Live Migration Guide: MySQL On-premises 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 live migration document only covers how to migrate your database from MySQL on-premises to MySQL HeatWave (HW) on OCI. Before performing the migration, you should have considered downtime (even though this is a live migration - some/minimal downtime will be required to make sure your database application points to the new MySQL HW database once migrated), 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 live migration method shown in this guide works for MySQL on-premises v5.7 and above. This can be a MySQL Community Edition, MySQL Standard Edition, MySQL Enterprise Edition, or a 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.
- This live migration method requires binary logs to be present on the on-premises MySQL instance. To enable binary logs - you must set the `log_bin` variable to ON. After you have made sure that binary logging is enabled for your on-premises MySQL, ensure that its `binlog_format` system variable is set to ROW (as MySQL HW on OCI only uses row-based binary logging). Any other values besides ROW will not work. For more information on how to enable the on-premises MySQL binary logging and how to change the binary log format, see MySQL Binary Log and Binary Logging Formats.
- This live migration can be performed using two replication methods - using GTIDs and binary log position. As MySQL HW only supports GTIDs on OCI, once you migrate your on-premises MySQL instance to MySQL HW - you cannot go back to using the binary log position for replication.
- 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 writer or reader instance, although it is recommended to use the reader instance for the migration when applicable. This is because if you have a high concurrency for your on-premises MySQL instance, performing the migration using the writer instance could negatively impact the database application performance.
- The Overview section of this live migration guide contains all the steps that are needed to finish the database migration from on-premises MySQL to MySQL HW on OCI.
- In the Walkthrough section of this live migration 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 live migration from MySQL on-premises 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 MySQL on-premises to MySQL HW on OCI using live migration (with zero or minimal downtime):

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 on-premises MySQL to connect to MySQL HW on OCI and it also ensures that your data in transit while it is being migrated is encrypted.]


III) On OCI, create a standalone MySQL HW instance.

[If you require High Availability for your MySQL HW instance, you must enable it after completing section VIII of this guide.]


IV) Install MySQL Shell 8.2 or above on an on-premises instance that can connect to MySQL on-premises.

[MySQL Shell will be used to copy DDL and data from on-premises MySQL to MySQL HW on OCI. You must download MySQL Shell 8.2 or above.]

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


V) For your on-premises MySQL, ensure log_bin is set to ON, binlog_format is set to ROW, and increase the binlog_expire_logs_seconds system variable if needed - to retain binary logs for a longer period (if using MySQL 5.6 or 5.7, increase the expire_logs_days system variable).

[The on-premises MySQL binary logs are needed to set up replication from MySQL on-premises to MySQL HW on OCI for data synchronization. The on-premises MySQL binary logs need to be retained until replication is set up from on-premises MySQL to MySQL HW and all the pending transactions from MySQL on-premises have been replicated to MySQL HW. Adjust your binlog_expire_logs_seconds or expire_logs_days accordingly. The default values for binlog_expire_logs_seconds and expire_logs_days are 2592000 and 0 respectively.]

Enabling MySQL Binary Logging:

Setting The MySQL Binary Log Format:

MySQL Binary Log Expiration:
VI) Connect to MySQL on-premises using MySQL Shell and create a replication user. Afterwards, execute MySQL Shell's `util.copyInstance()` utility to export all schemas (including users, indexes, routines, triggers) from MySQL on-premises to MySQL HW on OCI. After the `util.copyInstance()` utility finishes, save the MySQL Shell `Dump_metadata` values.

[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. MySQL Shell's `Dump_metadata` values will let the MySQL HW instance on OCI know where to start the replication from for data synchronization.]


VII) On OCI, create a replication channel to set up replication from MySQL on-premises to MySQL HW on OCI. During the channel creation process, if the on-premises MySQL instance is using binary log positioning - under the replication positioning section, select Source cannot use GTID auto-positioning and provide the binlogFile and binlogPosition values. If the on-premises MySQL instance is using GTIDs - select Source can use GTID auto-positioning (recommended). Create the replication channel afterwards.

