networking, and click Virtual cloud networks.
3. Ensure you are in your desired compartment - we have chosen the root compartment. Click **Start VCN Wizard**.

4. Select **Create VCN with Internet Connectivity** and click **Start VCN Wizard**.

5. Enter a **VCN name** and **configure your VCN’s IPv4 CIDR block** - including the public and the private **subnet**. The guide uses the default values for all. Make sure that the OCI VCN IPv4 CIDR block does not overlap with your on-premises network.
6. Click **Next** after the configuration for your VCN is completed.

7. On the Review and create page, validate the information for your VCN and click **Create**.
8. Click **View VCN** after your VCN creation has been completed.

9. From the OCI navigation menu, click **Networking** and click **Site-to-Site VPN**.

10. Click **marketplace solution** on the right side of the page.

11. On the OpenVPN Access Server page, from the dropdown, select the compartment where your VCN resides. Check the **terms of use and conditions** checkbox and click **Launch Stack**.
12. On the **Stack information** page of **Create stack**, leave everything as-is and click **Next**.

13. On the **Configure variables** page, under **Compute Shape** select either **VM.Standard2.2** or **VM.StandardE2.2**. For **Application Configuration**, create an admin **username and password**. Make a note of the admin credentials.
14. For **Network Configuration**, under **Network Strategy**, select **Use Existing VCN** and select the **VCN that we created earlier** from the **Existing Network** dropdown. For the **Existing Subnet**, select the **Public Subnet** of your VCN. Under **Additional Configuration**, ensure the compartment is where your VCN resides. Click **Next**.

15. On the Review page of Create stack, click **Create**.

16. Finishing the previous step will provision a compute instance for the VPN. From the OCI navigation menu, click **Compute** and click **Instances**. It may take a few minutes for your compute host to be ready.
17. Copy and save the Public and the Private IP of the openvpn_access_server.

18. Open a web browser and enter the following in the search bar.

   https://<openvpn-access-server-public-ip>/admin/

   ![Image of web browser with URL](image)

   Note: in the web browser when prompted, click Advanced and click Proceed to <openvpn-access-server-public-ip> (unsafe) or Accept the Risk and Continue.

19. Enter the admin credentials that you configured earlier in step 13 to log in.
20. After logging in, from the left-hand side menu, select **Configuration** and click **VPN Settings**.

21. On the VPN Settings page, under **Dynamic IP Address Network** - input **172.27.233.0** for Network Address and **24** for # of Netmask bits. Under **Static IP Address Network**, input **172.27.232.0** for Network Address and **24** for # of Netmask bits. Leave the **Group Default IP Address Network** field as-is.
22. While on the VPN Settings page, scroll down to **Routing**. Select **Yes, using Routing**, and specify your OCI VCN public and private subnets IPv4 CIDR blocks next to **Specify the private subnets to which all clients should be given access (one per line)**.

23. Scroll down and click **Save Settings**.

24. From the left-hand OpenVPN Access Server menu, select **USER MANAGEMENT** and click **User Permissions**.
25. Enter a username in the **New Username** field and click the **More Settings** icon in the adjacent column.

26. Enter a **Password** for the user you created in the previous step. For **Select IP Addressing**, click **Use Static** and specify the IP address to assign to the new user in the **VPN Static IP Address** field. This IP address must be in the range defined in the **Static IP Address Network** field of the VPN Configuration, see step 21. For this guide, we have chosen **172.27.232.25**. Select **Use Routing** for **Select addressing method** and specify your OCI VCN public and private subnets IPv4 CIDR blocks in the **Allow Access To these Networks** field. For **Allow Access From**, select all server-side private subnets. Click **Save Settings**.
27. After saving the completed previous step, click **Update Running Server**.

28. Log out and log in using the new user credentials that you created in step 26. Remove the `/admin` from the URL when logging in if you did not assign the new user to be an admin.

   `https://<openvpn-access-server-public-ip>/`
29. Once logged in as the new user, click **Yourself (user-locked profile)** to download `client.ovpn` profile.

30. Click the appropriate platform icon depending on the Operating System (OS) you are running to download the OpenVPN client. For this guide, we are using macOS. After downloading the client, install it. For more information see, [Installation guide for macOS], [Installation guide for Windows], and [Connecting to Access Server with Linux].

