Migration Guide: On-premises MySQL to MySQL HeatWave on Oracle Cloud Infrastructure (OCI)

Copyright © 2023, Oracle and/or its affiliates.
Public
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 on-premises MySQL 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 on-premises MySQL v5.7 and above. This can be a MySQL Community Edition, MySQL Standard Edition, MySQL Enterprise Edition, or Percona Server.
- When following the guide, you should always execute the commands/steps shown as an admin/root user wherever applicable.
  - On OCI you must have the ability to create and manage resources.
  - For your on-premises MySQL instance, use an admin/root user.
- You do not need to make any configuration changes to your on-premises MySQL for this migration.
- If you have MySQL replication configured in your current on-premises 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 MySQL 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 MySQL 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 MySQL 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 on-premises MySQL to MySQL HW on OCI:

I) Have an Oracle Cloud Infrastructure (OCI) account.
OCI Sign in/Sign up page: [https://cloud.oracle.com](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 on-premises MySQL 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 8.1 (or above) on an on-premises instance that can connect to your on-premises MySQL.
[MySQL Shell will be used to copy DDL and data from on-premises MySQL to MySQL HW on OCI. You must download MySQL Shell 8.1 or above.]
Download MySQL Shell: [https://dev.mysql.com/downloads/shell/](https://dev.mysql.com/downloads/shell/)

V) Connect to the on-premises MySQL using MySQL Shell. Afterwards, execute the MySQL Shell `util.copyInstance()` utility to export all schemas (including users, indexes, routines, triggers) from on-premises MySQL 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 on-premises MySQL to MySQL HW on OCI.
[Cloud Shell is a web browser-based terminal accessible from the Oracle Cloud Console.]

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.]
Walkthrough:

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

OCI Sign in/Sign up page: [https://cloud.oracle.com](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 on-premises MySQL with OCI MySQL HW. 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](https://cloud.oracle.com) or [FastConnect](https://cloud.oracle.com).

1. Below is the on-premises MySQL instance version and the sample database ("world") that will be migrated for this guide. The sample world database consists of 3 tables.

   ```
   @VERSION
   8.0.33
   1 row in set (0.0015 sec)
   SHOW SCHEMAS;
   Database
   information_schema
   mysql
   performance_schema
   sys
   world
   5 rows in set (0.0036 sec)
   SHOW TABLES IN world;
   Tables_in_world
   city
   country
   countrylanguage
   3 rows in set (0.0038 sec)
   ```

2. Log in to [OCI](https://cloud.oracle.com) 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/

   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)**.

![VPN Settings](image)

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

![Save Settings](image)

24. From the left-hand OpenVPN Access Server menu, select **USER MANAGEMENT** and click **User Permissions**.

![OpenVPN Access Server](image)
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**.

![Update Running Server](image)

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>**.

40. Click **Add Ingress Rules**.

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**.

![Image of Security Lists](image1.png)

45. Click on **Default Security List for <vcn-name>**.

![Image of Security Lists](image2.png)
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. Click Create DB System.

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** (OCPUs 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 the 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’s 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**.

![MySQL version selection](image)

57. Click **Create** to finish the MySQL HW DB system creation process.

![Create button](image)
58. Your MySQL HW DB system will start **CREATING**.

![MySQL HW DB system details](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 DB system details](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 DB system details](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 8.1 (or above) on an on-premises instance that can connect to your on-premises MySQL.

61. Have an on-premises instance that can connect to your on-premises MySQL. Go to the below website and download MySQL Shell 8.1 on your on-premises instance. For this guide, we have deployed our on-premises MySQL on a Linux instance. From the MySQL Shell download page, ensure 8.1.x Innovation is selected under Select Version. MySQL Shell 8.1 is fully compatible with MySQL 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.

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**.

![MySQL Community Downloads](image)

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

```
$ wget <MySQL-Shell-Download-Link>
```

Replace the link with what you have.

```
$ wget https://dev.mysql.com/get/Downloads/MySQL-Shell/mysql-shell-8.1.1-1.el8.x86_64.rpm
```

```
[opc@linux-8 ~]$ wget https://dev.mysql.com/get/Downloads/MySQL-Shell/mysql-shell-8.1.1-1.el8.x86_64.rpm
--2023-08-15 19:29:50--  https://dev.mysql.com/get/Downloads/MySQL-Shell/mysql-shell-8.1.1-1.el8.x86_64.rpm
Resolving dev.mysql.com (dev.mysql.com)... 23.49.176.249, 2600:1408:c400:188c::2e31, 2600:1408:c400:1881::2e31
Connecting to dev.mysql.com (dev.mysql.com)|23.49.176.249|:443... connected.
HTTP request sent, awaiting response... 302 Moved Temporarily
Location: https://cdn.mysql.com/Downloads/MySQL-Shell/mysql-shell-8.1.1-1.el8.x86_64.rpm [following]
Resolving cdn.mysql.com (cdn.mysql.com)... 23.219.8.226, 2600:1408:c400:1884::1d68, 2600:1408:c400:188d::1d68
Connecting to cdn.mysql.com (cdn.mysql.com)|23.219.8.226|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 28857020 (28M) [application/x-redhat-package-manager]
Saving to: ‘mysql-shell-8.1.1-1.el8.x86_64.rpm’

mysql-shell-8.1.1-1.e 100%[======================================>] 27.52M 72.7MB/s in 0.4s
2023-08-15 19:29:51 (72.7 MB/s) - ‘mysql-shell-8.1.1-1.el8.x86_64.rpm’ saved [28857020/28857020]
64. After downloading the MySQL Shell rpm, install MySQL Shell:

```bash
$ sudo yum localinstall mysql-shell*
```

![Image of yum command output]

### Table: MySQL Shell Installation and Dependencies

<table>
<thead>
<tr>
<th>Package</th>
<th>Arch</th>
<th>Version</th>
<th>Repository</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>mysql-shell</td>
<td>x86_64</td>
<td>8.1.1-1.el8</td>
<td>@commandline</td>
<td>28 M</td>
</tr>
<tr>
<td>python39-libs</td>
<td>x86_64</td>
<td>3.9.16-1.module+el8.8.0+21116+ee8c18cf.1</td>
<td>o18_appstream</td>
<td>8.2 M</td>
</tr>
<tr>
<td>python39-pip-wheel</td>
<td>noarch</td>
<td>20.2.4-7.module+el8.6.0+20625+ee813db2</td>
<td>o18_appstream</td>
<td>1.1 M</td>
</tr>
<tr>
<td>python39-setuptools-wheel</td>
<td>noarch</td>
<td>50.3.2-4.module+el8.5.0+20364+c7fe1181</td>
<td>o18_appstream</td>
<td>497 k</td>
</tr>
<tr>
<td>python39</td>
<td>x86_64</td>
<td>3.9.16-1.module+el8.8.0+21116+ee8c18cf.1</td>
<td>o18_appstream</td>
<td>33 k</td>
</tr>
<tr>
<td>python39-pip</td>
<td>noarch</td>
<td>20.2.4-7.module+el8.6.0+20625+ee813db2</td>
<td>o18_appstream</td>
<td>1.9 M</td>
</tr>
<tr>
<td>python39-setuptools</td>
<td>noarch</td>
<td>50.3.2-4.module+el8.5.0+20364+c7fe1181</td>
<td>o18_appstream</td>
<td>871 k</td>
</tr>
</tbody>
</table>

65. You can now verify if MySQL Shell has successfully installed on your on-premises instance by executing the below command:

```bash
$ mysqlsh --version
```

![Image of mysqlsh --version output]

66. To login to your on-premises MySQL using MySQL Shell, use the below commands:

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

-OR-

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

![Image of MySQL Shell login]

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`.

30 Migration Guide: On-premises MySQL to MySQL HeatWave on Oracle Cloud Infrastructure (OCI)
V) Connect to the on-premises MySQL using MySQL Shell. Afterwards, execute the MySQL Shell's `util.copyInstance()` utility to export all schemas (including users, indexes, routines, triggers) from on-premises MySQL to the MySQL HW on OCI.

67. Before connecting to on-premises MySQL 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 &`

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

- $ tmux
- $ mysqlsh <user>@<hostname>:<port-number>
- OR-
- $ mysqlsh -u <user> -p -h <hostname> -P <port-number>

- [opc@linux-8 ~] $ tmux

- [opc@linux-8 ~] $ mysqlsh root@localhost:3306

Please provide the password for 'root@localhost:3306': ********
Save password for 'root@localhost:3306'? [Y]es/[N]o/[N]e[v]er (default No):
MySQL Shell 8.1.1

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 'root@localhost:3306'
Fetching schema names for auto-completion... Press ^C to stop.
Your MySQL connection id is 12
Server version: 8.0.33 MySQL Community Server - GPL
No default schema selected; type \use <schema> to set one.

MySQL localhost:3306 ssl JS >
69. Change to the JavaScript mode of MySQL Shell and run the `util.copyInstance()` utility to export all on-premises MySQL data into OCI MySQL HW.

```javascript
MySQL JS> \js
MySQL JS> util.copyInstance('mysql://admin@10.0.1.140', {"compatibility":
"force_innodb", "skip_invalid_accounts", "strip_definers",
"strip_restricted_grants", "strip_tablespaces", "ignoreWildcard_grants",
"strip_invalid_grants", "createInvalid_pks"}, users: "true", threads: 4, dryRun:"true")
```

Note: replace the username (admin) and IP address (10.0.1.140) with your MySQL HW username and IP address (not the on-premises MySQL username and IP address).

