MySQL HeatWave on AWS Service Guide

F82097-05

MySQL HeatWave on AWS Service Guide ,

F82097-05

Primary Author: Aijaz Fatima

Contents

4 HeatWave Clusters

5 Connecting to a DB System

6 Importing Data

Preface and Legal Notices

This is the *MySQL HeatWave on AWS Service Guide*.

between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

This documentation is NOT distributed under a GPL license. Use of this documentation is subject to the following terms:

You may create a printed copy of this documentation solely for your own personal use. Conversion to other formats is allowed as long as the actual content is not altered or edited in any way. You shall not publish or distribute this documentation in any form or on any media, except if you distribute the documentation in a manner similar to how Oracle disseminates it (that is, electronically for download on a Web site with the software) or on a CD-ROM or similar medium, provided however that the documentation is disseminated together with the software on the same medium. Any other use, such as any dissemination of printed copies or use of this documentation, in whole or in part, in another publication, requires the prior written consent from an authorized representative of Oracle. Oracle and/or its affiliates reserve any and all rights to this documentation not expressly granted above.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at https://www.oracle.com/corporate/accessibility/.

Access to Oracle Support for Accessibility

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit https://www.oracle.com/corporate/accessibility/learning-support.html#support-tab.

1 Overview

MySQL HeatWave on AWS is a fully managed service, developed and supported by the MySQL team at Oracle. Oracle automates tasks such as database and operating system patching. You are responsible for managing your data, schema designs, and access.

With MySQL HeatWave on AWS, you can create and manage a MySQL DB System with a HeatWave Cluster to use with AWS applications.

MySQL Database

The MySQL Database is built on the MySQL Enterprise Edition Server, which allows developers to quickly create and deploy secure cloud native applications using the world's most popular open source database. It is the only MySQL database that supports HeatWave. For MySQL Server documentation, refer to the MySQL Reference Manual .

MySQL HeatWave on AWS supports MySQL Enterprise Edition 8.0. You can choose from the latest MySQL 8.0 releases when creating a DB System. See Creating a DB System.

HeatWave

HeatWave is a massively parallel, high performance, in-memory query accelerator that accelerates MySQL performance by orders of magnitude for analytics and mixed workloads.

With MySQL HeatWave on AWS, you can provision DB Systems and HeatWave Clusters. HeatWave consists of a MySQL DB System and one or more HeatWave nodes. The MySQL DB System node is responsible for cluster management, query scheduling, and returning query results. HeatWave nodes store data in memory and process analytics queries.

When a HeatWave Cluster is enabled, queries that meet certain prerequisites are automatically offloaded from the MySQL DB System to the HeatWave Cluster for accelerated processing. Queries are issued from Query Editor in the MySQL HeatWave Console or from a MySQL client or application that interacts with the HeatWave Cluster by connecting to the MySQL DB System.

> **Tip:**
>
> This *MySQL HeatWave on AWS Service Guide* covers what you need to know to set up and use DB Systems and HeatWave Clusters on AWS, mainly using the MySQL HeatWave Console. For more detailed information about using HeatWave and related utilities, refer to the MySQL HeatWave User Guide . Some of that information does not apply to MySQL HeatWave on AWS, so be sure to check the *MySQL HeatWave on AWS Service Guide* first for the most relevant information.

Integration with Oracle Cloud Infrastructure (OCI)

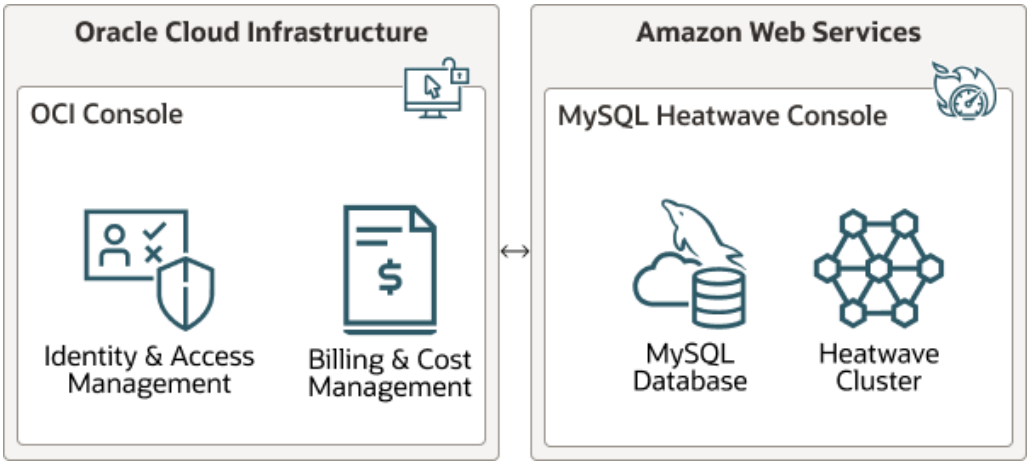
MySQL HeatWave on AWS is integrated with Oracle Cloud Infrastructure (OCI) for identity and access management, and for billing and cost management.

When you sign up for MySQL HeatWave on AWS, you are directed to OCI where you must sign up for an Oracle Cloud Account if you do not have one. After signing up, you are directed to the OCI Console to complete the sign-up process. When signing in to the MySQL HeatWave Console, you are seamlessly directed to OCI for authentication and then back to the MySQL HeatWave Console.

Identity and access management and billing for MySQL HeatWave on AWS is managed in OCI. For more information, see Billing.

The following diagram illustrates MySQL HeatWave on AWS integration with Oracle Cloud Infrastructure (OCI).

Figure 1-1 MySQL HeatWave on AWS and OCI Integration



Region Availability

MySQL HeatWave on AWS is currently available in five regions. Each AWS Region maps to an OCI region, and requires a subscription to that OCI region.

Table 1-1 OCI and AWS Regions

OCI Region name	OCI Region identifier	AWS Region name	AWS Region code
Germany Central (Frankfurt)	eu-frankfurt-1	Europe (Frankfurt)	eu-central-1
India West (Mumbai)	ap-mumbai-1	Asia Pacific (Mumbai)	ap-south-1
Japan East (Tokyo)	ap-tokyo-1	Asia Pacific (Tokyo)	ap-northeast-1
UK South (London)	uk-london-1	Europe (London)	eu-west-2
US East (Ashburn)	us-ashburn-1	US East (N. Virginia)	us-east-1

See Subscribed Region Limits.

Identity and Access Management

MySQL HeatWave on AWS integrates with Oracle Cloud Infrastructure (OCI) for identity and access management.

MySQL HeatWave on AWS uses predefined OCI Identity and Access Management (IAM) groups to control access to the MySQL HeatWave Console. For example, predefined group membership determines who can create DB Systems and HeatWave Clusters and who can use them. An Administrator in your organization is responsible for adding users to the appropriate groups. For more information, see User and Group Management.

MySQL HeatWave on AWS also supports federation with third-party Identity Providers (IdPs). See Federating with Identity Providers.

If you are a regular user (not an Administrator) who needs access to MySQL HeatWave on AWS, contact your Administrator to set up an account. An OCI IAM user account is required to access the MySQL HeatWave Console, and that user must be added to one of the predefined OCI IAM groups. For more information, see User and Group Management.

A separate MySQL user account is required to access MySQL HeatWave on AWS from a MySQL client or application. This can be the MySQL Administrator user specified when creating the DB System (see Creating a DB System) or a MySQL user account created on the DB System using `CREATE USER` . If you are not the MySQL Administrator and you do not have a MySQL user account, have your MySQL Administrator create one for you.

Security

Oracle considers cloud security its highest priority. The following security features help keep your data safe and secure.

- Database access control and account management

  MySQL provides security features to control access and manage your account. See Access Control and Account Management.

- Connection Control

  MySQL HeatWave on AWS supports a plugin library that enables Administrators to introduce an increasing delay in server response to connection attempts after a configurable number of consecutive failed attempts. This capability provides a deterrent that slows down brute force attacks against MySQL user accounts. The default connection control settings, which cannot be modified, are as follows:

  - `connection_control_failed_connections_threshold`
  - `connection_control_max_connection_delay`
  - `connection_control_min_connection_delay`

  See The Connection-Control Plugins.

- Encryption at rest

  Your data is always encrypted at rest. MySQL HeatWave on AWS uses Amazon EBS encryption. Boot volumes are encrypted on MySQL DB System and HeatWave nodes, and the database volume is encrypted on the MySQL DB System. For information about EBS encryption, see Amazon EBS Encryption, in the *Amazon EC2 User Guide for Linux Instances*.

- Encryption in transit

  Your data is always encrypted while in transit. MySQL HeatWave on AWS supports TLSv1.2 and requires that all MySQL client and application connections are encrypted. For added security, download a signed certificate bundle to enable host name identity verification for connecting clients and applications. For more information, see Enabling Host Name Identity Verification.

- MySQL Enterprise Data Masking and De-Identification

  Use data masking and de-identification to protect your sensitive data. MySQL HeatWave on AWS supports various MySQL data masking functions that mask data to remove identifying characteristics and generate random data with specific characteristics. The following data masking functions are supported:

  - `gen_range()`
  - `gen_rnd_email()`
  - `gen_rnd_ssn()`
  - `gen_rnd_us_phone()`
  - `mask_inner()`
  - `mask_outer()`
  - `mask_pan()`
  - `mask_pan_relaxed()`
  - `mask_ssn()`

  For more information, see MySQL Enterprise Data Masking and De-Identification.

- Password Validation

  MySQL HeatWave on AWS enforces strong passwords with the `validate_password` component, serves to improve security by requiring account passwords and enabling strength testing of potential passwords. The default value of the variables of the validate_password component are as follows, and you cannot change the default values:

  - `validate_password.check_user_name`
  - `validate_password.length`
  - `validate_password.mixed_case_count`
  - `validate_password.number_count`
  - `validate_password.policy`
  - `validate_password.special_char_count`

  Make sure your applications comply with the password requirements. For more information, see The Password Validation Component.

- MySQL Enterprise Firewall

  MySQL Enterprise Firewall enables database Administrators to permit or deny SQL statement execution based on matching against lists of accepted statement patterns. This helps harden MySQL against attacks such as SQL injection or attempts to exploit applications by using them outside of their legitimate query workload characteristics. For more information, see MySQL Enterprise Firewall.

2 Getting Started

This chapter describes how to access MySQL HeatWave on AWS, how to sign up for the MySQL HeatWave on AWS service, how to sign in, and provides an introduction to the MySQL HeatWave Console.

Accessing MySQL HeatWave on AWS

Access MySQL HeatWave on AWS with the MySQL HeatWave Console which is a browser-based interface. Access the MySQL HeatWave Console at the following address:

https://cloud.mysql.com/

If necessary, first register for MySQL HeatWave on AWS, see: Signing Up. Then sign-in to MySQL HeatWave on AWS, see Signing In.

For instructions to use the MySQL HeatWave Console, see: Using the Console.

After signing in and creating a DB System, access MySQL HeatWave on AWS from a MySQL client or application. A MySQL user account is required, see Connecting from a Client.

Signing Up

This procedure describes how to sign up for the MySQL HeatWave on AWS Service.

Sign-up Overview

After the initial Oracle Cloud Account setup, use the MySQL HeatWave on AWS Administration page on the OCI console for these items:

- Request an OCI account upgrade from the Oracle Cloud Free Tier to a paid account.
- Subscribe to an OCI region, see: Region Availability.
- Enable the MySQL HeatWave on AWS service in one or more AWS regions.

The MySQL HeatWave on AWS Administration page shows the current status of the sign-up process, and the status of each request.

> **Tip:**
>
> An Oracle Cloud Account has a single region limit by default. For the Home Region, choose the OCI region that maps to the preferred AWS region to avoid the need for a region increase. This would require a support request, and can take some time. See: Region Availability and Subscribed Region Limits.

NOT_SUPPORTED:

Do not change the Oracle Cloud Account name after enabling the MySQL HeatWave on AWS service, as it can cause a loss of access to the MySQL HeatWave on AWS service, and require assistance from Customer Support.

Note:

After upgrading to a paid account, any credits from the Oracle Cloud Free Tier free trial remain eligible for use with MySQL HeatWave on AWS for the duration of the trial period.

The OCI user account created during the sign-up process is added to the `Administrators` group in the OCI `Default` identity domain. See: The Default Identity Domain.

To sign up for MySQL HeatWave on AWS, the registered user must either be a member of the `Administrators` group in the OCI `Default` identity domain or have the following permissions:

- `manage heatwave-registration`
- `manage policies`
- `manage domains`
- `manage groups`
- `read limits`
- `update tenancies`

For information on creating policies, see Getting Started with Policies.

Sign-up Procedure

For the **Home Region**, choose the OCI region that maps to the preferred AWS region. See: Region Availability.

To sign up for MySQL HeatWave on AWS:

1. Start here to create an Oracle Cloud Account, or proceed to Step 2 to start the sign-up process.
   a. Navigate to https://cloud.mysql.com and click **Sign Up**.
   b. Enter an email address and click **Continue**.
   c. Enter the **Account Information** and click **Verify my email**.
   d. From the verification email, click **Verify email** to verify the email address and continue the account setup process.
   e. Complete the **Account Information** and click **Continue**. This includes an OCI subscription to the **Home Region**.

f. Complete the **Address Information** and click **Continue**.

g. Click **Add payment verification method** and complete **Payment Verification**.

h. Select **Agreement**, and click **Start my free trial**.

i. Click **Go to service**. Proceed to step 3.

2. Existing Oracle Cloud account subscribers start here.

   a. Navigate to the OCI Console at https://cloud.oracle.com.

   b. Enter the Cloud Account name and click **Next**.

   c. Enter the user credentials and click **Sign In**.

   d. From the OCI Console, open the navigation menu, and select **Databases**. Under MySQL HeatWave on AWS, click **Administration**. .

3. Request an account upgrade, if necessary.

   a. Click **Request Upgrade**.

   b. From the **Payment Method** page, choose an **Account Type**, and click **Select**.

   c. Complete the payment information, select **terms and conditions**, and click **Start Paid Account**.

   d. From the **Upgrade Confirmation** dialog box, click **Continue**.

   e. From the OCI Console, open the navigation menu, and select **Databases**. Under MySQL HeatWave on AWS, click **Administration**.
   The account upgrade might take some time.

4. Enable the MySQL HeatWave on AWS service. The MySQL HeatWave on AWS **Administration** page shows **MySQL HeatWave on AWS Region Management**. If the OCI region that maps to the preferred AWS region shows **Subscribed**, do the following to enable the MySQL HeatWave on AWS service:

   a. Click **Enable** next to the preferred AWS region.

   b. From the **Enable Service** dialog box, click **Enable**.