31. After installing the OpenVPN client on your OS, import the `client.ovpn` profile. For more information see, [Import a Profile].

32. Once the profile has been imported, **start the OpenVPN Client**. It is now time to configure the OCI VCN to enable communications from the OpenVPN Access Server.

33. Login to [OCI] and open the navigation menu. Select **Networking** and click **Virtual Cloud Networks**.

34. Save the VCN IPv4 CIDR Block for later use and click on the **name of your VCN**.
35. On the Virtual Cloud Network Details page, click Route Tables and click route table for private subnet-<vcn-name>.

36. Click Add Route Rules.
37. For **Target Type** select **Private IP**. Make sure **CIDR Block** is selected under **Destination Type**. For **Destination CIDR Block**, input the **Static IP Address Network CIDR Block** from step 21 - in our case, it is 172.27.232.0/24. Under **Target Selection**, enter the **Private IP** of the OpenVPN access server from step 17. Click **Add Route Rules**.

38. Go back to the Virtual Cloud Network Details page of your VCN and click **Security Lists**.
39. Click on the **security list for private subnet-<vcn-name>**.

![Security List](image)

40. Click **Add Ingress Rules**.

![Add Ingress Rules](image)

41. For **Source CIDR**, input the **Static IP Address Network CIDR Block** from step 21 - in our case, it is `172.27.232.0/24`. For **Destination Port Range**, specify `3306,33060`. Leave everything as-is and click **Add Ingress Rules**.
42. Stay on the same security list for private subnet-<vcn-name> page and click **Add Ingress Rules** again.

43. For **Source CIDR**, enter the IPv4 CIDR Block of your OCI VCN from step 34. For **Destination Port Range**, specify **3306,33060**. Leave everything as-is and click **Add Ingress Rules**.
44. Go back to the **Virtual Cloud Network Details** page of your VCN and click **Security Lists**.

45. Click on **Default Security List for <vcn-name>**.
46. Click **Add Ingress Rules**.

![Add Ingress Rules](image)

47. For **Source CIDR**, enter the IPv4 CIDR Block of your OCI VCN from step 34. For **Destination Port Range**, specify **3306,33060**. Leave everything as-is and click **Add Ingress Rules**.

![Add Ingress Rules](image)

48. The VPN connection from on-premises to OCI is now set up. Make sure the OpenVPN client is started/running. We are now ready to perform the migration.
III) On OCI, create a MySQL HW instance.

49. From the OCI Console, click on the navigation menu, click **Databases**, and click **MySQL HeatWave**.

50. Pick **Production** or **Development or testing** and enter a **MySQL DB system name**.
51. Select **Standalone** or **High Availability**. Turn **ON** the button for MySQL HeatWave - if you want to run OLTP, OLAP, and ML workloads. Afterwards, create your **Administrator credentials** that will be used to manage the MySQL HeatWave database.

52. For **Configuring Networking** - choose the earlier created VCN and make sure the **Private Subnet** is selected under **Subnet in <compartment-name>**. For **Configure Placement** leave it as-is.
53. **Configure hardware** (OCPU and Memory) for MySQL by choosing an appropriate DB Shape. For this guide, we will use the default HeatWave shape. For the **Data Storage Size** be sure to make the size large enough for future growth.

54. **Configure a backup plan** according to what suits your needs. Lastly, scroll down until you see **Show advanced options**. Click on it to expand.
55. From the advanced options screen, go to the **Configuration** tab. If you have a custom configuration that you would like to apply to your MySQL HW instance - you can do so by clicking **Select configuration**. Custom configurations allow you to tweak MySQL variables (i.e., max connections, binary log expire seconds, etc.) rather than using the default values. You must create a custom configuration in advance before applying. For more information regarding custom configurations, see [Configuration of a DB System](#). For this guide, we have chosen the default configuration.
56. For **MySQL version**, choose either **Innovation** or **Bug fix**. With the new MySQL versioning model, you have the flexibility to select an innovation or a bug fix release. Both releases are production-grade quality. MySQL innovation releases allow you to access the latest features and improvements. Innovation releases are ideal for fast-paced development environments with high levels of automated tests and modern continuous integration techniques for faster upgrade cycles. MySQL bug fix releases (aka long-term support releases) allow you to reduce the risks associated with changes in the database software behavior, as these releases only contain necessary fixes (bugfix and security patches). For more information regarding MySQL innovation and bug fix releases, see *Introducing MySQL Innovation and Bug fix versions*. For this guide, we have chosen **8.0.34 - Bug fix**.