[Setting up this replication channel will propagate all the pending data changes to MySQL HW that had occurred on the on-premises MySQL after the execution of MySQL Shell `util.copyInstance()` utility.]


VIII) After the replication channel is up, connect to MySQL HW and execute the `SHOW REPLICA STATUS` command. From the query output, look for the `seconds_behind_source` and `Replica_SQL_Running_State` fields. If the `seconds_behind_source` field displays a value of 0 and the `Replica_SQL_Running_State` field displays a message of Replica has read all relay log; waiting for more updates - this indicates that the MySQL HW instance has fully caught up with the on-premises MySQL changes and the replication channel can now be disabled.

[During this step, it is recommended to stop the database application for ~5 minutes to ensure that no writes are happening to the on-premises MySQL instance before the replication channel between MySQL HW and on-premises MySQL is disabled. After the replication channel has been disabled, you may turn on High Availability for your MySQL HW instance.]


IX) At this point, the live migration process for the database is complete. The database applications can now point to MySQL HW on OCI.

X) (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

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 or FastConnect.

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.

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 **VM.Standard2.2**. 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/

   ![OpenVPN Access Server](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).

![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**.

![User Permissions](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**. **Save the VPN Static IP Address** for later use. 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. Scroll down and 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>**.

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.

![Security List Details](image)

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**.

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**.

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 Live Migration.
III) On OCI, create a standalone 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**, do not choose High Availability (HA) here as replicating to a MySQL HA instance on OCI for this migration may create some complications. You may enable HA after you have completed section **VIII** of this live migration guide. Information on how to enable HA later can be found [here](#). 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 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 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 **8.0.35 - Bug fix**.

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

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

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.

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 or above on an on-premises instance that can connect to MySQL on-premises.

61. Have an on-premises instance that can connect to your on-premises MySQL. Go to the below website and download MySQL Shell 8.2 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 8.2.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.

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 MySQL and execute the below command to download MySQL Shell:

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

Replace the below link with what you have.

```bash
$ wget https://dev.mysql.com/get/Downloads/MySQL-Shell/mysql-shell-8.2.0-1.e18.x86_64.rpm
```

```
...--2023-10-26 20:42:16--  https://dev.mysql.com/get/Downloads/MySQL-Shell/mysql-shell-8.2.0-1.e18.x86_64.rpm
Resolving dev.mysql.com (dev.mysql.com)... 104.108.116.193, 2600:1408:c400:1880::2e31, 2600:1408:c400:1881::2e31
Connecting to dev.mysql.com (dev.mysql.com)|104.108.116.193|:443... connected.
HTTP request sent, awaiting response... 302 Moved Temporarily
Location: https://cdn.mysql.com//Downloads/MySQL-Shell/mysql-shell-8.2.0-1.e18.x86_64.rpm [following]
...--2023-10-26 20:42:16--  https://cdn.mysql.com//Downloads/MySQL-Shell/mysql-shell-8.2.0-1.e18.x86_64.rpm
Resolving cdn.mysql.com (cdn.mysql.com)... 23.204.255.142, 2600:1408:c400:1884::1d68, 2600:1408:c400:188d::1d68
Connecting to cdn.mysql.com (cdn.mysql.com)|23.204.255.142|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 31610964 (30M) [application/x-redhat-package-manager]
Saving to: ‘mysql-shell-8.2.0-1.e18.x86_64.rpm’
```

```
mysql-shell-8.2.0-1 100%[===================================================================================] 30.15M 83.5MB/s in 0.4s
2023-10-26 20:42:17 (83.5 MB/s) - ‘mysql-shell-8.2.0-1.e18.x86_64.rpm’ saved [31610964/31610964]
```
64. After downloading the MySQL Shell rpm, install MySQL Shell:

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

```
[opc@linux-8 ~]$ sudo yum localinstall mysql-shell*
Last metadata expiration check: 0:00:05 ago on Thu 26 Oct 2023 08:52:38 PM GMT. Dependencies resolved.

Package Arch Version Repository Size
==============================================
Installing:
mysql-shell x86_64 8.2.0-1.el8 @commandline 30 M
Installing dependencies:
python39-libs x86_64 3.9.16-1.module+el8.8.0+90007+d415a2d2.2 ol8_appstream 8.2 M
python39-pip-wheel noarch 20.2.4-7.module+el8.6.0+20625+ee813db2 ol8_appstream 1.1 M
python39-setuptools-wheel noarch 50.3.2-4.module+el8.5.0+20364+c7fe1181 ol8_appstream 497 k
```

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

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

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

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>
```

```
[opc@linux-8 ~]$ mysqlsh root@localhost:3306
Please provide the password for 'root@localhost:3306': ********
Save password for 'root@localhost:3306'? [Y]/[N]/[o]/[Ne][v][er] (default No): Y
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 session to 'root@localhost:3306'
Fetching schema names for auto-completion... Press ^C to stop.
Your MySQL connection id is 10
Server version: 8.0.33 MySQL Community Server - GPL
No default schema selected; type \use <schema> to set one.
```

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) For your on-premises MySQL, ensure `log_bin` is set to ON, `binlog_format` is set to ROW, and increase the `binlog_expire_logs_seconds` system variable if needed - to retain binary logs for a longer period (if using MySQL 5.6 or 5.7, increase the `expire_logs_days` system variable).

67. Stay connected to your on-premises MySQL and execute the below commands to ensure your on-premises MySQL is configured correctly for the live migration.

MySQL JS> \sql
MySQL SQL> SELECT @@log_bin;
MySQL SQL> SELECT @@binlog_format;