   If the OCI region that maps to the preferred AWS region does not show **Subscribed**, do the following to subscribe to an OCI region, and then enable the MySQL HeatWave on AWS service:

   a. Click **Subscribe** next to the OCI region that maps to the preferred AWS region. If a region limit increase is required, do the following to request an increase:

      i. From the **Limit increase needed** dialog box, click **Open Service Options**.

      ii. Follow these steps: Requesting a Limit Increase to the Subscribed Region Count.

      iii. When the region limit increase is complete, click **Subscribe** next to the OCI region that maps to the preferred AWS region.

   b. From the **Subscribe to New Region** dialog box, read the privacy notice, and then click **Subscribe**.

   c. Click **Enable** next to the preferred AWS region.

   d. From the **Enable Service** dialog box, click **Enable**.

Choose one of these options:

- Click **Open MySQL HeatWave on AWS Console**to create a DB System and HeatWave Cluster. See: DB Systems, and HeatWave Clusters.

- Select **Users and Groups** to create and manage users or integrate with an identity provider. See User and Group Management.
- Select **Billing** to monitor usage and expenses for MySQL HeatWave on AWS in OCI. See Billing.

Signing In

This procedure assumes that you are an Oracle Cloud Account customer registered to use MySQL HeatWave on AWS. If you are not registered, see Signing Up.

Signing in to MySQL HeatWave on AWS requires:

- An Oracle Cloud Account name. This is the Oracle Cloud Account you registered with, chosen during account signup, or that was provided to you by an Account Administrator. In either case, you can find your Oracle Cloud Account name in your Oracle Cloud Account welcome email.
- Your user name and password.

If you forgot your Oracle Cloud Account name, click **Get help** in the sign-in dialog and enter the email address associated with the Cloud Account. Oracle will send you an email with a summary of your account information.

To sign in to your account:

1. Navigate your browser to https://cloud.mysql.com.

   You are directed to the MySQL HeatWave on AWS welcome page.
2. Enter your Oracle Cloud Account name.
3. Click **Continue**.

   You are directed to the **Oracle Cloud Account Sign In** dialog.
4. Enter your user name and password and click **Sign In**.

   Once your user name and password are authenticated, you are directed to the MySQL HeatWave Console.

Using the Console

The MySQL HeatWave Console supports browser platforms supported by Oracle Jet. See Platforms supported by Oracle JET, in the *JavaScript Extension Toolkit* documentation.

Firewalls, proxy servers, or other devices that control access to the internet can affect the ability to connect to the MySQL HeatWave Console and OCI Console.

To allow network access to both the MySQL HeatWave Console and OCI Console, add the following URLs to the allowlist for firewalls and proxy servers:

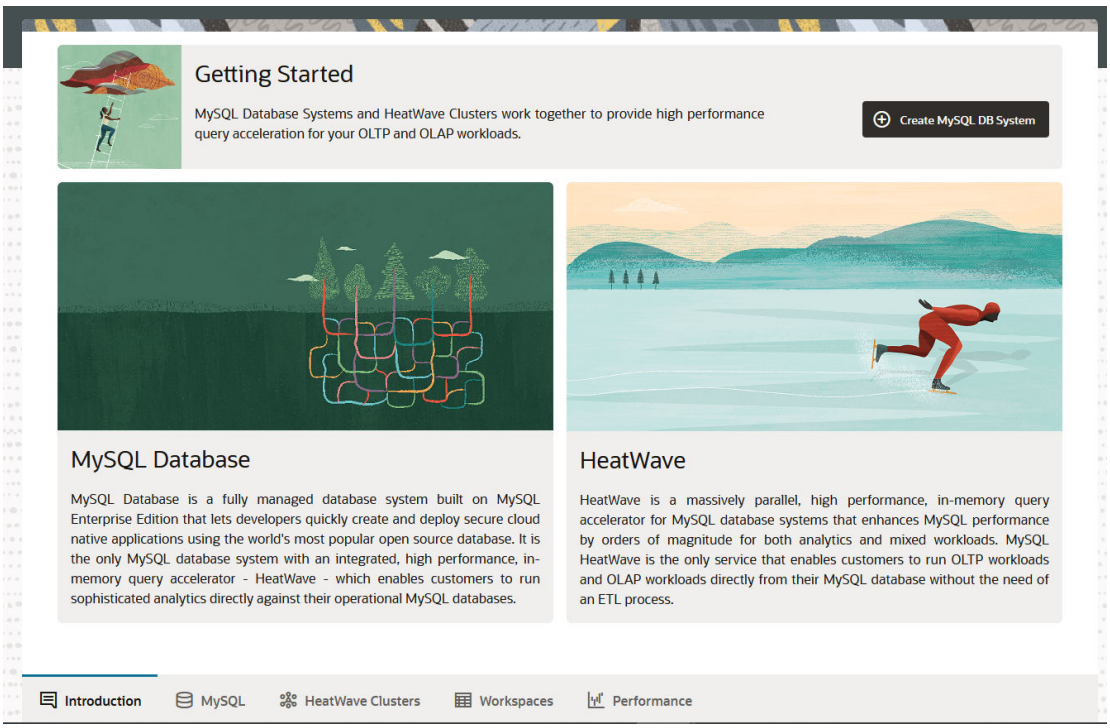
```
*.mysql.com
*.oracle.com
*.oraclecloud.com
*.oracleinfinity.io
oracle.112.2o7.net
consent.trustarc.com
```

The last URL is for OCI Console cookie preferences.

Console Overview

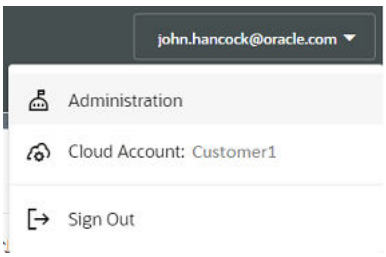
When you sign in to the MySQL HeatWave Console, you are directed to the **Introduction** page.

Figure 2-1 MySQL HeatWave Console Introduction Page



The profile menu on the MySQL HeatWave Console shows the account you are logged in to.

Figure 2-2 Profle Menu on the MySQL HeatWave Console



The navigation tabs allow you to navigate the pages of the MySQL HeatWave Console. The MySQL HeatWave Console pages include:

- **Introduction**

  This is the landing page where users are directed after signing in. It describes MySQL Database and HeatWave. The **Getting Started** region of the page provides a **Create**

MySQL DB System button that directs you to the **DB Systems** tab on the **MySQL** page and launches the **Create MySQL DB System** dialog.

- **MySQL**

  Displays the **MySQL** page, where you can create and manage MySQL DB Systems and backups.

- **HeatWave Clusters**

  Displays the **HeatWave Clusters** page, where you can create and manage HeatWave Clusters.

- **Workspaces**

  Displays the **Workspaces** page, where you can connect to a DB System, manage HeatWave Cluster data, and run DB System and HeatWave queries using the Query Editor.

- **Performance**

  Displays the **Performance** page, where you can monitor HeatWave and MySQL performance metrics.

Signing Out

To sign out of the MySQL HeatWave Console, open the profile menu and click **Sign Out**.

3
DB Systems

This chapter covers how to create and manage MySQL DB Systems.

About DB Systems

A DB System is a logical container for the MySQL instance. It provides an interface enabling management of tasks such as provisioning, backup, monitoring, and so on. It also provides a read/write endpoint enabling you to connect to the MySQL instance using the standard MySQL protocols.

DB System Components

A DB System consists of the following components:

- An Amazon EC2 instance type (with resources defined by the associated shape. See Supported Shapes for more information.)
- Oracle Linux Operating System

> **Note:**
>
> It is not possible to access the operating system. Only the MySQL instance is exposed through the DB System endpoint.

- MySQL Server Enterprise Edition 8.0. For more information, see MySQL Server.
- Virtual Network Interface Card (VNIC) which attaches the DB System to a subnet of the Virtual Private Cloud (VPC). For connections to the DB System, a public endpoint is exposed as a fully-qualified domain name (FQDN). Specific IP addresses in CIDR format can be specified to limit access to the public endpoint.
- Network-attached block storage. MySQL HeatWave on AWS uses Amazon EBS block storage. For more information, see Amazon Elastic Block Store (Amazon EBS) in the *Amazon EC2 User Guide for Linux Instances*.

DB System Storage

Storage size should be determined based on your data size and performance requirements.

For information about increasing the storage available to the DB System, see Increasing DB System Storage.

Supported Shapes

MySQL HeatWave on AWS supports the following MySQL and HeatWave node shapes.

Table 3-1 MySQL Shapes

Shape Name	vCPUs	Memory (GiB)
MySQL.2.16GB	2	16
MySQL.4.32GB	4	32
MySQL.8.64GB	8	64
MySQL.32.256GB	32	256

Table 3-2 HeatWave Node Shapes

Shape Name	Memory (GiB)
HeatWave.16GB	16
HeatWave.256GB	256

> **Note:**
>
> If you intend to use HeatWave AutoML functionality, the `HeatWave.256GB` node shape is recommended when creating a HeatWave Cluster. The `HeatWave.16GB` node shape may not have enough memory for training on large data sets. If you see error messages about this (such as `ML003024`), use the larger shape instead.

MySQL Server

This section describes various aspects of MySQL Server including versioning, upgrades, error logging, unsupported features, and limitations.

Server Versioning

The MySQL Server included in the MySQL HeatWave on AWS uses a versioning system consisting of three numbers, an update version, and the `-cloud` suffix. For example: `8.0.30-u1-cloud`.

- *First number*: The major release number.
- *Second number*: The minor release number. Taken together, the major and minor numbers constitute the release series number.
- *Third number*: The version number within the release series. This is incremented for each new release.
- *uN* : The MySQL Server-specific update number. Fixes and feature development for the cloud version of MySQL Server are delivered according to a different schedule than the on-premise version.
- *cloud*: suffix indicating this version of MySQL Server was built for the cloud, only.

To retrieve the MySQL Server version number, connect to your DB System using a MySQL client, and run `SELECT @@version;`. The following example shows the command and typical output:

```
mysql> SELECT @@version;
+-----------------+
| @@version       |
+-----------------+
| 8.0.30-u1-cloud |
+-----------------+
```

Server Upgrades

MySQL HeatWave on AWS supports the most recent release version of MySQL.

- Minor versions, such as 8.0.*nn*, must be applied manually. See Upgrade MySQL Version for information.
- Upgrade versions, such as 8.0.*nn*-u*n*, are applied automatically and according to the maintenance window defined on the DB System.

For more information about MySQL Server versioning, see Server Versioning.

It is not possible to roll back (downgrade) an upgrade. It is recommended to perform a full backup before upgrading your DB System.

Server Error Logging

MySQL HeatWave on AWS logs MySQL Server diagnostic information such as errors, warnings, and status updates to the Performance Schema `error_log` table. You can use a command-line client such as MySQL Client or MySQL Shell to view this data. For information about connecting to the DB System from a client, see Connecting from a Client. If connecting from MySQL Shell, you must switch to SQL mode by typing `\sql` at the MySQL Shell prompt.

To view logs in the `error_log` table, run the following statement:

```
mysql> SELECT * FROM performance_schema.error_log;
```

You get a response similar to the following, which displays the event timestamp, the thread ID, the event priority, the event error code (if present), subsystem in which the event occurred, and text describing the event:

```
*************************** 1. row ***************************
    LOGGED: 2022-04-07 19:17:03.981201
 THREAD_ID: 0
      PRIO: Warning
ERROR_CODE: MY-011068
 SUBSYSTEM: Server
      DATA: The syntax 'skip_slave_start' is deprecated and will be removed
            in a future release. Please use skip_replica_start instead.
*************************** 2. row ***************************
    LOGGED: 2022-04-07 19:17:03.98388
 THREAD_ID: 0
      PRIO: Note
```

```
ERROR_CODE: MY-010096
 SUBSYSTEM: Server
      DATA: Ignoring --secure-file-priv value as server is running
with
            --initialize(-insecure).
*************************** 3. row ***************************
    LOGGED: 2022-04-07 19:17:03.983921
 THREAD_ID: 0
      PRIO: Note
ERROR_CODE: MY-010949
 SUBSYSTEM: Server
      DATA: Basedir set to /usr/.
....
```

To filter the logs to show only errors, run this statement:

```
mysql> SELECT * FROM performance_schema.error_log
          WHERE PRIO='error';
```

To filter the logs to only show errors from the `RAPID`, HeatWave, subsystem, run this statement:

```
mysql> SELECT * FROM performance_schema.error_log
          WHERE SUBSYSTEM IN ('RAPID');
```

You can select from the following subsystems:

- `Server`
- `InnoDB`
- `RAPID`

To filter on the `RAPID` (HeatWave) subsystem over a 2 hour time interval, run the following command:

```
mysql> SELECT * FROM performance_schema.error_log
          WHERE SUBSYSTEM IN ('RAPID');
```

To retrieve a count of how many instances of a specific error occurred in a single day, run the following command:

```
mysql> SELECT HOUR(LOGGED), count(*)
          FROM performance_schema.error_log
          WHERE ERROR_CODE = 'MY-010914'
          AND LOGGED > DATE_SUB(NOW(),INTERVAL 1 DAY)
          GROUP BY HOUR(LOGGED);
```

To retrieve a count of how many instances of a specific error occurred in a week, run the following command:

```
mysql> SELECT DAY(LOGGED), count(*)
          FROM performance_schema.error_log
          WHERE ERROR_CODE = 'MY-010914'
          AND LOGGED > DATE_SUB(NOW(),INTERVAL 1 WEEK)
          GROUP BY DAY(LOGGED);
```

Unsupported MySQL Server Features

The following features of MySQL Server are currently unsupported in MySQL HeatWave on AWS:

- Authentication plugins other than Native Pluggable Authentication (`mysql_native_password`), SHA-256 Pluggable Authentication (`sha256_password`), and Caching SHA-2 Pluggable Authentication (`caching_sha2_password`).
- Modification of system tables
- Binary log access
- Error Logging to MySQL Server error log. Error logging is available through the Performance Schema. See Server Error Logging.
- Group Replication plugin
- InnoDB Tablespace Encryption
- Setting global variables
- Persisted system variables
- Replication filters
- Semisynchronous replication
- Transportable tablespaces

MySQL Storage Engines

MySQL HeatWave on AWS supports only the InnoDB storage engine.

If you intend to migrate to MySQL HeatWave on AWS, and are not currently using the InnoDB storage engine, your data must be converted to use the `InnoDB` storage engine before the data is imported.

You can manually convert tables to InnoDB using the following `ALTER TABLE` statement:

```
mysql> ALTER TABLE table_name ENGINE=InnoDB;
```

The recommended method for migrating data is to use the MySQL Shell client. MySQL Shell dump utilities provide a `ocimds` option that checks for various incompatibilities, including tables defined with unsupported storage engines. Should incompatibilities exist, the MySQL Shell `compatibility` option can be used to alter MySQL Shell dump files to fix the incompatibilities, including changing the storage engine to InnoDB. For more information, see Importing Data.

Plugins and Components

The following MySQL Server plugins and components are loaded by default. You do not need to install any of these plugins.

- MySQL Enterprise Thread Pool

  Implements a thread pool that increases server performance by efficiently managing statement execution threads for large numbers of client connections. For more information, refer to MySQL Enterprise Thread Pool, in the *MySQL Reference Manual*.