57. Click **Create** to finish the MySQL HW DB system creation process.
58. Your MySQL HW DB system will start **CREATING**.

![MySQL HW System Creating](image)

59. Within a few minutes, MySQL HW DB system will change its state from **CREATING** to **ACTIVE** once the instance is ready.

![MySQL HW System Active](image)

60. On the same DB system details page, click **Connections** to grab the **private IP address** for MySQL HW. Save the private IP Address for later use.

![MySQL HW System Connections](image)

Note: you can navigate to the **DB System Details** page by going to the Navigation menu in OCI. Click **Databases** and click **MySQL HeatWave**. Click on the name of your MySQL DB System to open the **DB System Details** page.
IV) Install MySQL Shell 8.2.1 (or above) on an on-premises instance that can connect to your MariaDB.

1. MySQL Community Downloads

   MySQL Shell

   [Image - General Availability (GA) Releases]

   MySQL Shell 8.2.1 Innovation

   Select Version: 8.2.1 Innovation
   Select Operating System: Red Hat Enterprise Linux / Oracle Linux
   Select OS Version: Red Hat Enterprise Linux 8 / Oracle Linux 8 (x86, 64-bit)

   RPM Package
   (mysql-shell-8.2.1-1.el8.x86_64.rpm)
   8.2.1
   30.1M
   [Download]

   RPM Package, Debug Information
   (mysql-shell-debuginfo-8.2.1-1.el8.x86_64.rpm)
   8.2.1
   496.0M
   [Download]

   We suggest that you use the MD5 checksums and GnuPG signatures to verify the integrity of the packages you download.

61. Have an on-premises instance that can connect to your MariaDB. Go to the below website and download MySQL Shell 8.2.1 on your on-premises instance. For this guide, we have deployed MySQL Shell on a Linux instance. From the MySQL Shell download page, ensure 8.2.x Innovation is selected under Select Version. MySQL Shell 8.2 is fully compatible with MySQL 8.2, 8.1, 8.0, and 5.7. For Operating System and OS Version - pick the appropriate option depending on the OS and the OS Version that you are running. Click Download.

   [Website - https://dev.mysql.com/downloads/shell/]

   Note: for this guide, we will show you how to install MySQL Shell on a Linux environment. For other environments, see Installing MySQL Shell on Windows, Installing MySQL Shell on Linux, and Installing MySQL Shell on macOS.
62. Right-click on No thanks, just start my download and click Copy link address.

63. Go back to the on-premises instance that can connect to your on-premises MariaDB and execute the below command to install MySQL Shell:

```bash
$ sudo yum install -y <MySQL-Shell-Download-Link>
```

Replace the link with what you have.

```bash
$ sudo yum install https://dev.mysql.com/get/Downloads/MySQL-Shell/mysql-shell-8.2.1-1.el8.x86_64.rpm
```

```
[opc@linux-8 ~]# sudo yum install https://dev.mysql.com/get/Downloads/MySQL-Shell/mysql-shell-8.2.1-1.el8.x86_64.rpm
Last metadata expiration check: 1:10:27 ago on Tue 21 Nov 2023 06:09:14 PM GMT.
mysql-shell-8.2.1-1.el8.x86_64.rpm | 88 MB/s | 30 MB 00:00
```

Transaction Summary

```
Install 7 Packages

Total size: 43 M
Total download size: 12 M
Installed size: 261 M
```
64. You can now verify if MySQL Shell has successfully installed on your on-premises instance by executing the below command:

$ mysqlsh --version

[opc@linux-8 ~]$ mysql --version
mysql Ver 8.2.1 for Linux on x86_64 - for MySQL 8.2.0 (MySQL Community Server [GPL])

65. To login to your MariaDB using MySQL Shell, use the below commands:

$ mysqlsh mysql://<user>@<hostname>[:<port-number>]

OR

$ mysqlsh -u <user> -p -h <hostname> [-P <port-number>]

Note: you can interact with MySQL Shell using JavaScript, Python, or SQL mode. The default is JavaScript. To switch between the different modes, execute /js for JavaScript, /py for Python, and /sql for SQL mode inside MySQL Shell. To exit out of MySQL Shell, execute /q.