```
MySQL localhost:33060+ ssl JS> \sql
Switching to SQL mode... Commands end with ;
Fetching global names for auto-completion... Press ^C to stop.
MySQL localhost:33060+ ssl SQL> SELECT @@LOG_BIN;
+-------+
<table>
<thead>
<tr>
<th>@LOG_BIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
</tbody>
</table>
+-------+
1 row in set (0.0020 sec)
MySQL localhost:33060+ ssl SQL> SELECT @@BINLOG_FORMAT;
+-------------------+
<table>
<thead>
<tr>
<th>@BINLOG_FORMAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROW</td>
</tr>
</tbody>
</table>
+-------------------+
1 row in set (0.0020 sec)
```

Note: you must have a value of 1 for `log_bin` and a value of ROW for `binlog_format`.  


68. In order to perform the live database migration - we will need to retain the current binary log that is in use/will be used during the data export of on-premises MySQL to OCI MySQL HW and the binary logs that will be generated afterwards. The binary logs will be needed until the replication setup is completed on OCI. Since the sample database ‘world’ (the one that will be migrated to MySQL HW on OCI for this example step-by-step guide) is fairly small, we have kept the `binlog_expire_logs_seconds` to its default value of 2592000. Set the `binlog_expire_logs_seconds` (if using MySQL v8.0 or above) or `expire_logs_days` (if using MySQL v5.6 or v5.7) accordingly depending on the data that you are migrating; high volumes of data will require a longer retention period. View if the binary logs are currently present and verify if the binary log retention period is set to your desired value.

MySQL SQL> SHOW BINARY LOGS;
MySQL SQL> SELECT @@binlog_expire_logs_seconds;
or
MySQL SQL> SELECT @@expire_logs_days;

<table>
<thead>
<tr>
<th>Log_name</th>
<th>File_size</th>
<th>Encrypted</th>
</tr>
</thead>
<tbody>
<tr>
<td>binlog.000104</td>
<td>220</td>
<td>No</td>
</tr>
<tr>
<td>binlog.000105</td>
<td>220</td>
<td>No</td>
</tr>
<tr>
<td>binlog.000106</td>
<td>220</td>
<td>No</td>
</tr>
<tr>
<td>binlog.000107</td>
<td>220</td>
<td>No</td>
</tr>
<tr>
<td>binlog.000108</td>
<td>220</td>
<td>No</td>
</tr>
<tr>
<td>binlog.000109</td>
<td>220</td>
<td>No</td>
</tr>
<tr>
<td>binlog.000110</td>
<td>220</td>
<td>No</td>
</tr>
<tr>
<td>binlog.000111</td>
<td>220</td>
<td>No</td>
</tr>
<tr>
<td>binlog.000112</td>
<td>220</td>
<td>No</td>
</tr>
<tr>
<td>binlog.000113</td>
<td>220</td>
<td>No</td>
</tr>
<tr>
<td>binlog.000114</td>
<td>508</td>
<td>No</td>
</tr>
</tbody>
</table>

11 rows in set (0.0142 sec)

MySQL SQL> SHOW BINARY LOGS;

<table>
<thead>
<tr>
<th>@binlog_expire_logs_seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>2592000</td>
</tr>
</tbody>
</table>

1 row in set (0.0017 sec)

Note: you can change the value of `binlog_expire_logs_seconds` and `expire_logs_days` by executing:
MySQL SQL> SET GLOBAL binlog_expire_logs_seconds = <number_of_seconds>;
MySQL SQL> SET GLOBAL expire_logs_days = <number_of_days>;

VI) Connect to MySQL on-premises using MySQL Shell and create a replication user. Afterwards, execute the MySQL Shell `util.copyInstance()` utility to export all schemas (including users, indexes, routines, triggers) from MySQL on-premises to MySQL HW on OCI. After the `util.copyInstance()` utility finishes, save the MySQL Shell `Dump_metadata` values.

69. Before 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 MySQL using MySQL Shell.

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

-OR-

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

71. Change to the SQL mode of MySQL Shell and create a replication user, we will use this user to establish a replication connection from on-premises MySQL to MySQL HW on OCI.

MySQL SQL> CREATE USER 'repl'@'%' IDENTIFIED BY '<password>');
MySQL SQL> GRANT REPLICATION SLAVE ON *.* TO 'repl'@'%';

Query OK, 0 rows affected (0.0143 sec)

MySQL SQL> CREATE USER 'repl'@'%' IDENTIFIED BY 'MySQL8.0';
MySQL SQL> GRANT REPLICATION SLAVE ON *.* TO 'repl'@'%';

Query OK, 0 rows affected (0.0017 sec)
72. Change to the JavaScript mode of MySQL Shell and run the `util.copyInstance()` utility to export all on-premises MySQL data into MySQL HeatWave on OCI.

MySQL JS> \js
MySQL JS> util.copyInstance('mysql://admin@10.0.1.73', {
"compatibility": [
"force_innodb", "skip_invalid_accounts", "strip_definers",
"strip Restricted_grants", "strip_tablespaces", "ignore wildcard grants",
"strip_invalid_grants", "create invisible pks"],
updateGtidSet: "append", users: 
"true", threads: 4, dryRun:"true"})

Note: replace the username (admin) and IP address (10.0.1.73) with your MySQL HW username and IP address (not the on-premises MySQL username and IP address). You will then be prompted to enter your MySQL HW password after executing the copy command.

MySQL localhost:3306+ ssl JS > util.copyInstance('mysql://admin@10.0.1.73', {
"compatibility": [
"force_innodb", "skip invalid accounts", "strip_definers",
"strip Restricted_grants", "strip_tablespaces", "ignore wildcard grants",
"strip invalid grants", "create invisible pks"],
updateGtidSet: "append", users: 
"true", threads: 4, dryRun:"true"})

SRC: dryRun enabled, no locks will be acquired and no files will be created.
SRC: Acquiring global read lock
SRC: Global read lock acquired
SRC: Initializing - done
SRC: 1 out of 5 schemas will be dumped and within them 5 tables, 0 views.
SRC: 8 out of 11 users will be dumped.

[... output truncated]

TGT: Starting data load
?% (0 bytes / ?), 0.00 B/s, 0 / 5 tables done
Recreating indexes - done
TGT: Executing common postamble SQL
TGT: No data loaded.
TGT: 0 accounts were loaded
TGT: 0 warnings were reported during the load.

---

Dump metadata:
Binlog_file: binlog.000114
Binlog_position: 197
Executed_GTID_set: a1fbebdc3a71-8ad2-4f2d-8977:1-24

MySQL localhost:3306 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 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.

- `updateGtidSet`: append: If your RDS MySQL is using GTIDs, for inbound replication, adds the transaction IDs from the source `gtid_executed` GTID set, to the replica `gtid_purged` GTID set. This lets you begin replication from the source without re-executing every past transaction from the source. Adding the GTIDs to `gtid_purged` tells the replica that those transactions have already been executed, although they are not present in the source binary log. This must be set to `append` during a live migration.

- `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.

73. Once you have run the command in step 72 and did not see any errors in the output (warnings are okay), run the same step 72 command but this time change the dryRun option to false.


Note: replace the username (admin) and IP address (10.0.1.73) with your MySQL HW username and IP address (not the on-premises MySQL username and IP address).