- Connection Control Plugins

  Introduces an increasing delay in server response to connection attempts after a number of consecutive failed attempts. This capability provides a deterrent that slows down brute force attacks against MySQL user accounts. For more information, refer to The Connection-Control Plugins, in the *MySQL Reference Manual*.

- Password Validation Component

  Improves security by requiring account passwords and enabling strength testing of potential passwords. For more information, refer to The Password Validation Component, in the *MySQL Reference Manual*.

- MySQL Enterprise Data Masking and De-Identification

  Helps protect sensitive data from unauthorized uses by hiding and replacing real values with substitutes. For more information, refer to MySQL Enterprise Data Masking and De-Identification, in the *MySQL Reference Manual*.

- MySQL Enterprise Firewall

  Protects your data by monitoring, alerting, and blocking unauthorized database activity. For more information, refer to MySQL Enterprise Firewall, in the *MySQL Reference Manual*.

- MySQL Enterprise Encryption

  Provides built-in, server-side asymmetric encryption, key generation, digital signatures, and other cryptographic features to help protect confidential data using public and private keys. Only the MySQL Enterprise Encryption *component* functions are supported. The MySQL Enterprise Encryption *plugin* functions are deprecated and not supported by MySQL HeatWave on AWS. For more information, refer to MySQL Enterprise Encryption, in the *MySQL Reference Manual*.

MySQL HeatWave on AWS Service Restrictions

The following items are currently not permitted in MySQL HeatWave on AWS:

- Custom TLS certificates.
- Changing the shape associated with a DB System. To change a shape, you must restore a backup of the DB System to a new DB System, and change the shape as part of that process. See Restoring a Backup to a New DB System for more information.

MySQL HeatWave on AWS Service Limitations

MySQL HeatWave on AWS does not support inbound replication.

Inbound replication is only supported with MySQL Database Service on OCI.

Default MySQL Privileges

This section lists the MySQL privileges granted to the MySQL Administrator user on the DB System and those explicitly revoked on the `mysql` and `sys` schemas. Revoked privileges cannot be granted to any MySQL user. For more information, see MySQL Privileges.

- *Global Privileges Granted*
  - `ALTER`
  - `ALTER ROUTINE`
  - `CREATE`
  - `CREATE ROLE`
  - `CREATE ROUTINE`
  - `CREATE TEMPORARY TABLES`
  - `CREATE USER`
  - `CREATE VIEW`
  - `DELETE`
  - `DROP`
  - `DROP ROLE`
  - `EVENT`
  - `EXECUTE`
  - `INDEX`
  - `INSERT`
  - `LOCK TABLES`
  - `PROCESS`
  - `REFERENCES`
  - `REPLICATION CLIENT`
  - `REPLICATION SLAVE`
  - `SELECT`
  - `SHOW DATABASES`
  - `SHOW VIEW`
  - `TRIGGER`
  - `UPDATE`
- *Global Dynamic Privileges Granted*

  - APPLICATION_PASSWORD_ADMIN
  - BACKUP_ADMIN
  - CONNECTION_ADMIN
  - FIREWALL_ADMIN
  - FIREWALL_USER
  - FLUSH_OPTIMIZER_COSTS
  - FLUSH_STATUS
  - FLUSH_TABLES
  - FLUSH_USER_RESOURCES
  - REPLICATION_APPLIER
  - ROLE_ADMIN
  - XA_RECOVER_ADMIN
- *Privileges Revoked from mysql Schema*
  - ALTER
  - ALTER ROUTINE
  - CREATE
  - CREATE ROUTINE
  - CREATE TEMPORARY TABLES
  - CREATE VIEW
  - DELETE
  - DROP
  - EVENT
  - EXECUTE
  - INDEX
  - INSERT
  - LOCK TABLES
  - REFERENCES
  - TRIGGER
  - UPDATE
- *Privileges Revoked from sys Schema*
  - ALTER
  - ALTER ROUTINE
  - CREATE
  - CREATE ROUTINE
  - CREATE TEMPORARY TABLES

  - `CREATE VIEW`
  - `DROP`
  - `EVENT`
  - `INDEX`
  - `LOCK TABLES`
  - `REFERENCES`
  - `TRIGGER`

Reserved User Names

The following user names are reserved and cannot be used for the MySQL Administrator user name:

- `administrator`
- `ociadmin`
- `ocimonitor`
- `ocirpl`
- `mysql.sys`
- `mysql.session`
- `mysql.infoschema`

Creating a DB System

Check that the correct AWS region is shown at the top of the MySQL HeatWave Console. If it is not, see: Manage Regions

Creating a DB System requires administrator credentials, a hardware configuration, an AWS Availability Zone, a MySQL configuration and version, a maintenance window, network settings, backup policy, and a HeatWave Cluster configuration.

- Administrator credentials.

  MySQL HeatWave on AWS grants a specific set of MySQL Server privileges to an administrator. See Default MySQL Privileges. The administrator user name can have up to 32 characters. Certain names are reserved. See Reserved User Names.

  The administrator password must be between 8 and 32 characters, and include at least one number, one uppercase letter, one lowercase letter, and one character from ,.-+*;:_! #%&/()=?><

- Hardware configuration.

  The shape determines the resources that MySQL HeatWave on AWS allocates to the DB System. See Supported Shapes.

> **Note:**
>
> MySQL HeatWave on AWS does not support changes to the MySQL shape. To use a different shape requires a new DB System with the new shape. Restore the data from a backup of the old DB System to the new DB System.

MySQL uses the data storage for all data, logs, and temporary files. Binaries are not stored in this block storage. The maximum storage allocation is 65536 GiB.

> **Note:**
>
> Ensure that the data storage capacity is sufficient for any data import.

- AWS Availability Zone.

  An Availability Zone ID, AZ ID, identifies the physical Availability Zone. See Availability Zone IDs for your AWS resources, in the *AWS RAM User Guide*.

  > **Note:**
  >
  > To maximize performance, place the DB System in the same Availability Zone as the applications that will use the DB System.

- MySQL configuration.

  Use a MySQL configuration to determine the values of global system variables. See Configuration.

  > **Note:**
  >
  > Not all MySQL configurations support HeatWave. Make sure to select a MySQL configuration that does support HeatWave to attach a HeatWave Cluster to the DB System. The default configuration will support HeatWave.

- Maintenance Window.

  The maintenance window is a two-hour period specified in Coordinated Universal Time, UTC. When an update is available, MySQL HeatWave on AWS initiates it during this window. The time required to apply patches and updates may extend beyond the maintenance window and require a DB System restart. For more information, see Maintenance.

  Modify the Maintenance Window later by editing the DB System. See Edit DB System.

- Network settings: allowed client addresses.

Specify the public-facing client IPv4 addresses that are permitted to connect to the DB System endpoint. These must all be public IP addresses that are capable of accessing the Internet, and not private IP addresses. For example:

- The public IP address for the system you are using to set up MySQL HeatWave on AWS.
- The public IP address for a system that an administrator will use to manage the HeatWave Cluster.
- The public IP addresses of application servers where client applications that will use the HeatWave Cluster are running.
- A public IP address range assigned to your organization.

The IP addresses are specified in Classless Inter-Domain Routing, CIDR, format; for example: `1.2.3.4/24`. Multiple addresses in CIDR format can be specified in a semicolon-separated list; for example: `1.2.3.4/24; 1.2.3.4/32`.

CIDR notation permits specifying a range of IP addresses in a single address. A CIDR address looks like a regular IP address but is suffixed with a forward slash followed by a number. The number is a compact representation of the subnet mask; for example, `/24` is equivalent to `255.255.255.0`, where 24 is the number of bits in the binary representation of `255.255.255.0` (`11111111.11111111.11111111.00000000`). `1.2.3.4/24` is equivalent to `1.2.3.4/255.255.255.0`, which permits this IP address range: `1.2.3.0` to `1.2.3.255`.

The following table shows common subnet masks, the equivalent CIDR notation, and the number of IP addresses in the range.

Table 3-3 CIDR Notation Examples

Subnet Mask	CIDR Notation	Number of IP Addresses
255.0.0.0	/8	16777216
255.255.0.0	/16	65536
255.255.255.0	/24	256
255.255.255.128	/25	128
255.255.255.192	/26	64
255.255.255.224	/27	32
255.255.255.240	/28	16
255.255.255.248	/29	8
255.255.255.252	/30	4
255.255.255.254	/31	2
255.255.255.255	/32	1

Specify a combination of CIDR addresses to permit a specific range of IP address. For example, to permit five IP addresses in the range of `192.0.2.0` to `192.0.2.4`, specify these CIDR addresses: `192.0.2.0/30; 192.0.2.4/32`. To permit a single IP address such as `192.0.2.10`, specify `192.0.2.10/32`. More information about CIDR notation including IP range to CIDR calculators can be found by searching online.

- Backup policy.

  Automatic backups are optional. The retention period is from 1 to 35 days for automatic backups, with a default of 7 days. An automatic backup will start during the hour following the **Backup start time**, if this is set. The default time to start an automatic backup is a time between 11:00 and 07:00 UTC, and will be the same time every day.

- HeatWave Cluster configuration.

  The HeatWave Cluster shape determines the resources allocated to each HeatWave node. See Supported Shapes.

  The cluster size is the number of HeatWave nodes, and can be between 1 and 128.

  Because the new DB System does not yet contain any data, MySQL Autopilot cannot estimate the required cluster size, see: Estimating Cluster Size with MySQL Autopilot.

  > **Tip:**
  >
  > To make an estimate after loading the data to the DB System, delete the HeatWave Cluster and create a new one following the instructions in Creating a HeatWave Cluster.

  MySQL HeatWave on AWS does not support changes to the HeatWave Cluster shape. To use a different shape requires removing the existing HeatWave Cluster and creating a new one.

  > **Note:**
  >
  > For HeatWave AutoML functionality, the `HeatWave.256GB` node shape is recommended when creating a HeatWave Cluster. The `HeatWave.16GB` node shape may not have enough memory for training on large data sets. If error messages appear, such as `ML003024`, use the larger shape.

1. In the MySQL HeatWave Console, select the **MySQL** tab.
2. On the **DB Systems** page, click **Create MySQL DB System** to open the **Create MySQL DB System and HeatWave Cluster** dialog.
3. **Basic information**: Provide a name and description for the DB System:
   - **Display Name**: Specify a display name for the DB System or use the generated default name.
   - **Description**: Specify a user friendly description for the DB System.
4. **Administrator credentials**: Specify the administrator credentials.
   - **Username**: Specify a user name for the administrator.
   - **Password**: Specify a password for the administrator.
   - **Confirm Password**: Re-enter the administrator password.
5. **Hardware configuration**: Select a suitable hardware configuration.
   - **Shape**: Select a shape for the DB System.
   - **Data Storage Size (GiB)**: Specify the amount of block storage, in GiB, to allocate to the DB System.
6. **Availability zone**: This determines the physical location of the DB System:
   - **Automatic**: AWS selects the physical AWS Availability Zone.

- **Manual**: Select a physical AWS Availability Zone.

7. **MySQL Configuration**: Select a MySQL configuration.
   - Click **Change** to change the default configuration.
   - Select a configuration from the list.
   - Click **Apply MySQL Configuration**.
8. **Database Version**: Select the MySQL Server version to deploy. The latest MySQL Server version is selected by default.
9. **Maintenance Window**: Select the start time for the maintenance window.
   - Select **Automatic** for MySQL HeatWave on AWS to choose the **Start day** and **Start time**.
   - Select **Manual** to specify the maintenance window **Start day** and **Start time**.
10. **Networking**: Configure network settings:
    - **Allowed Client Addresses**: Specify the public-facing client IPv4 addresses that are permitted to connect to the DB System endpoint.
    - **Port**: The port on which the MySQL server listens. The default is port 3306. Specify a port number between 1024 and 65535.
    - **X Protocol Port**: The X Protocol port on which the MySQL server listens, supported by clients such as MySQL Shell. The default port is 33060. Specify a port number between 1024 and 65535.
11. **Backup policy**: Select the backup retention period, and the start time for automatic backups.
    - Select **Enable Automatic Backups** for MySQL HeatWave on AWS to take backups.
    - Set a retention period between 1 and 35 days.
    - Select **Select Backup Window** to set a preferred time to start a backup.
    - Set the **Backup start time**.
12. Click **Next**.
13. **Provision HeatWave Cluster**: Select whether to provision a HeatWave Cluster.

    This is only available if the chosen MySQL configuration supports HeatWave.
    - **Basic information**: Provide a name and description for the HeatWave Cluster:
      - **Display Name**: Specify a display name for the HeatWave Cluster or use the generated default name.
      - **Description**: Specify a user-friendly description of the HeatWave Cluster.
    - **HeatWave Cluster configuration**: Specify the shape and cluster size.
      - **Shape**: Select the shape to use for the HeatWave Cluster.
      - **Cluster Size**: Specify the size of the cluster between 1 and 128 HeatWave nodes.
14. Click **Create**.

    MySQL HeatWave Console returns to the **DB Systems** page to monitor the state of the operation, which may take some time to complete. The state will change from **Creating** to **Active** when the operation has completed successfully.

Click the name of the DB System to open the **DB System Details** page, and review the **DB System Information**.

Managing a DB System

This section describes how to manage MySQL DB Systems using the Console.

Stopping, Starting, or Restarting a DB System

A DB System starts automatically after it is created, and the **Start** button is disabled. A running DB System shows the **Active** state. For information on DB System states, see MySQL DB System Details.

Restarting a DB System with the **Restart** button shuts down the DB System, and then restarts it immediately.

Stopping a DB System with the **Stop** button stops billing for it. However, billing continues for storage. Billing for the DB System resumes when the DB System starts again.

Start, stop, or restart operations on a DB System also affect an associated HeatWave Cluster. When a DB System restarts, the HeatWave Cluster also restarts, and it reloads data from the DB System. See: Data Load Progress and Status Monitoring.

To start, stop, or restart a DB System:

1. In the MySQL HeatWave Console, select the **MySQL** tab.
2. On the **DB Systems** tab, in the list of DB Systems, find the DB System you want to start, stop, or restart, and do one of the following:
   - Click on the row of the DB System to highlight it, and then click the **Start**, **Stop**, or **Restart** button.
   - Click the name of the DB System to open the **DB System Details** page, and then click the **Start**, **Stop**, or **Restart** button.
3. When you select **Stop** or **Restart**, the **Stop/Restart MySQL DB System** dialog is displayed for you to choose the shutdown type:
   - **Slow** flushes dirty pages and purges undo log pages for older transactions. The shutdown itself can take longer, but the subsequent startup is faster.
   - **Fast** flushes dirty pages before shutting down the DB System. Some flush operations must be performed during next startup, potentially increasing the duration of the startup process.
   - **Immediate** does not flush dirty pages and does not purge any undo log pages. Stops MySQL immediately. Page flushes and log purging will take place during the next startup, increasing the duration of the startup process.