V) Connect to the MariaDB using MySQL Shell. And perform some compatibility checks.

Storage Engines

66. MariaDB Community Edition contains various storage engines in alpha or beta stages. These engines are not included in MariaDB Enterprise Edition and are also not supported in MySQL. In MySQL, InnoDB is the main engine used for transaction processing. Before migration, you will need to convert data to InnoDB.

List all the Storage Engines used in your MariaDB instance:

MySQL> SELECT COUNT(*) as '# TABLES',
CONCAT(ROUND(sum(data_length) / ( 1024 * 1024 * 1024 ), 2), 'G') DATA,
CONCAT(ROUND(sum(index_length) / ( 1024 * 1024 * 1024 ), 2), 'G') INDEXES,
CONCAT(cast(sum(ROUND(( data_length + index_length ) / ( 1024 * 1024 * 1024 ), 2))
as decimal(5,3), 'G') 'TOTAL SIZE',
ENGINE FROM information_schema.TABLES
WHERE TABLE_SCHEMA
NOT IN ('mysql', 'information_schema', 'performance_schema', 'sys')
GROUP BY engine;
In this example, it’s fine as only InnoDB tables are present. If there was another line with another engine, those tables would need to be converted to InnoDB.

**Functions**

67. MariaDB also differs from MySQL by some different functions. For example, in MariaDB you have JSON_DETAILED which is called JSON_PRETTY in MySQL.

This is not a blocking factor unless those functions are present in the default value of a column. If the application uses some of these functions, it may be necessary to make some modifications to use the appropriate one in MySQL.

**Data Types**

68. MySQL and MariaDB have some different data types.

For example, MariaDB supports INET6 as a data type and in MySQL, IPv6 values are stored into VARBINARY(16).

But on the other hand, MySQL supports JSON data type that in MariaDB are stored as LONGTEXT.

To list all data types used in your database, you can execute the following query:

```sql
SELECT DATA_TYPE, count(*) TOT FROM information_schema.COLUMNS WHERE TABLE_SCHEMA NOT IN ('mysql', 'sys', 'information_schema', 'performance_schema') GROUP BY 1;
```

If you find some data types that are not present in MySQL, you need to convert them. See: https://dev.mysql.com/doc/refman/8.0/en/data-types.html
V) Connect to MariaDB using MySQL Shell. Afterwards, execute the MySQL Shell `util.copyInstance()` utility to export all schemas (including users, indexes, routines, triggers) from the MariaDB instance to MySQL HW on OCI.

69. Before connecting to MariaDB using MySQL Shell and proceeding with the below steps, it is highly recommended that you use a command like `screen` or `tmux`. These commands will allow you to reconnect to a dropped session in case your connection drops in the middle of performing the MySQL Shell export using `util.copyInstance()`. For small databases, the screen or tmux may not be necessary. For this guide, we will use tmux. To learn more about tmux, see [A beginner’s guide to tmux](#). Below are the basics of using the tmux command:

- Install tmux on Linux: 
  ```
  $ sudo yum install tmux
  ```
- Start a new tmux session, from your terminal execute: 
  ```
  $ tmux
  ```
- List all the active tmux sessions: 
  ```
  $ tmux ls
  ```
- Detach from a tmux session and leave it running in the background: 
  ```
  $ Ctrl+B d
  ```
- Attach a tmux session running in the background: 
  ```
  $ tmux attach
  ```
- End a tmux session: 
  ```
  $ Ctrl+B &
  ```

70. Start a tmux session and connect to your on-premises MariaDB using MySQL Shell.

```bash
$ tmux
$ mysqlsh mysql://<user>@<hostname>[:<port-number>]
OR-
$ mysqlsh -u <user> -p -h <hostname> [-P <port-number>]
```

```java
[opc@linux-8 ~]$ tmux
Optionally, please provide the password for 'root@localhost': *********
MySQL Shell 8.2.0

Copyright (c) 2016, 2023, 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 Classic session to 'root@localhost'
Fetching schema names for auto-completion... Press ^C to stop.
Your MySQL connection id is 23
Server Version: 11.1.2-MariaDB MariaDB Server
No default schema selected; type 'use <schema>' to set one.
MySQL localhost:3306 > |
```
71. Change to the JavaScript mode of MySQL Shell if needed, and run the `util.copyInstance()` utility to export all MariaDB data into OCI MySQL HW.