```
Copying DDL, Data and Users from in-memory FS, source: linux-8:3306, target: wxchfcv4fxym5b6:3306.
SRC: dryRun enabled, no locks will be acquired and no files will be created.
SRC: Acquiring global read lock
SRC: Global read lock acquired
Initializing - done
SRC: 1 out of 5 schemas will be dumped and within them 3 tables, 0 views.
SRC: 1 out of 4 users will be dumped.
Gathering information - done
SRC: All transactions have been started
SRC: Locking instance for backup
SRC: Global read lock has been released
SRC: Checking for compatibility with MySQL Database Service 8.1.1

 [... output truncated]
TGT: Executing view DDL...
TGT: Executing view DDL - done
TGT: Loading data...
TGT: Recreating indexes...
TGT: Starting data load
TGT: Waiting for more data to become available...
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
TGT: Executing common postamble SQL
?% (0 bytes / ?), 0.00 B/s, 3 / 3 tables done
Recreating indexes - done
TGT: No data loaded.
TGT: 0 accounts were loaded
TGT: 0 warnings were reported during the load.

---

Dump metadata:
  Binlog_file: binlog.000001
  Binlog_position: 736682
  Executed_GTID_set: ''
MySQL localhost:33060+ ssl JS >
```
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 the 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 which 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.
70. Once you have run the command in step 69 and did not see any errors in the output (warnings are okay), run the same step 69 command but this time change the `dryRun` option to `false`.

MySQL JS> util.copyInstance('mysql://admin@10.0.1.140', {"compatibility": ["force_innodb", "skip_invalid_accounts", "strip_definers", "strip_restricted_grants", "strip_tablespaces", "ignore_wildcard_grants", "strip_invalid_grants", "create_invisible_pks"], users: "true", threads: 4, dryRun:"false")

Note: replace the username (admin) and IP address (10.0.1.140) with your MySQL HW username and IP address (not the on-premises MySQL username and IP address).

MySQL localhost:3306+ ssl JS> util.copyInstance('mysql://admin@10.0.1.140', {"compatibility": ["force_innodb", "skip_invalid_accounts", "strip_definers", "strip_restricted_grants", "strip_tablespaces", "ignore_wildcard_grants", "strip_invalid_grants", "create_invisible_pks"], users: "true", threads: 4, dryRun:"false")


SRC: Acquiring global read lock
SRC: Global read lock acquired
Initializating - done
SRC: 1 out of 5 schemas will be dumped and within them 3 tables, 0 views.
SRC: 1 out of 4 users will be dumped.
Gathering information - done
SRC: All transactions have been started
SRC: Locking instance for backup
SRC: Global read lock has been released
SRC: Checking for compatibility with MySQL Database Service 8.1.1

[...output truncated]

SRC: Starting data dump
100% (5.30K rows / 5.27K rows), 0.00 rows/s, 0.00 B/s
SRC: Dump duration: 00:00:00s
SRC: Total duration: 00:00:00s
SRC: Schemas dumped: 1
SRC: Tables dumped: 3
SRC: Data size: 194.62 KB
SRC: Rows written: 5302
SRC: Bytes written: 194.62 KB
SRC: Average throughput: 194.62 KB/s
TGT: Executing common postamble SQL
100% (194.62 KB / 194.62 KB), 38.91 KB/s, 3 / 3 tables done
Recreating indexes - done
TGT: 3 chunks (5.30K rows, 194.62 KB) for 3 tables in 1 schemas were loaded in 5 sec (avg throughput 38.45 KB/s)
TGT: 1 accounts were loaded
TGT: 0 warnings were reported during the load.

Dump_metadata:
  Binlog_file: binlog.000001
  Binlog_position: 735682
  Executed_GTID_set: '"

MySQL localhost:3306+ ssl JS> 

Note: once the MySQL Shell copy utility finishes, all your data will be copied over from on-premises MySQL to OCI MySQL HW. This completes the migration process. You can end your tmux session.
VI) (Optional) On OCI, use the Cloud Shell to verify whether the data was migrated successfully from on-premises MySQL to MySQL HeatWave on OCI.

71. Login to OCI, navigate to the top right corner and click on Developer tools right next to your OCI Region.

72. Click Cloud Shell.

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

75. On the Private network definition list form, select **Create private network definition**.
76. 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**.

77. Click **Close**.
78. 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.

   ![Cloud Shell](image)

   **Welcome to Oracle Cloud Shell.**

79. 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>
   ```
80. Change to the SQL mode of MySQL Shell and run the below commands:

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

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

```sql
MySQL SQL> SELECT COUNT(*) FROM <schema-name>.<table-name>;
```
82. Here is our row count comparison for on-premises MySQL and OCI MySQL HW:

On-premises MySQL 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)
```

OCI 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)
```

83. After validating, you can have your application/s point to the new OCI 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.

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

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

86. Click More actions and click Add HeatWave cluster.
87. Click **Estimate node**.

88. 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.
89. 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.

90. 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**.
91. 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.

92. The HeatWave cluster will be ready within a few minutes. You should see the HeatWave state change from **Creating** to **Active**.
93. 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]/[N]/o/[N]e[v]er (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-u1-cloud MySQL Enterprise - Cloud
No default schema selected; type \use <schema> to set one.
```

$ MySQL 10.0.1.140:33060+ ssl JS >
94. 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 JS> \sql
MySQL SQL> CALL sys.heatwave_load(JSON_ARRAY('world'), NULL);

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

![SQL output](image)

95. You now have a complete MySQL HeatWave cluster.

To learn more about using HeatWave, please visit [our documentation](#).