   Select a shutdown type and click the **Stop** or **Restart** button, depending on the action you are taking.

Edit DB System

Edit a DB System, and change the details about the DB System, the maintenance window and the backup policy.

- Maintenance Window.

  The maintenance window is a two-hour period specified in Coordinated Universal Time, UTC. When an update is available, MySQL HeatWave on AWS initiates it during this window. The time required to apply patches and updates may extend beyond the maintenance window and require a DB System restart. For more information, see Maintenance.

- Backup policy.

  Automatic backups are optional. The retention period is from 1 to 35 days for automatic backups, with a default of 7 days. An automatic backup will start during the hour following the **Backup start time**, if this is set. The default time to start an automatic backup is a time between 11:00 and 07:00 UTC, and will be the same time every day.

To edit a DB System:

1. In the MySQL HeatWave Console, select the **MySQL** tab.
2. On the **DB Systems** tab, in the list of DB Systems, find the DB System to edit, and do one of the following:
   - Click on the row of the DB System to highlight it, and choose **Edit DB System** from the **Actions** menu.
   - Click the name of the DB System to open the **DB System Details** page. Click the **Edit DB System** button.
3. **Basic information**: Edit the name and description for the DB System:
   - **Display Name**: Edit the display name for the DB System.
   - **Description**: Edit the user friendly description for the DB System.
4. **Maintenance Window**: Edit the start time for the maintenance window.
   - Select **Automatic** for MySQL HeatWave on AWS to choose the **Start day** and **Start time**.
   - Select **Manual** to specify the maintenance window **Start day** and **Start time**.
5. **Backup policy**: Edit the backup retention period, and the start time for automatic backups.
   - Select **Enable Automatic Backups** for MySQL HeatWave on AWS to take backups.
   - Set a retention period between 1 and 35 days.
   - Select **Select Backup Window** to set a preferred time to start a backup.
   - Set the **Backup start time**.
6. Click **Save** to save the changes.

Update Networking

For the allowed client addresses, specify public-facing client IPv4 addresses that are permitted to connect to the DB System endpoint. Use CIDR format, for example: `1.2.3.4/24`. Specify multiple addresses in CIDR format with a semicolon-separated list, for example: `1.2.3.4/24; 1.2.3.4/32`. For information about the CIDR format, see Creating a DB System.

To update the network for a DB System:

1. In the MySQL HeatWave Console, select the **MySQL** tab.

2. On the **DB Systems** tab, in the list of DB Systems, find the DB System to edit, and do one of the following:
   - Click on the row of the DB System to highlight it, and choose **Update Networking** from the **Actions** menu.
   - Click the name of the DB System to open the **DB System Details** page. Click the **Update Networking** button.
3. Edit the **Allowed Client Addresses**.
4. Click **Save** to save the changes.

Update MySQL Configuration

Update the MySQL configuration for a DB System. This will change the user configurable system variables for the DB System to the values in the new configuration. See User Configurable System Variables.

> **Note:**
>
> Any change to the configuration for a DB System cannot change the system initialization variables. See System Initialization Variables.

The MySQL HeatWave Console only shows alternative configurations that support the DB System shape. If a DB System has a HeatWave Cluster attached, MySQL HeatWave Console only shows configurations that support HeatWave. To change the configuration to one that does not support HeatWave, first remove the HeatWave Cluster.

If the new configuration changes any system variables that are not dynamic, then the MySQL database process, mysqld, will require a restart. If the DB System has an attached HeatWave Cluster, it will restart and reload data from the DB System.

To update the MySQL configuration for a DB System:

1. In the MySQL HeatWave Console, select the **MySQL** tab.
2. On the **DB Systems** tab, in the list of DB Systems, find the DB System to edit, and do one of the following:
   - Click on the row of the DB System to highlight it, and choose **Update MySQL Configuration** from the **Actions** menu.
   - Click the name of the DB System to open the **DB System Details** page. Click the **Update MySQL Configuration** button.
3. Click **Change**.
4. Select the preferred MySQL configuration, and click **Apply MySQL Configuration**.
5. Click **Update** to save the changes.

   If any problems occur updating the configuration, the DB System will roll back to the previous configuration.

Upgrade MySQL Version

> **Note:**
>
> It is not possible to roll back a DB System upgrade. It is recommended to perform a full backup of the DB System before upgrading. See Creating a Backup.

See Server Versioning. The available options will depend upon the current version. These include:

- **Do not change**

  This option is always available, and shows the current version. For instance: **8.0.27-u2-cloud**.

- **Update**

  This option is only available if the current version is still supported. Updates happen automatically during the Maintenance Window, see Edit DB System. This option provides a manual update.

- ***A more recent MySQL version***

  This option is only available if a more recent version, or versions, is available. It shows the version number, for instance, **8.0.28** or **8.0.29**.

To upgrade the MySQL version:

1. In the MySQL HeatWave Console, select the **MySQL** tab.
2. On the **DB Systems** tab, in the list of DB Systems, find the DB System to edit, and do one of the following:
   - Click on the row of the DB System to highlight it, and choose **Upgrade MySQL Version** from the **Actions** menu.
   - Click the name of the DB System to open the **DB System Details** page. Click the **Upgrade MySQL Version** button.
3. Choose one of the options:
   - **Do not change**: Choose this option to ignore any update or upgrade option.
   - **Update**: Choose this option to update the current version.
   - ***A more recent MySQL version*** : Choose this option to upgrade to a more recent version.

Increasing DB System Storage

You can increase the DB System storage by updating the data storage size of an active DB System, by backup and restore, or by export and import.

Increasing Storage by Updating the Data Storage Size of an Active DB System

Use the Console to increase the data storage size of an active DB System. If the DB System is active and healthy, updating the data storage size of the DB System does not restart the DB System, and you can continue to query it while the storage is being increased.

You can increase the data storage size once every six hours. It is recommended to create a backup of the DB System before updating the data storage size. See Creating a Backup.

1. In the MySQL HeatWave Console, select the **MySQL** tab.
2. On the **DB Systems** tab, in the list of DB Systems, find the DB System for which you want to update the data storage size, and do either of the following:
   - Click on the row of the DB System to highlight it, click **Actions**, and then click **Update Data Storage Size**.
   - Click the name of the DB system to open the **DB System Details** page. Click **Actions**, and then click **Update Data Storage Size**.
3. In the **Update Data Storage Size** dialog, enter the required storage size in GiB. You can only increase the storage size, and the maximum storage size is 65536.
4. Click **Update**.

Increasing Storage by Backup and Restore

1. Create a backup of the DB System. See Creating a Backup.
2. Create a new DB System from the backup, and define a larger storage in the new DB System. See Restoring a Backup to a New DB System.

Increasing Storage by Export and Import

1. Export the data from your current DB System using MySQL Shell. See Exporting Data with MySQL Shell .
2. Use the MySQL HeatWave Console to create a new DB System with a larger storage size. See Creating a DB System.
3. Import the data into the new DB System using MySQL Shell. See Importing Data.

Deleting a DB System

Deleting a DB System permanently deletes it, along with all of the data in the database. Ensure all the data is appropriately backed up before deleting a DB System (see Backups).

To delete a DB System:

1. In the MySQL HeatWave Console, select the **MySQL** tab.
2. On the **DB Systems** tab, in the list of DB Systems, find the DB System you want to delete, and do one of the following:
   - Click on the row of the DB System to highlight it, and choose the **Delete** action from the **Actions** menu.
   - Click the name of the DB System to open the **DB System Details** page. Click the **Delete** button.

   The **Delete MySQL DB System** dialog is displayed.
3. Click **Delete MySQL DB System** to go ahead with the deletion.

Viewing DB System Details

To view DB System details:

1. In the MySQL HeatWave Console, select the **MySQL** tab.
2. On the **DB Systems** tab, in the list of DB Systems, find the DB System you want to view details for, and do one of the following:
   - Click on the row of the DB System to highlight it. The DB System details appear below the list of DB Systems.
   - Click the name of the DB System to open the **MySQL DB System Details** page for the selected DB System.

For descriptions of DB System details, see MySQL DB System Details.

MySQL DB System Details

The **MySQL DB System Details** page is divided into the following sections:

Table 3-4 MySQL DB System Details Page

Name	Description
Summary	DB System summary details. See Table 3-5.
DB System Information	DB System general information, configuration, endpoint, and HeatWave Cluster details. See Table 3-6.
Backups	DB System backup details. See Table 3-7.

Table 3-5 MySQL DB System Summary Details

Field	Description
State	The state of the DB System. • **CREATING**: Resources are being reserved for the DB System, the system is booting, and the initial database is being created. Creating can take several minutes. The system is not ready to use yet. • **ACTIVE**: The DB System was successfully created and is ready to be used. • **STARTING**: The DB System is being started. • **STOPPING**: The DB System is being stopped. • **RESTARTING**: The DB System is being restarted. • **INACTIVE**: The DB System is powered off. • **UPDATING**: The DB System is in the process of being updated. • **UPGRADING**: The DB System is in the process of being upgraded. • **DELETING**: The DB System is being deleted. • **DELETED**: The DB System has been deleted and is no longer available. • **FAILED**: An error condition prevented the creation or continued operation of the DB System. • **STARTING_ERROR**: An attempt to start the DB System failed. • **STOPPING_ERROR**: An attempt to stop the DB System failed. • **RESTARTING_ERROR**: An attempt to restart the DB System failed. • **UPDATING_ERROR**: An attempt to update the DB System failed. • **UPGRADING_ERROR**: An attempt to upgrade the DB System failed. • **DELETING_ERROR**: An attempt to delete the DB System failed.
Host Name	The host name of the DB System. The host name is a fully qualified domain name (FQDN).
Resource ID	The unique resource identifier assigned to the DB System when it is created.
Shape	The resource template for the MySQL instance. For information about shapes, see Supported Shapes.

Table 3-6 DB System Information Details

Field	Description
General Information	• **Description**: The user-defined description of the DB System. • **Created**: The date and time the DB System was created. • **Last Updated**: The date and time the DB System was last updated.
DB System Configuration	• **Storage Size**: The amount of storage available to the DB System. • **MySQL Version**: The MySQL Server version used by the DB System. • **Maintenance Window**: When updates are available, they are initiated during the maintenance window, which is a two-hour period starting at this time. The time required to apply patches and updates may extend beyond the maintenance window and require DB System restarts. For more information, see Maintenance.
Endpoint	• **Availability Zone**: The physical Availability Zone where the DB System resides. • **MySQL Port**: The port on which the MySQL Server listens. • **MySQL Port X**: The port on which the MySQL Server listens for clients using MySQL's X Protocol. • **Allowed Client Addresses**: The public-facing client IPv4 addresses that are permitted to connect to the DB System endpoint, specified in CIDR format. For information about how IP addresses are specified in CIDR format, see Creating a DB System.

Table 3-6 (Cont.) DB System Information Details

Field	Description
HeatWave Cluster	• **State**: The state of the HeatWave Cluster. – **CREATING**: Resources are being reserved for the HeatWave Cluster, and the cluster is being created. Creating can take several minutes. The system is not ready to use yet. – **ACTIVE**: The HeatWave Cluster was successfully created and is ready to be used. – **STARTING**: The HeatWave Cluster is being started. – **STOPPING**: The HeatWave Cluster is being stopped. – **RESTARTING**: The HeatWave Cluster is being restarted. – **INACTIVE**: The HeatWave Cluster is powered off. – **UPDATING**: The HeatWave Cluster is in the process of being updated. – **UPGRADING**: The HeatWave Cluster is in the process of being upgraded. – **DELETING**: The HeatWave Cluster is being deleted. – **DELETED**: The HeatWave Cluster has been deleted and is no longer available. – **FAILED**: An error condition prevented the creation or continued operation of the HeatWave Cluster. – **STARTING_ERROR**: An attempt to start the HeatWave Cluster failed. – **STOPPING_ERROR**: An attempt to stop the HeatWave Cluster failed. – **RESTARTING_ERROR**: An attempt to restart the HeatWave Cluster failed. – **UPDATING_ERROR**: An attempt to update the HeatWave Cluster failed. – **UPGRADING_ERROR**: An attempt to upgrade the HeatWave Cluster failed. – **DELETING_ERROR**: An attempt to delete the HeatWave Cluster failed. • **Name**: The name of the HeatWave Cluster. • **Cluster Size**: The number of HeatWave Cluster nodes.

Table 3-7 MySQL DB System Backup Details

Field	Description
Name	The name of the backup

Table 3-7 (Cont.) MySQL DB System Backup Details

Field	Description
State	The state of the backup • **CREATING**: Resources are being reserved for the DB System backup and the backup is being created. Creating can take several minutes. The backup is not ready to use yet. • **ACTIVE**: The backup was successfully created. • **UPDATING**: The backup is in the process of being updated. • **INACTIVE**: The backup is not available. • **DELETING**: The backup is being deleted. • **DELETED**: The backup has been deleted and is no longer available. • **FAILED**: An error condition prevented the creation or continued availability of the backup.
MySQL DB System	The name of the MySQL DB System used to create the backup.
Size (GiB)	The size of the backup - note that a backup is the size of the whole storage allocation for the DB System from which it was created, even if the volume does not contain that much data.
Created	The date and time the backup was created.

4
HeatWave Clusters

This chapter describes how to create and manage HeatWave Clusters.

A HeatWave Cluster consists of a MySQL DB System and one or more HeatWave nodes. The MySQL DB System includes a HeatWave plugin that is responsible for cluster management, query scheduling, and returning query results to the MySQL DB System. HeatWave nodes store data in memory and process queries.

When a HeatWave Cluster is enabled and data is loaded, queries that meet certain prerequisites are automatically offloaded from the MySQL DB System to the HeatWave Cluster for accelerated processing.

Creating a HeatWave Cluster

A HeatWave Cluster must be associated with an active DB System, and a DB System can have only one HeatWave Cluster. Before you create a HeatWave Cluster, ensure that you have created a DB System and that the DB System does not already have a HeatWave Cluster. With MySQL HeatWave on AWS, you create a DB System and HeatWave nodes at the same time, but you can delete the HeatWave Cluster afterwards if you want to re-create it. See Creating a DB System for instructions.

When you create a HeatWave Cluster, you are presented with an option to estimate the required HeatWave Cluster size based on data that is loaded on your DB System (see Estimating Cluster Size with MySQL Autopilot). If you have not loaded data into your DB System, and you want to estimate the optimal HeatWave Cluster size, load data into the DB System before you create a HeatWave Cluster. See Importing Data.

> **Note:**
>
> Changing the number of nodes in an existing HeatWave Cluster is not supported. If you require a larger or smaller cluster, you must delete the existing cluster and create a new one with the desired number of nodes.