```
100% (194.62 KB / 194.62 KB), 128.54 KB/s, 5 / 5 tables done
Recreating indexes - done
TGT: 5 chunks (5.30K rows, 194.62 KB) for 5 tables in 1 schemas were loaded in 16 sec (avg throughput 128.47 KB/s)
TGT: 8 accounts were loaded
TGT: 0 warnings were reported during the load.
```
VII) On OCI, create a replication channel to set up replication from MySQL on-premises to MySQL HW on OCI. During the channel creation process, if the on-premises MySQL instance is using binary log positioning - under the replication positioning section, select Source cannot use GTID auto-positioning and provide the binlogFile and binlogPosition values. If the on-premises MySQL instance is using GTIDs - select Source can use GTID auto-positioning (recommended). Create the replication channel afterwards.

75. After your data has successfully imported into MySQL HW, from the OCI Console, click on the navigation menu again, go to Databases, and click Channels.

76. Click Create channel to set up replication between MySQL on-premises and MySQL HW on OCI.

77. Ensure you are in the right compartment and enter a replication channel name. Ensure that the Enabled automatically upon creation box is checked.
78. Under **Source connection**, for **Hostname** input your OpenVPN’s **VPN Static IP Address** from step 26. For **Port**, specify the port number the on-premises MySQL listens on - the default is **3306**. For **Username** and **Password** - specify the **replication username and password** for the account that you created on the on-premises MySQL instance.

79. For **SSL mode** select the one that meets your need. For this guide, we have chosen **Required (REQUIRED)**.
80. For **Replication positioning**, if your on-premises MySQL uses binary log positioning – select **Source cannot use GTID auto-positioning**. Keep the **UUID** field as-is, for **Binary log file name** and **Binary log offset**, input the **Binlog_file** and **Binlog_position** values respectively from the MySQL Shell’s **Dump_metadata** that you had saved from step 73.

81. For **Replication positioning**, if your on-premises MySQL uses GTIDs – select **Source can use GTID auto-positioning (recommended)**.
82. Scroll down until you see **Tables without primary key**. If you plan on using the High Availability or HeatWave option, select **Generate primary key** since these options require primary keys on every table. If you don’t plan on using High Availability or HeatWave – you can either select **Raise an error** or **Allow**. For this guide, we have chosen **Allow**.

83. Under Tables without primary key, you should see **Target DB system**. Click **Select DB system**.
84. A list of your MySQL DB systems will open after completing the previous step. Select the **MySQL HW system** that you created earlier and click **Select DB system**.

85. Click **Create channel**.
The replication channel from your on-premises MySQL to MySQL HW on OCI will now start CREATING so that we can propagate all the pending data changes to MySQL HW that had occurred on the on-premises MySQL after the execution of MySQL Shell `util.copyInstance()` utility. Your channel should change its status to ACTIVE shortly if everything was done correctly.
VIII) After the replication channel is up, connect to MySQL HW and execute the `SHOW REPLICA STATUS\G` command. From the query output, look for the `seconds_behind_source` and `Replica_SQL_Running_State` fields. If the `seconds_behind_source` field displays a value of 0 and the `Replica_SQL_Running_State` field displays a message of `Replica has read all relay log; waiting for more updates` this indicates that the MySQL HW instance has fully caught up with the on-premises MySQL changes and the replication channel can now be disabled.

Note: During this step, it is recommended to stop the database application for ~5 minutes to ensure that no writes are happening to the on-premises MySQL instance before the replication channel between MySQL HW and on-premises MySQL is disabled.

87. Connect to your MySQL HW on OCI instance using MySQL Shell which is installed on your on-premises environment.

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

88. Switch to the SQL mode of MySQL Shell and run the below statement:

   MySQL SQL> SHOW REPLICA STATUS\G

89. If the replication is successfully ongoing from on-premises MySQL to MySQL HW, you should see the status of `Replica_IO_Running` and `Replica_SQL_Running` as `Yes`. If one or the other shows an output different than `Yes`, your replication has failed or encountered an error.
90. When executing the above SQL statement `SHOW REPLICA STATUS\G`, also look for `Seconds_Behind_Source` and `Replica_SQL_Running_State` values. If you see a value of 0 for `Seconds_Behind_Source` and a value string of `Replica has read all relay log; waiting for more updates` for `Replica_SQL_Running_State` - this suggests that MySQL HW instance has fully caught up with the on-premises MySQL instance and there are no pending transactions on MySQL on-premises that needs to be replicated to MySQL HW.

![MySQL REPLICA STATUS](image)

91. You can go back to the OCI MySQL Channels page and **Disable** the Channel.

![Disable Channel](image)

92. Once the channel is disabled, you may [enable HA](#) for your MySQL HW instance.
IX) At this point, the live migration process for the database is complete. The database applications can now point to MySQL HW on OCI.

X) (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.

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

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

95. Click More actions and click Add HeatWave cluster.
96. Click **Estimate node**.

97. 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.
98. 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.

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

102. The HeatWave cluster will be ready within a few minutes. You should see the HeatWave state change from Creating to Active.
Connect to your MySQL HW instance using MySQL Shell that is installed on your on-premises environment.

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

-OR-

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

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@10.0.1.73'
Fetching schema names for auto-completion... Press ^C to stop.
Your MySQL connection id is 82 (X protocol)
Server version: 8.0.35-cloud MySQL Enterprise - Cloud
No default schema selected; type \use <schema> to set one.
104. 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 engine.

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

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