MySQL JS> \js
MySQL JS> util.copyInstance('mysql://admin@10.0.1.105', {"compatibility":
["force_innodb", "skip_invalid_accounts", "strip_definers",
"strip_restricted_grants", "strip_tablespaces", "ignore_wildcard_grants",
"strip_invalid_grants", "create_invisible_pks"], users: "false", threads: 4,
ignoreVersion: "true", dryRun:"true"})

Note: replace the username (admin) and IP address (10.0.1.105) with your MySQL HW username and IP address (not the on-premises MariaDB username and IP address).

Note: Migrating MariaDB users is not a supported operation.

MySQL localhost:3306> util.copyInstance('mysql://admin@10.0.1.105', {"compatibility":
["force_innodb", "skip_invalid_accounts", "strip_definers", "strip_restricted_grants", "strip_tablespaces", "ignore_wildcard_grants", "strip_invalid_grants", "create_invisible_pks"], users: "false", threads: 4,
ignoreVersion: "true", dryRun:"true"})

Copying DDL and Data from In-memory FS, source: linux-8:3306, target: stcbj26fjag6cmx:3306.
SRC: dryRun enabled, no locks will be acquired and no files will be created.
NOTE: SRC: Backup lock is not supported in MySQL 5.6 and DDL changes will not be blocked. The dump may fail with an error if schema changes are made while dumping.
SRC: Acquiring global read lock
SRC: Global read lock acquired
Initializing - done
WARNING: SRC: Failed to fetch value of @GLOBAL_GTID_EXECUTED.
SRC: 2 out of 6 schemas will be dumped and within them 3 tables, 0 views.
Gathering information - done
SRC: All transactions have been started
SRC: Global read lock has been released
NOTE: SRC: When migrating to MySQL HeatWave Service, please always use the latest available version of MySQL Shell.
SRC: Checking for compatibility with MySQL HeatWave Service 8.0.35
NOTE: SRC: MySQL Server 5.6 detected, please consider upgrading to 8.0 first.
SRC: Compatibility checks finished.compatibility \ 0 / 5
Validating MySQL HeatWave Service compatibility - done
SRC: Writing global DDL files

[... output truncated]

Writing schema metadata - done
Writing DDL - done
Writing table metadata - done
SRC: Starting data dump
0% (0 rows / -5.27K rows), 0.00 rows/s, 0.00 B/s
GTID: Executing common postamble SQL
9% (0 bytes /?), 0.00 B/s, 3 / 3 tables done
Recreating indexes - done
GTID: No data loaded.
GTID: 0 warnings were reported during the load.

Dump metadata:
Binlog_file: mysql-bin.000001
Binlog_position: 328
Executed_GTID_set: ''

Note:
- **util.copyInstance(connectionData[, options]):** MySQL instance copy utility enables copying of an entire instance to another server. By default, this utility includes all schemas, users, indexes, routines, and triggers. See Copy Utilities.
  - connectionData: Defines the connection details for the destination server you want to copy to.

- **compatibility:** Apply the specified requirements for compatibility with MySQL HeatWave for all tables in the dump output, altering the dump files as necessary.
  - force_innodb: Change CREATE TABLE statements to use the InnoDB storage engine for any tables that do not already use it.
  - skip_invalid_accounts: You cannot export a user that has no password defined. This option skips any such users.
  - strip_definers: Remove the DEFINER clause from views, routines, events, and triggers, so these objects are created with the default definer (the user invoking the schema), and change the SQL SECURITY clause for views and routines to specify INVOKER instead of DEFINER. MySQL HeatWave requires special privileges to create these objects with a definer other than the user loading the schema. If your security model requires that views and routines have more privileges than the account querying or calling them, you must manually modify the schema before loading it.
  - strip_restricted_grants: Certain privileges are restricted in MySQL HeatWave. Privileges such as RELOAD, FILE, SUPER, BINLOG_ADMIN, and SET_USER_ID. You cannot create users granting these privileges. This option strips these privileges from dumped GRANT statements.
  - strip_tablespaces: Tablespaces have some restrictions in MySQL HeatWave. If you need tables created in their default tablespaces, this option strips the TABLESPACE= option from CREATE TABLE statements.
  - ignore_wildcard_grants: If enabled, ignores errors from grants on schemas with wildcards, which are interpreted differently in systems where the partial_revokes system variable is enabled.
  - strip_invalid_grants: If enabled, strips grant statements which would fail when users are copied. Such as grants referring to a specific routine that does not exist.
  - create_invisible_pks: Primary keys are required by High Availability and HeatWave. If you intend to export data for use in a highly available DB system or a HeatWave DB system, add primary keys as they are not defined on the tables. This compatibility flag adds invisible primary keys to each table that requires them.