To create a HeatWave Cluster, do the following:

1. In the MySQL HeatWave Console, select the **HeatWave Clusters** tab.
2. Click **Create HeatWave Cluster**. The **Create HeatWave Cluster** dialog is displayed.
3. On the **Create HeatWave Cluster** dialog, provide the following information:
   - **Basic Information**
     - **Display Name**: Specify a user-friendly display name for the HeatWave Cluster.
     - **Description**: Specify a user-friendly description of the HeatWave Cluster.
   - **DB System Name**: Select a DB System from the drop down menu.
   - **HeatWave Cluster Configuration**

- **Shape**: Select a HeatWave node shape. For information about supported shapes, see Supported Shapes.

  > **Note:**
  >
  > If you intend to use HeatWave AutoML functionality, the `HeatWave.256GB` node shape is recommended when creating a HeatWave Cluster. The `HeatWave.16GB` node shape may not have enough memory for training on large data sets. If you see error messages about this (such as `ML003024`), use the larger shape instead.

- **Cluster Size**: The number of HeatWave nodes to create. Enter a number between 1 and 128. Optionally, click **Estimate Cluster Size** to estimate the required cluster size using MySQL Autopilot. For instructions, see Estimating Cluster Size with MySQL Autopilot.

4. Click **Create** to create the HeatWave Cluster.

You are returned to the **HeatWave Clusters** page where you can monitor the state of the operation, which may take some time to complete. The state changes from **Creating** to **Active** when the operation has completed successfully.

Estimating Cluster Size with MySQL Autopilot

This topic describes how to estimate the optimal HeatWave Cluster size for your data.

A cluster size estimate is generated using MySQL Autopilot machine learning techniques. MySQL Autopilot analyzes the data on your MySQL DB System and recommends a cluster size. If you have not loaded data into your DB System, and you want to estimate the optimal HeatWave Cluster size, load data into the DB System before you create a HeatWave Cluster. See Importing Data.

Prerequisites:

- The data you intend to load into the HeatWave Cluster must be available on the DB System.
- Optionally, log into your DB System and run `ANALYZE TABLE` on tables you intend to load into the HeatWave Cluster. Estimates should generally be valid without running `ANALYZE TABLE` , but running `ANALYZE TABLE` ensures that estimates are as accurate as possible.

To estimate a cluster size:

1. Click **Estimate Cluster Size**.

   The **Estimate Cluster Size with Autopilot** dialog is displayed.

2. Select the schemas and tables you want to include in the estimate. Schemas are displayed in the **Schemas** pane. Tables belonging to the selected schema appear in the **Tables from selected schemas** pane.

   When schemas and tables are selected, the **Summary** details are adjusted automatically.

   The **Schemas** pane provides the following information:

   - **Name**: The schema name.

- **HeatWave Cluster Memory Usage (GiB)**: The estimated amount of HeatWave Cluster memory used by the schema.
- **Tables Selected**: The number of tables selected expressed as a fraction of the total number of tables.
- **Warnings**: The number of table warnings.

The **Tables from selected schemas** pane provides the following information:

- **Name**: The table name.
- **Warnings**: The number of table warnings. For a description of table warnings, see Cluster Size Estimate Table Warnings.
- **Memory Size Estimate (GiB)**: The estimated amount of HeatWave Cluster memory required for the table.
- **Rows Estimate**: The estimated number of table rows.

3. Review the **Summary** details, which include memory required by the schemas and tables selected, memory provided per node, HeatWave Cluster nodes required, and memory provided by the cluster.
4. To apply the cluster size estimate, click **Apply Cluster Size Estimate**.

   You are returned to the **Create HeatWave Cluster** dialog where the estimate is applied to the **Cluster Size** field.

Cluster Size Estimate Table Warnings

This topic describes table warnings that may appear in the **Tables from selected schemas** pane, in the **Estimate Cluster Size with MySQL Autopilot** dialog.

Table 4-1 Cluster Size Table Warnings

Table Status Issue	Description
`TOO MANY COLUMNS TO LOAD`	The table has too many columns. The column limit is 1017.
`ALL COLUMNS MARKED AS NOT SECONDARY`	There are no columns to load. All table columns are defined as `NOT SECONDARY`. Columns defined as `NOT SECONDARY` are excluded from the estimate. For more information, see Excluding Table Columns, in the *MySQL HeatWave User Guide*.
`CONTAINS VARLEN COLUMN WITH >65532 BYTES`	A VARLEN column exceeds the 65532 byte limit. For more information on `VARLEN`, see Variable-length Encoding in the *MySQL HeatWave User Guide*
`ESTIMATION COULD NOT BE CALCULATED`	The estimate could not be calculated. For example, a table estimate may not be available if statistics for `VARLEN` columns are unavailable.
`UNABLE TO LOAD TABLE WITHOUT PRIMARY KEY`	A table must be defined with a primary key before it can be loaded into HeatWave.

Managing a HeatWave Cluster

Starting, Stopping, or Restarting a HeatWave Cluster

A HeatWave Cluster starts automatically after it is created, and the **Start** button is disabled. A running HeatWave Cluster shows the **Active** state. For information on HeatWave Cluster states, see HeatWave Cluster Details.

Restarting a HeatWave Cluster with the **Restart** button shuts down the HeatWave Cluster and then restarts it immediately. Stopping a HeatWave Cluster with the **Stop** button stops it without restarting.

When a HeatWave Cluster is stopped through a stop or restart action, the data loaded in HeatWave Cluster memory is unloaded. When a HeatWave Cluster restarts, data is reloaded from AWS S3, and includes any changes that occur on the DB System while the HeatWave Cluster is offline.

Start, stop, or restart actions on a HeatWave Cluster have no effect on the DB System with which the HeatWave Cluster is associated.

To start, stop, or restart a HeatWave Cluster:

1. In the MySQL HeatWave Console, select the **HeatWave Clusters** tab.
2. In the list of HeatWave Clusters, find the HeatWave Cluster you want to start, stop, or restart, and do one of the following:
   - Click on the row of the HeatWave Cluster to highlight it, and then click the **Start**, **Stop**, or **Restart** button.
   - Click the name of the HeatWave Cluster to open the **HeatWave Cluster Details** page, and then click the **Start**, **Stop**, or **Restart** button.

Editing a HeatWave Cluster

Use the Console to edit the display name and description of a HeatWave cluster.

1. In the MySQL HeatWave Console, click the **HeatWave Clusters** tab.
2. In the list of HeatWave Clusters, find the HeatWave Cluster you want to edit, and do one of the following:
   - Click the row of the HeatWave Cluster to highlight it, and click **Edit HeatWave Cluster** from the **Actions** menu.
   - Click the name of the HeatWave Cluster to open the **Details** page and click **Edit HeatWave Cluster** from the **Actions** menu.
3. On the **Edit HeatWave Cluster** panel, provide the following information:
   - **Display Name**: Specify a name of the HeatWave Cluster.
   - **Description**: Specify a description of the HeatWave Cluster.
4. Click **Save**.

The details of the selected HeatWave Cluster is updated.

Deleting a HeatWave Cluster

Deleting a HeatWave Cluster removes the HeatWave Cluster nodes permanently. The DB System with which the HeatWave Cluster is associated is unaffected.

To delete a HeatWave Cluster:

1. In the MySQL HeatWave Console, select the **HeatWave Clusters** tab.
2. In the list of HeatWave Clusters, find the HeatWave Cluster you want to delete, and do one of the following:
   - Click on the row of the HeatWave Cluster to highlight it, and choose the **Delete** action from the **Actions** menu.
   - Click the name of the HeatWave Cluster to open the **HeatWave Cluster Details** page. Click the **Delete** button.

   The **Delete HeatWave Cluster** dialog is displayed.
3. Click **Delete HeatWave Cluster** to go ahead with the deletion.

Viewing HeatWave Cluster Details

To view HeatWave Cluster Details:

1. In the MySQL HeatWave Console, select the **HeatWave Clusters** tab.
2. In the list of HeatWave Clusters, find the HeatWave Cluster you want to view details for, and do one of the following:
   - Click on the row of the HeatWave Cluster to highlight it. The HeatWave Cluster details appear below the list of HeatWave Clusters. Click on the arrow icon to display or hide the details.
   - Click the name of the HeatWave Cluster to open the **HeatWave Cluster Details** page for the selected HeatWave Cluster.

For descriptions of HeatWave Cluster details, see HeatWave Cluster Details.

HeatWave Cluster Details

The **HeatWave Cluster Details** page is divided into the following sections:

Table 4-2 HeatWave Cluster Details Page

Name	Description
Summary	HeatWave Cluster summary details. See Table 4-3.
General Information	General HeatWave Cluster details. See Table 4-4.
MySQL DB System	MySQL DB System details. See Table 4-5.
HeatWave Nodes	HeatWave node details. See Table 4-6.

Table 4-3 HeatWave Cluster Summary Details

Field	Description
State	The state of the HeatWave Cluster. • **CREATING**: Resources are being reserved for the HeatWave Cluster, and the cluster is being created. Creating can take several minutes. The system is not ready to use yet. • **ACTIVE**: The HeatWave Cluster was successfully created and is ready to be used. • **STARTING**: The HeatWave Cluster is being started. • **STOPPING**: The HeatWave Cluster is being stopped. • **RESTARTING**: The HeatWave Cluster is being restarted. • **INACTIVE**: The HeatWave Cluster is powered off. • **UPDATING**: The HeatWave Cluster is in the process of being updated. • **UPGRADING**: The HeatWave Cluster is in the process of being upgraded. • **DELETING**: The HeatWave Cluster is being deleted. • **DELETED**: The HeatWave Cluster has been deleted and is no longer available. • **FAILED**: An error condition prevented the creation or continued operation of the HeatWave Cluster. • **STARTING_ERROR**: An attempt to start the HeatWave Cluster failed. • **STOPPING_ERROR**: An attempt to stop the HeatWave Cluster failed. • **RESTARTING_ERROR**: An attempt to restart the HeatWave Cluster failed. • **UPDATING_ERROR**: An attempt to update the HeatWave Cluster failed. • **UPGRADING_ERROR**: An attempt to upgrade the HeatWave Cluster failed. • **DELETING_ERROR**: An attempt to delete the HeatWave Cluster failed.
Cluster Size	The number of HeatWave Cluster nodes.
Resource ID	The unique resource identifier assigned to the HeatWave Cluster when it is created.
Shape	The shape used for HeatWave Cluster nodes. See Supported Shapes.

Table 4-4 General Information

Field	Description
Description	A user-specified description of the HeatWave Cluster.
Created	The date and time the HeatWave Cluster was created.
Last Updated	The date and time the HeatWave Cluster was last updated.

Table 4-5 MySQL DB System Details

Field	Description
Name	The name of the DB System that the HeatWave Cluster is associated with
State	The state of the DB System that the HeatWave Cluster is associated with. • **CREATING**: Resources are being reserved for the DB System, the system is booting, and the initial database is being created. Creating can take several minutes. The system is not ready to use yet. • **ACTIVE**: The DB System was successfully created and is ready to be used. • **STARTING**: The DB System is being started. • **STOPPING**: The DB System is being stopped. • **RESTARTING**: The DB System is being restarted. • **INACTIVE**: The DB System is powered off. • **UPDATING**: The DB System is in the process of being updated. • **UPGRADING**: The DB System is in the process of being upgraded. • **DELETING**: The DB System is being deleted. • **DELETED**: The DB System has been deleted and is no longer available. • **FAILED**: An error condition prevented the creation or continued operation of the DB System. • **STARTING_ERROR**: An attempt to start the DB System failed. • **STOPPING_ERROR**: An attempt to stop the DB System failed. • **RESTARTING_ERROR**: An attempt to restart the DB System failed. • **UPDATING_ERROR**: An attempt to update the DB System failed. • **UPGRADING_ERROR**: An attempt to upgrade the DB System failed. • **DELETING_ERROR**: An attempt to delete the DB System failed.

Table 4-5 (Cont.) MySQL DB System Details

Field	Description
Description	The user-defined description of the DB System

Table 4-6 HeatWave node Details

Field	Description
Node ID	The HeatWave node ID
State	The state of the HeatWave node. • **CREATING**: Resources are being reserved for the HeatWave node, and the cluster is being created. Creating can take several minutes. The system is not ready to use yet. • **ACTIVE**: The HeatWave node was successfully created and is ready to be used. • **STARTING**: The HeatWave node is being started. • **STOPPING**: The HeatWave node is being stopped. • **RESTARTING**: The HeatWave node is being restarted. • **INACTIVE**: The HeatWave node is powered off. • **UPDATING**: The HeatWave node is in the process of being updated. • **UPGRADING**: The HeatWave node is in the process of being upgraded. • **DELETING**: The HeatWave node is being deleted. • **DELETED**: The HeatWave node has been deleted and is no longer available. • **FAILED**: An error condition prevented the creation or continued operation of the HeatWave node. • **STARTING_ERROR**: An attempt to start the HeatWave node failed. • **STOPPING_ERROR**: An attempt to stop the HeatWave node failed. • **RESTARTING_ERROR**: An attempt to restart the HeatWave node failed. • **UPDATING_ERROR**: An attempt to update the HeatWave node failed. • **UPGRADING_ERROR**: An attempt to upgrade the HeatWave node failed. • **DELETING_ERROR**: An attempt to delete the HeatWave node failed.

HeatWave Cluster Failure and Recovery

HeatWave triggers the recovery process whenever the HeatWave cluster is in an unhealthy state.

The HeatWave cluster status is being monitored regularly and if the cluster does not get a response from the cluster node, the cluster becomes unhealthy due to a node failure and error recovery is triggered.

During the recovery process, HeatWave automatically attempts to bring the node online, and reload data that was previously loaded. HeatWave reloads data from the HeatWave Storage layer, which is created when you enable the HeatWave cluster for the first time. To facilitate recovery, data is persisted to AWS S3 when data is loaded into HeatWave and when data changes is propagated from the DB system to HeatWave. Loading data from AWS S3 is faster because the data does not need to be converted to the HeatWave storage format, as is required when loading data from the DB system.

When you unload a table, the data is removed from HeatWave, and in a background operation, it is removed from AWS S3 too.

5
Connecting to a DB System

Clients and applications interact with a HeatWave Cluster by connecting to the MySQL DB System. This chapter describes how to connect to the MySQL DB System from the MySQL HeatWave Console and from MySQL clients including MySQL Shell, MySQL Command-Line-Client, and MySQL Workbench. For connecting from applications, refer to the MySQL Connectors documentation. See MySQL Connectors.

Connecting from the Console

Connecting to a DB System from the MySQL HeatWave Console requires a Oracle Cloud Account user name and password for signing into the MySQL HeatWave Console, and a MySQL user account for connecting to the DB System.

Use the MySQL Administrator user specified when creating the DB System, see Creating a DB System, or use a MySQL user account created on the DB System using `CREATE USER`. Otherwise, ask the MySQL Administrator to create an account.

Use the **Workspaces** page to connect to a DB System, and then manage HeatWave Cluster data and use the **Query Editor** to run DB System and HeatWave queries. For more information, see Manage HeatWave Data with Workspaces, and Running Queries.