```
MySQL | 10.0.1.73:33060+ ssl SQL > CALL sys.heatwave_load(JSON_ARRAY('world'), NULL);

INITIALIZING HEATWAVE AUTO PARALLEL LOAD

Version: 2.20
Load Mode: normal
Load Policy: disable_unsupported_columns
Output Mode: normal

6 rows in set (1.9769 sec)

OFFLOAD ANALYSIS

Verifying input schemas: 1
User excluded items: 0

<table>
<thead>
<tr>
<th>SCHEMA NAME</th>
<th>OFFLOADABLE TABLES</th>
<th>OFFLOADABLE COLUMNS</th>
<th>SUMMARY OF ISSUES</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>world</code></td>
<td>5</td>
<td>26</td>
<td></td>
</tr>
</tbody>
</table>

Total offloadable schemas: 1

[...output truncated]

LOAD SUMMARY

<table>
<thead>
<tr>
<th>SCHEMA NAME</th>
<th>TABLES LOADED</th>
<th>TABLES FAILED</th>
<th>COLUMNS LOADED</th>
<th>LOAD DURATION</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>world</code></td>
<td>5</td>
<td>0</td>
<td>26</td>
<td>1.86 s</td>
</tr>
</tbody>
</table>

6 rows in set (1.9769 sec)

Query OK, 0 rows affected (1.9769 sec)
```

105. You now have a complete MySQL HeatWave cluster.

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