- **users:** Include (true) or exclude (false) users and their roles and grants in the dump.

- **threads:** (Optional) The number of parallel threads to use to copy chunks of data from the MySQL instance. Each thread has its own connection to the MySQL instance. The default is 4. The copy utilities require twice the number of threads, one thread to copy and one thread to write. If threads is set to N, 2N threads are used.

- **dryRun:** Displays information about the copy with the specified set of options, and about the results of MySQL HeatWave Service compatibility checks, but does not proceed with the copy. Setting this option enables you to list out all of the compatibility issues before starting the copy.
72. Once you have run the command in step 71 and did not see any errors in the output (warnings are okay), run the same step 71 command but this time change the dryRun option to false.

MySQL JS> util.copyInstance('mysql://admin@10.0.1.105', {'compatibility': ['force_innodb', 'skip_invalid_accounts', 'strip_definers', 'strip_restricted_grants', 'strip_tablespaces', 'ignoreWildcard_grants', 'strip_invalid_grants', 'create_invisible_pks'], 'users': 'false', 'threads': 4, 'ignoreVersion': 'true', 'dryRun': 'false'})

Note: replace the username (admin) and IP address (10.0.1.105) with your MySQL HW username and IP address (not the on-premises MariaDB username and IP address).
Note: once the MySQL Shell copy utility finishes, all your data will be copied over from MariaDB to MySQL HW. This completes the migration process. You can end your tmux session.

Note: you will have to manually recreate your MariaDB users in MySQL HW.

VI) (Optional) On OCI, use the Cloud Shell to verify whether the data was migrated successfully from MariaDB to MySQL HW on OCI.

73. Login to OCI, navigate to the top right corner and click on Developer tools right next to your OCI Region.
74. Click **Cloud Shell**.

![Cloud Shell](image)

75. Within a few minutes, you will be connected to the OCI Cloud Shell as below:
76. Click on the **down arrow** next to **Network: Public** and select **Private network definition list**.

77. On the Private network definition list form, select **Create private network definition**.
78. Enter a **private network definition name**. From the **VCN in <compartment-name>** dropdown, select the **VCN associated with MySQL HW**. For **Subnet in <compartment-name>** dropdown, select the **private subnet**. Leave the **Network security groups** as-is and **check the box** where it says **Use as active network**. Click **Create**.

![Create private network definition](image)

79. Click **Close**.

![Private network definition list](image)

80. Within a few minutes, you will be able to access your private subnet (where MySQL HW resides) from the Cloud Shell. You should see the **Network** change from Public to the **private network definition name** that...
you entered in step 76.

81. From the Cloud Shell terminal, login to your MySQL HW instance (by providing the username and private IP of MySQL HW) using MySQL Shell to validate whether the migration was successful:

$ mysqlsh <user>@<hostname>:<port-number>

-OR-

$ mysqlsh -u <user> -p -h <hostname> -P <port-number>

Welcome to Oracle Cloud Shell.
82. Change to the SQL mode of MySQL Shell and run the below commands:

MySQL JS> \sql
MySQL SQL> SHOW SCHEMAS;
MySQL SQL> SHOW TABLES IN <schema-name>;

83. You can run the below query on every table that you have for your on-premises MariaDB and MySQL HW on OCI to ensure that the row count matches on both sides:

MySQL SQL> SELECT COUNT(*) FROM <schema-name>.<table-name>;
84. Here is our row count comparison for MariaDB and MySQL HW:

MariaDB row count:

```
MySQL localhost:33060+ ssl SQL > USE world;
Default schema set to 'world'.
Fetching global names, object names from 'world' for auto-completion... Press ^C to stop.
MySQL localhost:33060+ ssl world SQL > SELECT COUNT(*) FROM city;

+-------+
<table>
<thead>
<tr>
<th>COUNT(*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4079</td>
</tr>
</tbody>
</table>
+---------+
1 row in set (0.0015 sec)
MySQL localhost:33060+ ssl world SQL > SELECT COUNT(*) FROM country;

+-------+
<table>
<thead>
<tr>
<th>COUNT(*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>239</td>
</tr>
</tbody>
</table>
+---------+
1 row in set (0.0008 sec)
MySQL localhost:33060+ ssl world SQL > SELECT COUNT(*) FROM countrylanguage;

+-------+
<table>
<thead>
<tr>
<th>COUNT(*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>984</td>
</tr>
</tbody>
</table>
+---------+
1 row in set (0.0009 sec)
```

MySQL HW row count:

```
MySQL 10.0.1.140:33060+ ssl SQL > USE world;
Default schema set to 'world'.
Fetching global names, object names from 'world' for auto-completion... Press ^C to stop.
MySQL 10.0.1.140:33060+ ssl world SQL > SELECT COUNT(*) FROM city;

+-------+
<table>
<thead>
<tr>
<th>COUNT(*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4079</td>
</tr>
</tbody>
</table>
+---------+
1 row in set (0.0030 sec)
MySQL 10.0.1.140:33060+ ssl world SQL > SELECT COUNT(*) FROM country;

+-------+
<table>
<thead>
<tr>
<th>COUNT(*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>239</td>
</tr>
</tbody>
</table>
+---------+
1 row in set (0.0160 sec)
MySQL 10.0.1.140:33060+ ssl world SQL > SELECT COUNT(*) FROM countrylanguage;

+-------+
<table>
<thead>
<tr>
<th>COUNT(*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>984</td>
</tr>
</tbody>
</table>
+---------+
1 row in set (0.0166 sec)
```

85. After validating, you can have your application/s point to the new MySQL HW instance.
VII) (Optional) On OCI, if the HeatWave option was enabled during MySQL HW DB creation, add the HW Cluster and load data from MySQL InnoDB storage into the HW Cluster using automation.

86. Login to OCI. Click on the navigation menu, go to Databases, and click MySQL HeatWave.

87. Click on the name of your MySQL HW instance to go to the DB System Details page.

88. Click More actions and click Add HeatWave cluster.
89. Click **Estimate node**.

90. Click **Generate estimate**. This step will estimate the number of HeatWave nodes required by selecting the schemas or tables you want to analyze with HeatWave.
91. Within a few minutes, the list of your schemas that are in the MySQL InnoDB storage engine will be listed. **Check the box** next to the schema or table name that you wish to load in HeatWave for query acceleration and to run OLAP and ML workloads - alongside OLTP.

92. After selecting the schemas or tables, scroll down on that page until you see the **Load command**. Copy the **CALL sys.heatwave_load** command and save it. **Click Apply estimated node.**
93. Executing the previous step will change the HeatWave node count depending on the data you have selected to load into HeatWave’s in-memory engine. Click **Add HeatWave cluster** to finish adding the HeatWave cluster creation process.

94. The HeatWave cluster will be ready within a few minutes. You should see the HeatWave state change from **Creating** to **Active**.
95. Connect to your MySQL HeatWave system using MySQL Shell via Cloud Shell.

$ mysqlsh <user>@<hostname>:<port-number>

-OR-

$ mysqlsh -u <user> -p -h <hostname> -P <port-number>

Please provide the password for 'admin@10.0.1.140': ********
Save password for 'admin@10.0.1.140'? [Y]es/[N]o/Ne[ver] (default No): Y
MySQL Shell 8.0.34-commercial

Copyright (c) 2016, 2023, 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 'admin@10.0.1.140'
Fetching schema names for auto-completion... Press ^C to stop.
Your MySQL connection id is 2332 (X protocol)
Server version: 8.0.34-ul1-cloud MySQL Enterprise - Cloud
No default schema selected; type \use <schema> to set one.

MySQL 10.0.1.140:33060+ ssl JS >
96. Switch to the SQL mode of MySQL Shell and execute the Load command that we had copied earlier to load data into HeatWave from the MySQL InnoDB storage.

MySQL SQL> \sql
MySQL SQL> CALL sys.heatwave_load(JSON_ARRAY('world'), NULL);

**Note:** replace the `sys.heatwave_load` command with what you have.

97. You now have a complete MySQL HeatWave cluster.

To learn more about using HeatWave, please visit [our documentation](https://www.oracle.com).