To connect to a DB System from the MySQL HeatWave Console:

1. Sign-in to the MySQL HeatWave Console. For instructions, see Signing In.
2. Select the **Workspaces** tab in the MySQL HeatWave Console, and then click **Connect to MySQL DB System**.
3. Select a DB System from the drop-down menu.
4. Enter a MySQL user name and password for the DB System.
5. Click **Connect**.

To disconnect, click **Disconnect**.

Connecting from a Client

Connecting to a DB System from a MySQL client requires a MySQL user account on the MySQL DB System. You can use the MySQL Administrator user that you specified when creating the DB System (see Creating a DB System) or a MySQL user account created on the DB System using `CREATE USER`. If you are not the MySQL Administrator and you do not have a MySQL user account, have your MySQL Administrator create one for you.

You cannot connect from a MySQL client to a DB System using the Oracle Cloud Account user name and password used to access the MySQL HeatWave Console.

For MySQL client connections to the DB System, a public endpoint is exposed as a fully qualified domain name (the *host name* of the DB System). The host name is found on the **MySQL DB System Details** page. See Viewing DB System Details.

The MySQL HeatWave on AWS Administrator may have restricted access to your DB System to certain public-facing IPv4 client IP addresses or address ranges. Allowed client addresses are specified in CIDR format and are found on the **MySQL DB System Details** page. See Viewing DB System Details. To edit allowed client addresses, see Edit DB System. Specifying IP addresses in CIDR format is discussed in Creating a DB System.

If you are connecting from a MySQL client that resides in a private subnet, you have the option of connecting to a DB System through a public Network Address Translation (NAT) gateway, which permits clients and applications in a private subnet to access services outside of the private subnet while preventing external services from initiating inbound connections. When establishing a NAT gateway, ensure that the elastic IP address of the NAT gateway is added as an *Allowed Client Address*, as described above. For example, if your NAT gateway elastic IP address is 1.2.3.4, edit your DB System to add `1.2.3.4/32` (the address in CIDR notation) to your DB System's **Allowed Client Addresses**. See Edit DB System. For more information about NAT gateways, refer to NAT Gateways, in the *Amazon VPC User Guide*.

MySQL HeatWave on AWS supports TLSv1.2 and requires that all connections are encrypted. For added security, you can download a signed certificate bundle and enable host name identity verification. For more information, see Enabling Host Name Identity Verification.

To reduce network costs and avoid potential latency issues and bandwidth fluctuations, it is recommended that connecting clients reside in the same *Region* as the MySQL HeatWave on AWS instance. Latency and bandwidth fluctuations experienced by connections from outside the MySQL HeatWave on AWS *Region* are outside of the control MySQL HeatWave on AWS service managers. Connecting from the same *Availability Zone* is also recommended to avoid potential latency issues.

Connecting with MySQL Shell

This topic describes how to connect to a MySQL DB System using MySQL Shell.

Prerequisites:

- A machine or compute instance with internet connectivity for connecting to the MySQL DB System.
- A MySQL Shell command-line utility installed on the machine or compute instance. For installation instructions, refer to Installing MySQL Shell.
- A MySQL user account on the MySQL DB System to connect with. You can use the MySQL Administrator user that you specified when creating the DB System or a MySQL user account created on the DB System using `CREATE USER` .
- A public-facing IP address for your machine or compute instance that is permitted to connect to the DB System. **Allowed Client Addresses** information is available on the **MySQL DB System Details** page. See Viewing DB System Details.
- The host name of the MySQL DB System as defined on the **MySQL DB System Details** page. See Viewing DB System Details.

To connect to a DB System:

1. Start MySQL Shell and connect to the MySQL DB System using the following command:

```
$> mysqlsh Username@HostNameOfMySQLDBSystem

Please provide the password for 'Username@HostNameOfMySQLDBSystem':
```

This command starts a global session. MySQL Shell attempts to connect to port 33060 by default and, if that port is not available, falls back to port 3306.

The connection is made and message similar to the following is displayed:

```
                                MySQL Shell 8.0.30

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

Type '\help' or '\?' for help; '\quit' to exit.
Creating a session to 'Username@HostNameOfMySQLDBSystem'
Fetching schema names for autocompletion... Press ^C to stop.
Your MySQL connection id is 38 (X protocol)
Server version: 8.0.30-u1-cloud MySQL Enterprise - Cloud
No default schema selected; type \use <schema> to set one.
MySQL  HostNameOfMySQLDBSystem.dbsystem JS >
```

Connecting with MySQL Command-Line Client

This topic describes how to connect to a MySQL DB System using a MySQL Command-Line Client.

Prerequisites:

- A machine or compute instance with internet connectivity for connecting to the MySQL DB System.
- A MySQL Command-Line Client installed on the machine or compute instance.
- A MySQL user account on the MySQL DB System to connect with. You can use the MySQL Administrator user that you specified when creating the DB System or a MySQL user account created on the DB System using `CREATE USER` .
- A public-facing IP address for your machine or compute instance that is permitted to connect to the DB System. **Allowed Client Addresses** information is available on the **MySQL DB System Details** page. See Viewing DB System Details.
- The host name of the MySQL DB System as defined on the **MySQL DB System Details** page. See Viewing DB System Details.

To connect to a DB System:

1. Start MySQL client and connect to the MySQL DB System using the following command:

```
$> mysql --host HostNameOfMySQLDBSystem -u Username -p
```

The connection is made and a message similar to the following is displayed:

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 40
Server version: 8.0.30-u1-cloud MySQL Enterprise - Cloud

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input
statement.

mysql>
```

Connecting with MySQL Workbench

This topic describes how to connect to a MySQL DB System with MySQL Workbench.

Prerequisites:

- A machine or compute instance with internet connectivity for connecting to the MySQL DB System.
- A MySQL Workbench client installed on your machine or compute instance. For instructions, refer to the MySQL Workbench Reference Manual.
- A MySQL user account on the MySQL DB System to connect with. You can use the MySQL Administrator user that you specified when creating the DB System or a MySQL user account created on the DB System using `CREATE USER` .
- A public-facing IP address for your machine or compute instance that is permitted to connect to the DB System. **Allowed Client Addresses** information is available on the **MySQL DB System Details** page. See Viewing DB System Details.
- The host name of the MySQL DB System as defined on the **MySQL DB System Details** page. See Viewing DB System Details.

To connect to a DB System from MySQL Workbench:

1. In MySQL Workbench, select **Database** > **Connect to Database** to open the **Connect to Database** dialog.
2. Enter the connection details.
   - **Connection Method**: The connection method. Select **Standard (TCP/IP)**.
   - **Hostname**: The host name of the MySQL DB System
   - **Port**: The port address that the MySQL DB System is listening on. Use the default `3306` port.
   - **Username**: The user name of the MySQL user that will connect to the DB System
   - **Password**: The password of the MySQL user
   - **Default Schema**: Optionally, specify a default schema. Leave blank to select it later.

3. Click **OK** to establish the connection.

MySQL Connectors

You can use MySQL Connectors to connect your application to a DB System. The following list provides links to the MySQL Connectors documentation:

The requirements outlined for connecting from a MySQL client also apply when connecting from an application. See Connecting from a Client.

- Connector/J
- Connector/Python
- Connector/C++
- Connector/NET
- Connector/Node.js

Enabling Host Name Identity Verification

MySQL HeatWave on AWS supports TLSv1.2 and requires that all MySQL client and application connections are encrypted. For added security, you can download a signed certificate bundle and enable host name identity verification for your connecting clients and applications.

When a DB System is provisioned, a TLS certificate is installed on the MySQL Server. The certificate, which defines the DB System host name as the `Common Name`, is signed by a regional Certificate Authority (CA). When a client connects to the DB System with host name identity verification enabled and a CA certificate matching the one used by the server, the server and client place their trust in the same CA certificate and the client verifies that the host to which it connected is the one intended.

To obtain a CA certificate file, you must download a MySQL HeatWave on AWS certificate bundle. Store the certificate bundle in a secure location that is accessible to the client or application. The bundle is a regional CA certificate file in PEM format. You can download the regional certificate bundle for the AWS US East (N. Virginia) Region (`us-east-1`) from:

https://cloud.mysql.com/us-east-1/aws-us-east-1-cabundle.pem.

> **Note:**
>
> MySQL HeatWave on AWS is currently supported only in the AWS US East (N. Virginia) Region (us-east-1). Certificate bundles for other regions will be made available as support for MySQL HeatWave on AWS is extended to other regions.

Alternatively, you can download the regional certificate bundle using cURL, as shown:

```
$> curl -o aws-us-east-1-cabundle.pem \
   https://cloud.mysql.com/us-east-1/aws-us-east-1-cabundle.pem
```

> **Note:**
>
> It is recommend that you update your MySQL HeatWave on AWS certificate bundle quarterly to ensure that you always have the latest version. Issues connecting with `--ssl-mode` may indicate that your certificate bundle is outdated.

To establish an encrypted connection, launch the MySQL client with the `--ssl-ca` and `--ssl-mode` options; for example:

MySQL Shell:

```
$> mysqlsh --host HostNameOfMySQLDBSystem \
           --user=user1 \
           --password \
           --port=3306 \
           --ssl-mode=VERIFY_IDENTITY \
           --ssl-ca=aws-us-east-1-cabundle.pem
```

MySQL Command-Line Client:

```
$> mysql --host=HostNameOfMySQLDBSystem \
         --user=user1 \
         --password \
         --protocol=TCP \
         --port 3306 \
         --ssl-mode=VERIFY_IDENTITY \
         --ssl-ca=aws-us-east-1-cabundle.pem
```

where:

- `--host` specifies the host name of the DB System. The host name is found on the **MySQL DB System Details** page. See Viewing DB System Details.
- `--user` specifies the user name of the MySQL account to use for connecting to the server.
- `-p` specifies the password of the MySQL account used for connecting to the server. The password value is optional. If not given as in the examples above, `mysql` prompts for one.
- `--protocol` specifies transport protocol to use for connecting to the server. This option is not applicable to MySQL Shell.
- `--ssl-mode` is the security state of the connection to server. The `VERIFY_IDENTITY` mode ensures that an encrypted connection is established, that the TLS certificate is verified against the configured CA certificate, and that the host name identity is verified by checking the host name the client uses for connecting to the server against the identity in the certificate that the server sends to the client.
- `--ssl-ca` specifies the fully qualified path to the CA certificate file.

For more information about `ssl-*` connection options, see Command Options for Connecting to the Server.

6
Importing Data

This chapter describes how to export data from a source MySQL Server and import it into a DB System on the MySQL HeatWave on AWS using MySQL Shell.

Data must be imported to a DB System before it can be loaded into HeatWave. For information about loading data into HeatWave, see Manage HeatWave Data with Workspaces.

About MySQL Shell

MySQL Shell is the recommended utility for exporting data from a source MySQL Server and importing it into a DB System on the MySQL HeatWave on AWS. MySQL Shell dump and load utilities are purpose-built for use with MySQL DB Systems. To get the best functionality, always use the most recent version of MySQL Shell that is available to you.

The MySQL Server version used by MySQL HeatWave on AWS is fully supported by MySQL Shell. The minimum supported source MySQL Server versions supported by MySQL Shell are:

- MySQL 8.0.11
- MySQL 5.7.9

MySQL Shell provides the following utilities:

- `dumpInstance()`: MySQL instance export utility that exports all compatible schemas to local files. By default, this utility exports users, events, routines, and triggers. For more information, see MySQL Shell Instance and Schema Dump Utilities.
- `dumpSchemas()`: A schema export utility that exports selected schemas to local files. For more information, see MySQL Shell Instance and Schema Dump Utilities.
- `loadDump()`: An import utility that imports schemas to a DB System. For more information, see MySQL Shell Dump Loading Utility.

To import a schema to a MySQL DB System, MySQL Shell must be installed on a machine with access to the DB System. For instructions, see Connecting with MySQL Shell .

MySQL Shell dump files are exported as DDL files specifying the schema structure and tab-separated value (`.tsv`) files containing the data. The `.tsv` files are compressed using `zstd`, by default, but `gzip` is also available as an option. You can also choose no compression.

To improve performance, large tables are chunked by default. The default chunk size is 32MB. Chunking can be disabled, but this is not recommended for large databases. During import, the chunks can be imported by parallel threads, which can greatly improve import performance.

For more information about MySQL Shell, refer to the MySQL Shell User Guide.

MySQL Server Compatibility

MySQL HeatWave on AWS has several security-related restrictions that are not present in an on-premise instance of MySQL. To make it easier to load existing databases into the Service, the dump commands in MySQL Shell can detect potential issues and, in some cases, automatically adjust your schema definition to be compliant.

The `ocimds` option, when set to true, performs compatibility checks on the schemas for these issues and aborts the dump if any are found, while producing a detailed list of those issues and suggests additional steps to correct those issues. `loadDump` command only allows import of dumps created with the `ocimds` option enabled.

Some issues found by the `ocimds` option may require you to manually edit your schema before it can be loaded into the MySQL HeatWave on AWS. The MySQL Shell `compatibility` option can be used to automatically modify the dumped schemas, resolving some of these compatibility issues. You can pass one or more modifiers to the MySQL Shell `compatibility` option in a comma-separated list.

The MySQL Shell `compatibility` option applies the specified requirements for compatibility with MySQL HeatWave on AWS for all tables in the dump output, altering the dump files as necessary. From MySQL Shell 8.0.23, this option is available for all MySQL Shell utilities, and before that release, it is only available for the instance dump utility and schema dump utility.

MySQL Shell `compatibility` option modifiers include:

`force_innodb`
MySQL HeatWave on AWS supports the InnoDB storage engine, only. This option modifies the `ENGINE=` clause of `CREATE TABLE` statements to specify `INNODB` for tables that do not already use the `InnoDB` storage engine.

`skip_invalid_accounts`
Skips user accounts created with external authentication plugins that are not supported in MySQL HeatWave on AWS. From MySQL Shell 8.0.26, this option also removes user accounts that do not have passwords set, except where an account with no password is identified as a role, in which case it is dumped using the `CREATE ROLE` statement.

`strip_definers`
Removes the `DEFINER=account` clause from views, routines, events, and triggers. MySQL HeatWave on AWS requires special privileges to create these objects with a definer other than the user loading the schema. By removing the `DEFINER` clause, these objects will be created with that default definer. Views and routines have their `SQL SECURITY` clause changed from `DEFINER` to `INVOKER`. This ensures that the access permissions of the account querying or calling these objects are applied, instead of the user that created them. If your database security model requires views and routines have more privileges than their invoker, you must manually modify the schema before loading it. For more information, see The DEFINER Attribute and The SQL SECURITY Characteristic.

strip_restricted_grants
Certain privileges are restricted in MySQL HeatWave on AWS, such as `RELOAD` , `FILE` , `SUPER` , `BINLOG_ADMIN` , and `SET_USER_ID` . It is not possible to create users granting these privileges. This option removes those privileges from dumped `GRANT` statements.

strip_tablespaces
MySQL HeatWave on AWS has some restrictions on tablespaces. This modifier removes the `TABLESPACE` clause from `GRANT` statements, so all tables are created in their default tablespaces.

Additionally, `DATA DIRECTORY`, `INDEX DIRECTORY`, and `ENCRYPTION` options in `CREATE TABLE` statements are always commented out in DDL scripts if the MySQL Shell `ocimds` option is enabled.

> **Note:**
>
> If you intend to export data from an older version of MySQL, you should run the MySQL Shell Upgrade Checker Utility to generate a report of all potential issues with your migration. For more information, see MySQL Shell User Guide - Upgrade Checker Utility .

Exporting Data with MySQL Shell

This task describes how to export data from a source MySQL Server using MySQL Shell.

This task requires the following:

- The MySQL Shell command-line utility. The commands in this task use the JS (JavaScript) execution mode of MySQL Shell. For installation instructions, refer to Installing MySQL Shell.

  > **Note:**
  >
  > Exports created by MySQL Shell 8.0.27, or higher, cannot be imported by earlier versions of MySQL Shell. Using the latest version of MySQL Shell is always recommended.

- You have run the `dumpSchema` command with the `dryRun` and `ocimds` parameters set to true; for example:

  ```
  MySQL>JS> util.dumpSchemas(["tpch"], "tpchdump", {dryRun: true, ocimds:
  true})
  ```

  where `tpch` is the schema name and `tpchdump` is a local directory for the dump files.

  This performs a test run of the export, checking for compatibility issues. The compatibility issues and remediation steps are listed in the output. For more information about addressing compatibility issues, see MySQL Shell Instance and Schema Dump Utilities.

When exporting data from a source MySQL Server, you can either export an entire MySQL instance using `dumpInstance()` or specific schemas using `dumpSchemas()`. The syntax for each command is:

- `util.dumpInstance(outputUrl[, options])`
- `util.dumpSchemas(schemas, outputUrl[, options])`

The following task uses `dumpSchemas()` with the MySQL Shell `compatibility` option to export a schema named `tpch`.

To export a schema from a source MySQL Server:

1. Start a MySQL Shell session and connect to the source MySQL Server; for example:

   ```
   $> mysqlsh user@HostNameOfSourceMySQLServer
   ```

   For information about different MySQL Shell session and connection options, refer to the MySQL Shell User Guide.

2. Export the schema using the `dumpSchemas()` utility with any required MySQL Shell options and `compatibility` option modifiers. This command uses the `ocimds` option with several compatibility option modifiers.

   ```
   MySQL>JS> util.dumpSchemas(["tpch"], "/home/user1/tpchdump",
   {"ocimds": "true",
             "compatibility": ["force_innodb", "strip_definers",
   "strip_tablespaces"]})
   ```

   where:

   - `tpch` is the name of the schema being exported.
   - `/home/user1/tpchdump` is the location (a local directory) where the schema dump files are written.
   - `ocimds` checks your data for compatibility with MySQL HeatWave on AWS. If set to `true`, it is not possible to export data that is incompatible with MySQL HeatWave on AWS.
   - `compatibility` specifies compatibility modifiers that modify the exported data for compatibility with MySQL HeatWave on AWS.

> **Note:**
>
> For large datasets, it is recommended to use the MySQL Shell `bytesPerChunk` option to define larger chunks. The default chunk size is 32MB. To increase the size of the individual chunks, add the `bytesPerChunk` option to the command. For example: `bytesPerChunk: 128M` specifies a chunk size of 128MB.
>
> For more information about `dumpInstance` and `dumpSchemas` options, see Instance Dump Utility, Schema Dump Utility, and Table Dump Utility.

Importing Data with MySQL Shell

This task describes how to import a MySQL Shell dump to a MySQL DB System.

This task requires the following:

- MySQL Shell 8.0.27, or higher. Exports created by MySQL Shell 8.0.27, or higher, cannot be imported by earlier versions of MySQL Shell.
- Data exported from a source MySQL Server following the instructions described in Exporting Data with MySQL Shell .
- Enough storage space on your DB System for the data you intend to import. To check DB System storage space, see Viewing DB System Details.
- You have run the `loadDump()` command in `dryrun` mode to check that there will be no issues when the dump files are loaded from a local directory into the connected MySQL DB System; for example:

```
MySQL>JS> util.loadDump("/home/user1/tpchdump", {dryRun: true})
```

The following task uses `loadDump()` utility to load the schema that was exported in Exporting Data with MySQL Shell .

To import a schema:

1. Start a MySQL Shell session and connect to the DB System; for example:

   ```
   $> mysqlsh Username@HostNameOfMySQLDBSystem
   ```

   For information about connecting to a DB System using MySQL Shell, see Connecting with MySQL Shell .

2. Run the following command to import the `tpch` schema using the `loadDump()` utility.

   ```
   MySQL>JS> util.loadDump("/home/user1/tpchdump", {threads: 8})
   ```

   where:

   - `/home/user1/tpchdump` is the location of the MySQL shell dump files.

- `threads` specifies the number of processing threads to use for this task. The default is 4. For best performance, it is recommended to set this parameter to twice the number of CPUs used by the target MySQL DB System.

7

Manage HeatWave Data with Workspaces

The Workspaces page in the MySQL HeatWave Console permits loading data into or unloading data from a HeatWave Cluster. Load and unload operations are performed using *Auto Parallel Load*, which optimizes operation time and memory usage by predicting an using an optimal degree of parallelism. Data is loaded into HeatWave from the associated DB System. If you have not loaded data into your DB System, see Importing Data.

The MySQL user that is performing the load or unload operation must have the following MySQL privileges:

- The `PROCESS` privilege.
- The `EXECUTE` privilege on the `sys` schema.
- The `SELECT` privilege on the Performance Schema.

The time required to load data from the DB System into the HeatWave Cluster depends on the data size.

> **Note:**
>
> HeatWave Cluster data can also be managed using SQL or the Auto Parallel Load interface from a MySQL client. For more information, Loading Data, in the *MySQL HeatWave User Guide*.

To load data:

1. Connect to a DB System that has a HeatWave Cluster. For instructions, see Connecting from the Console.
2. Under **Select tables to unload or load into HeatWave**, select the schemas and tables that you want to load or unload. Selecting a schema selects all of the schema's tables. Expanding a table row using the drop-down control reveals the table's columns. The **Load Status** column provides the load status as a percentage value.
3. Optionally, enable the **Detail View** to view additional information about schemas and tables. Details include:
   - **Name**: The name of the schema or table
   - **Memory Size Estimate (GiB)**: An estimate of the memory required on the HeatWave Cluster for the schema or table
   - **Rows Estimate**: The estimated number of rows in the table
   - **Load Status**: The load status of the table as a percentage value. You can click the refresh icon to refresh the load status.
   - **String Column Encoding**: The number of columns to be encoded as `VARLEN` (variable-length) columns, expressed as a fraction of the total number of table columns. For information about HeatWave string column encoding, see Encoding String Columns, in the *HeatWave User Guide*.

- **Predicted Load Time (s)**: The predicted load time in seconds

4. After you have selected the tables you want to load or unload, click **Load** or **Unload**.

   If loading data, the **MySQL Auto Parallel Load tables into HeatWave** dialog appears, providing a summary of the load operation to be executed. The following information is shown:

   - **DB System**: The DB System name
   - **Estimated load size**: The estimated size of the data to be loaded
   - **Estimated load time**: The estimated time required to load the data
   - **Selected schemas and tables**: The selected schemas and tables to be loaded
     - **Name**: Name of the schema or table
     - **Memory Size Estimate (GiB)**: The memory size estimate for the schema or table on the HeatWave Cluster
   - **Estimated total memory footprint**: A pie chart showing the estimated percentage of HeatWave Cluster memory required for each table
   - **Estimated load times (seconds)**: A pie chart showing estimated load times for each table

   If unloading data, the **MySQL Auto Parallel Unload tables from HeatWave** dialog appears, providing a summary of the unload operation to be executed. The following information is shown:

   - **DB System**: The name of the DB System
   - **Estimated unload size**: The estimated size of the data to be unloaded
   - **Selected schemas and tables**: The selected schema and tables to be unloaded
     - **Name**: Name of the schema or table
     - **Memory Size Estimate (GiB)**: The memory size estimate for the schema or table on the HeatWave Cluster
   - **Estimated memory foot print (GiB)**: The estimated HeatWave Cluster memory footprint showing occupied memory, current free memory, and additional available free memory after the unload operation

5. Click **Confirm Load** or **Confirm Unload** to start the load or unload operation.

The **Cluster Memory Snapshot** shows the amount of HeatWave Cluster memory used.

8
Running Queries

The Workspaces tab in the MySQL HeatWave Console provides a Query Editor for running MySQL and HeatWave queries.

> **Note:**
>
> Queries can also be run from a MySQL client. See Connecting from a Client.

To run a query using the Query Editor:

1. Connect to a DB System. For instructions, see Connecting from the Console.
2. Enter a query into the **Query Editor**.
3. Click **Run Query**.

   The query results are displayed in tabular format by default. You can select the **JSON** tab to view results in `JSON` format.

   For multi-statement queries, only the results of the last query are displayed.

   The **Job Information** tab provides the **Job ID**, the number of rows returned, and the query statement that was processed.

Running HeatWave Queries

When a HeatWave Cluster is enabled and the data you want to query is loaded in HeatWave, queries that qualify are automatically offloaded from the MySQL DB System to the HeatWave Cluster for accelerated processing. No special action is required. Simply run the query from the Query Editor, as described above, or from a MySQL client that is connected to the DB System.

Before running a query, you can use `EXPLAIN` to determine if the query will be offloaded to HeatWave; for example:

```
mysql> EXPLAIN SELECT O_ORDERPRIORITY, COUNT(*)
          AS ORDER_COUNT FROM tpch.orders
          WHERE O_ORDERDATE >= DATE '1994-03-01'
          GROUP BY O_ORDERPRIORITY
          ORDER BY O_ORDERPRIORITY;
```

If the query can be offloaded to HeatWave, the `Extra` column of `EXPLAIN` output shows `"Using secondary engine RAPID"`. If that information does not appear, the query cannot be offloaded.

For more information about running HeatWave queries, see Running Queries, in the *MySQL HeatWave User Guide*. You can also refer to the HeatWave *Quickstarts* in the *MySQL*

HeatWave User Guide, which show how to import data into a DB System, load data into HeatWave, and run queries.

- tpch Analytics Quickstart
- AirportDB Analytics Quickstart

Query Editor Limitations

The Query Editor has the following limitations:

- It is not intended for loading data into a DB System. The most efficient method of loading data into a DB System is using MySQL Shell, which provides parallel load capabilities. For more information, see Importing Data.
- Only results for the last executed query are displayed.
- Creating or calling stored procedures is not supported.
- SQL code comments are not accepted.
- Long query results might be truncated.

9
HeatWave AutoML

Use the MySQL HeatWave Console to create and evaluate HeatWave AutoML models. For more information about HeatWave AutoML, see: HeatWave AutoML.

HeatWave AutoML Requirements

HeatWave AutoML has the following requirements:

- An operational DB System with sufficient resources and a HeatWave Cluster. See the following:
  - Supported Shapes.
  - Creating a DB System
  - Creating a HeatWave Cluster
- The MySQL account for the user who will train a model does not have a period character (".") in the username. For example, the username `'joesmith'@'domain'` can train a model, but the username `'joe.smith'@'domain'` cannot. For more information about this requirement, see Limitations.
- The MySQL account that will use HeatWave AutoML has been granted the following privileges:
  - `SELECT` and `ALTER` privileges on the schema that contains the machine learning datasets; for example:

    ```
    mysql> GRANT SELECT, ALTER ON schema_name.* TO 'user_name'@'%';
    ```

  - `SELECT` and `EXECUTE` on the MySQL `sys` schema where HeatWave AutoML routines reside; for example:

    ```
    mysql> GRANT SELECT, EXECUTE ON sys.* TO 'user_name'@'%';
    ```

Create a HeatWave AutoML model

Use the MySQL HeatWave Console to create a new HeatWave AutoML model.

Create a HeatWave AutoML model includes an advanced option to choose one or more algorithms. HeatWave AutoML uses the selected algorithms as a pool of algorithms, and will choose the most applicable algorithm from this pool to evaluate the model. By default, all algorithms are selected. To use a single, specific algorithm to evaluate the model, deselect all the other algorithms.

1. Select the **HeatWave AutoML** tab in the MySQL HeatWave Console, and then click **Connect to MySQL DB System**.

2. Select a DB System from the drop-down menu.
3. Enter a MySQL user name and password for the DB System.
4. Click **Connect**.
5. Click **Create Model**
6. **Name**: Specify a name for the model or use the generated default name.
7. **Description**: Specify a user friendly description for the model.
8. **Training table**: Select a training dataset.
9. **Columns to include in the model**: Select the **Target column**, and then choose which columns to **Include in model**.
10. **Machine learning task**: Choose a task.
11. **Advanced**:
    - **Optimization metric**: Choose an optimization metric.
    - **Algorithms to evaluate**: Choose which algorithms HeatWave AutoML should consider to evaluate the model.
12. Click **Create**.

    The MySQL HeatWave Console returns to the **HeatWave AutoML** page, and shows the new model at the top of the page.

Evaluate a HeatWave AutoML model

Evaluate a HeatWave AutoML model includes scores for the model, predictions and explanations. What if analysis compares the baseline results to alternative values.

During testing, the model can include a column of known values. After evaluation, if a prediction does not match the known value, MySQL HeatWave Console marks the prediction in red.

1. Select the **HeatWave AutoML** tab in the MySQL HeatWave Console, and then click **Connect to MySQL DB System**.
2. Select a DB System from the drop-down menu.
3. Enter a MySQL user name and password for the DB System.
4. Click **Connect**.
5. Select a model from the list. The lower pane shows details for the model: **Training table**, **Target column**, **Description**, **Selected algorithm**, **Machine learning task**, and **Training score**.
6. Click **Evaluate**.
7. **Select table**: The prompt has a reminder of the name of the training table. Select a test table compatible with the training table, and click **Next**.

   The **Evaluate Model: *model name*** dialog opens.
8. **Model score**:
    - Select one of the **Scoring metrics**, and click **Calculate score**. The **Results** shows the score.
9. **Explain model**:

- **Feature Importance**: Shows the names of each feature and their relative importance to the model.

10. **Predictions**:
    - Click **Generate Predictions** to create a table of predictions for each row in the test table.
    - Select a prediction row from the table, and click **Explain Prediction**.

      The dialog shows the **Selected row**, and **Feature Importance**. **Notes** explains which feature had the largest impact on the prediction, and might also include the feature that had the largest impact against the prediction.
    - Click **Back** to return to the predictions table.
    - Select a prediction row from the table, and click **What If**.
    - **Values for comparison from included features**: Click the i to show information for that feature column.

      This includes **Minimum**, **Mean** and **Maximum**, and a bar chart of values.
    - Click **Back** to return to the predictions table.
    - Select a prediction row from the table, and click **What If**.
    - **Values for comparison from included features**: Adjust the values, and click **Create**.

      The dialog shows the **Comparison data**, and **Feature Importance**. **Notes** has explanations for the Baseline and the Comparison. Both explain which feature had the largest impact on that prediction, and might also include the feature that had the largest impact against that prediction.
    - Click **Back** to return to the predictions table.

10

System Variables

System Variables

The following global system variables are specific to MySQL HeatWave on AWS and are not available in the on-premise version of MySQL Server.

- `telemetry_log_disable`

Property	Description
Introduced	8.0.32
System Variable	`telemetry_log_disable`
Scope	Global
Dynamic	Yes
SET_VAR Hint Applies	No
Type	Boolean
Default Value	`OFF`

Whether to disable the telemetry log.

11

Performance Monitoring

The Performance tab in the MySQL HeatWave Console provides HeatWave *Cluster* and *Workload* performance data. An active HeatWave Cluster is required for cluster and workload performance monitoring.

HeatWave Cluster Performance

To view HeatWave Cluster performance data:

1. In the MySQL HeatWave Console, select the **Performance** tab.
2. Select a DB System and HeatWave Cluster from the drop-down.

   The **Cluster** tab shows cluster performance data for HeatWave and for the DB System.

   Click the **Refresh** icon to refresh the performance data.
3. To see performance data for another DB System and HeatWave Cluster, select them from the drop-down, and click the **Refresh** icon.

For descriptions of HeatWave Cluster performance data, see HeatWave Cluster Performance Data.

HeatWave Cluster Performance Data

- **HeatWave**
  - **Cluster Memory Utilization**

    The percentage of total HeatWave Cluster memory that is currently utilized.
  - **Node Memory Utilization**

    The percentage of memory utilized on each cluster node. Select from the drop-down menu to view data in order of node **ID** or **Memory Utilization**.
  - **Data Dictionary**

    The total data dictionary size. Dictionaries for dictionary-encoded string columns are stored on the MySQL DB System node. For more information, see String Column Encoding Reference, in the *MySQL HeatWave User Guide*.
  - **Dataset**

    The size of the dataset loaded on the HeatWave Cluster.
- **MySQL**
  - **CPU Utilization**

    The percentage of CPU utilization on the MySQL DB System.
  - **Memory Utilization**

    The percentage of memory utilization on the MySQL DB System,
  - **Buffer Pool**

The buffer pool size on the MySQL DB System.

- **Disk Operations**

  The number of disk operations on the MySQL DB System (write, read, and miscellaneous).

- **Connections**

  The number of idle and active client connections on the MySQL DB System.

Workload Performance

To view HeatWave workload performance data:

1. On the MySQL HeatWave Console, select the **Performance** tab.
2. Select a DB System and HeatWave Cluster from the drop-down.

   The **Workload** tab shows workload performance data for HeatWave and for the MySQL node.

   Click the **Refresh** icon to refresh the performance data.
3. To see workload performance data for another DB System and HeatWave Cluster, select them from the drop-down, and click the **Refresh** icon.

For HeatWave workload performance data descriptions, see HeatWave Workload Performance Data.

HeatWave Workload Performance Data

Information is shown for the last 1000 query executions.

- **Executions**

  The number of query executions on a time scale.

- **Recent Queries**

  Displays recently executed queries with the **Query Text**, the query **Start Time**, and the **Duration (ms)** of the query.

  You are provided with options to **Search by Query Text** for specific queries and to **Aggregate Executions** to view aggregated **Start Time** and **Duration (ms)** data for the same query.

Autopilot Shape Advisor

As of MySQL 8.0.32, use the Auto Shape Prediction feature in MySQL Autopilot for MySQL HeatWave on AWS to analyze the workload and assess the suitability of the current MySQL shape. The Auto Shape Prediction feature is available prior to MySQL 8.0.32, but not enabled by default.

The Auto Shape Prediction feature begins to collect MySQL statistics that reflect the current workload. It collects statistics at varying intervals, and Auto Shape Prediction creates a prediction every five minutes while it is active. If there is insufficient or no activity in a five minute interval, or if buffer pool usage is growing, Auto Shape Prediction cannot make a prediction for that interval.

Auto Shape Prediction looks at buffer pool usage, workload activity, and access patterns, and bases its recommendation on those factors. Choose a shape that will accommodate the recommended buffer pool size. If the buffer pool is too small, although MySQL will generally be able to run the workload with stability, the performance of the DB System will suffer through excessive disk I/O.

> **Tip:**
>
> If Auto Shape Prediction suggests a possible downsize, consider the DB System CPU usage and memory usage before downsizing. For heavy CPU utilization, downsizing to a shape with fewer CPUs is not recommended.

When Auto Shape Prediction is running it keeps a rotating history for just over seven days. While there is a monitoring overhead from periodic statistics collection and prediction events, for most workloads and shape combinations the overhead is negligible. If necessary, it is possible to disable Auto Shape Prediction.

When Auto Shape Prediction is disabled it clears the internal statistics tables, but keeps the predictions.

An upgrade to the DB System drops and re-installs the `mysql_autopilot` schema, which removes the predictions.

As of MySQL 8.0.32, there are two ways to access the Autopilot Shape Advisor, with the MySQL HeatWave Console or with MySQL statements. Prior to MySQL 8.0.32, use MySQL statements.

> **Note:**
>
> The SQL output can only provide hints. The MySQL HeatWave Console can provide much more information, and can recommend an improved shape. See: Autopilot Shape Advisor with MySQL HeatWave Console .

Autopilot Shape Advisor with MySQL HeatWave Console

To view the Autopilot Shape Advisor:

1. In the MySQL HeatWave Console, select the **Performance** tab.
2. Select a DB System and HeatWave Cluster from the drop-down.

   The **Autopilot Shape Advisor** tab shows Auto Shape Prediction data for the DB System.

   Click the **Refresh** icon to refresh the performance data.
3. To see performance data for another DB System and HeatWave Cluster, select them from the drop-down.

For descriptions of Auto Shape Prediction data, see Auto Shape Prediction Data.

Auto Shape Prediction Data

Auto Shape Prediction data shows data for the last seven days and contains the following information:

- **Time Periods**
  - **Selected**

    The two dates selected in **Buffer Pool Insights**.
  - **Available**

    Auto Shape Prediction data is available between these two dates a week apart.
- **MySQL Shape Prediction**

  This uses the **Selected** dates.
  - **Current**

    The current shape size, with **Shape Name**, **Memory Size**, and **Buffer Pool Size**.
  - **Recommended**

    The recommended shape size, with **Shape Name**, **Memory Size**, and **Buffer Pool Size**.
- **Recommended Action**

  This uses the **Selected** dates, and has five possible recommendations:
  - **Upsize**: Move the DB System to a larger MySQL shape.
  - **Downsize**: Move the DB System to a smaller MySQL shape.
  - **No Change**: No need to change the MySQL shape.
  - **Not Enough Data**: There is not enough workload activity, or Auto Shape Prediction cannot make a prediction.
  - **Prediction Data Not Available**: Auto Shape Prediction is disabled, or it is enabled but has not produced the first prediction, or the MySQL HeatWave Console data has not been refreshed within the last 5 minutes.

  Instructions to follow to adopt the recommended size.
- **Buffer Pool Insights**
  - **Average Statistics**
    - **Buffer Pool**

      A graph that shows **Average Buffer Pool Usage** and **Current Buffer Pool Size**. This uses the **Selected** dates.
    - **Buffer Pool Hit Rate**

      A percentage value calculated between the **Selected** dates.
  - **Recent Statistics**

    Use the left and right slider controls on one of the two time series graphs to adjust the **Selected** dates.
    - **Buffer Pool**

A detailed graph that shows **Buffer Pool Size**, **Buffer Pool Usage**, and **Recommended Buffer Pool Size**. The graph uses the **Selected** dates.

* **Buffer Pool Hit Rate**

  A detailed graph that shows the **Buffer Pool Hit Rate** Between the **Selected** dates.

* Workload markers on the two time series graphs. These markers indicate recent activity:
  * Green markers: The workload is running and stable. Good predictions.
  * Orange markers: The workload is running, but it is not stable. Difficult to predict.
  * No markers: No recent activity, and no predictions.

Autopilot Shape Advisor with MySQL

Auto Shape Prediction records predictions in the `shape_predictions` table in the `mysql_autopilot` schema. This is a system schema that is always present even if Auto Shape Prediction is not active. The schema also contains tables for the statistics used to calculate the predictions.

> **Note:**
>
> The SQL output can only provide hints. The MySQL HeatWave Console can provide much more information, and can recommend an improved shape. See: Autopilot Shape Advisor with MySQL HeatWave Console .

This example shows output from the `shape_predictions` table with predictions made over the course of an hour:

```
mysql> SELECT * FROM mysql_autopilot.shape_predictions;
+---------------------+-----------------+----------------------------
+-----------------------------------+
| prediction_time     | current_bp_size | hit_rate | prediction_value |
outcome                           |
+---------------------+-----------------+----------------------------
+-----------------------------------+
| 2022-10-28 09:46:48 | 5               | NULL     | 0                | NOT
ENOUGH FEATURE DATA SNAPSHOTS |
| 2022-10-28 09:51:48 | 5               | 0.866    | 4.97363          |
FEATURE DATA IS NOT STABLE        |
| 2022-10-28 09:56:48 | 5               | 0.8665   | 4.97778          |
FEATURE DATA IS NOT STABLE        |
| 2022-10-28 10:01:48 | 5               | 0.866    | 25.08            |
UPSIZE                            |
| 2022-10-28 10:06:48 | 5               | 0.86425  | 25.63            |
UPSIZE                            |
| 2022-10-28 10:11:48 | 5               | 0.8675   | 26.1             |
UPSIZE                            |
| 2022-10-28 10:16:48 | 5               | 0.8675   | 26.52            |
UPSIZE                            |
```

```
| 2022-10-28 10:21:48 | 5               | 0.86625  | 27.24
| UPSIZE                              |
| 2022-10-28 10:26:48 | 5               | 0.86525  | 27.42
| UPSIZE                              |
| 2022-10-28 10:31:48 | 5               | 0.866    | 27.83
| UPSIZE                              |
| 2022-10-28 10:36:48 | 5               | 0.866    | 28.18
| UPSIZE                              |
+---------------------+-----------------+----------+-----------------
+-------------------------------------+
```

The `shape_predictions` table has these columns:

- `prediction_time`: The timestamp for this prediction. A prediction is attempted every five minutes.
- `current_bp_size`: The current buffer pool size in GB.
- `hit_rate`: The current buffer pool hit rate.
- `prediction_value`: The predicted buffer pool size for optimal performance with this workload in GB.
- `outcome`: A recommendation to upsize, downsize, or stay with your current shape, or information on why a prediction cannot be made for that interval.

To use Auto Shape Prediction, follow these steps:

1. Connect to the DB System for the HeatWave Cluster, using the MySQL client (`mysql`) or MySQL Shell in SQL mode. For instructions to connect to MySQL HeatWave on AWS as a client, see Connecting from a Client.
2. Prior to MySQL 8.0.32, while a typical workload is running, enable Auto Shape Prediction by issuing the following statement using the client:

   ```
   mysql> CALL mysql_autopilot.shape_prediction(JSON_OBJECT("enable",
   TRUE));
   ```

3. Wait at least five minutes for the first prediction to be attempted, then start to check the results with this statement from the SQL client:

   ```
   mysql> SELECT * FROM mysql_autopilot.shape_predictions
             ORDER BY prediction_time DESC LIMIT 20;
   ```

   Auto Shape Prediction attempts a new prediction every five minutes. Re-issue the statement every so often while a typical workload is running, until the prediction has stabilized.
4. When the prediction has stabilized, make a note of the maximum `prediction_value` and `outcome`.

5. Choose whether to leave Auto Shape Prediction running to monitor other workloads or disable Auto Shape Prediction with this statement:

```
mysql> CALL mysql_autopilot.shape_prediction(JSON_OBJECT("enable",
FALSE));
```

6. To move to a different DB System shape, follow the steps in Creating a Backup and Restoring a Backup to a New DB System.

   The Auto Shape Prediction Data includes **Recommended Action**, which recommends a shape, and provides instructions.

12
Backups

This section describes how to create, restore, and manage MySQL DB System backups. Only manual backups are supported; that is, you must create your backups following the instructions in Creating a Backup. Backups are not created automatically.

DB System backups are Amazon EBS snapshots which are automatically saved to Amazon Simple Storage Service (Amazon S3). MySQL HeatWave on AWS limits the maximum number of backups that you can retain across all DB System instances. If you reach the limit, remove the oldest backups before creating new ones.

It is recommended to create backups on a regular schedule. Backups are incremental, but if the period between backups is too long, a backup operation may take a long time to complete due to the volume of changes.

Creating a Backup

Use the Console to create a backup.

1. In the MySQL HeatWave Console, click the **MySQL** tab.
2. On the **DB Systems** tab, in the list of DB Systems, find the DB System you want to create a backup for, and do one of the following:
   - Click the row of the DB System to highlight it, and click **Create Backup** from the **Actions** menu.
   - Click the name of the DB System to open the **Details** page and click **Create Backup** from the **Actions** menu.
3. In the **Create MySQL Backup** panel, edit the following fields:
   - **Display name**: Specify a name of the backup. If you do not define a name, one is generated for you in the format `NameOfTheBackup - Backup - Day, Date Time`.
   - **Description**: Specify a description of the backup. If you do not provide a description, one is generated for you in the format `NameOfTheBackup - Manual Backup - Day, Date Time`.
   - **Backup Retention Policy**: Specify how may days you want to retain the user initiated backup. By default, the backups are retained for 365 days.
4. Click **Create** to create the backup.

Under **MySQL**, click the **Backups** tab to view the backup you created. When the backup is created successfully, the backup state changes from **Creating** to **Active**.

Editing a Backup

Use the Console to edit the display name, description, and backup retention policy of a DB system backup.

1. In the MySQL HeatWave Console, click the **MySQL** tab.

2. On the **Backups** tab, in the list of DB Systems, find the backup you want to edit, and do one of the following:
   - Click the row of the backup to highlight it, and click **Edit Backup**.
   - Click the name of the backup to open the **Details** page and click **Edit Backup**.
3. In the **Edit MySQL Backup** panel, edit the following:
   - **Display Name**: Specify a name of the backup. If you do not define a name, one is generated for you in the format, `NameOfTheBackup - Backup - Day, Date Time`.
   - **Description**: Specify a description of the backup. If you do not provide a description, one is generated for you in the format, `NameOfTheBackup - Manual Backup - Day, Date Time`.
   - **Backup Retention Policy**: (Only for user initiated backups) Specify how may days you want to retain the user initiated backup. By default, the backups are retained for 365 days.
4. Click **Save**.

The details of the selected backup is updated.

Viewing Backup Details

To view backup details:

1. In the MySQL HeatWave Console, select the **MySQL** tab.
2. Select the **Backups** tab. In the list of backups, find the backup you want to view details for and click the name of the backup to open the **MySQL Backup Details** page.

For descriptions of MySQL backup details, see Backup Details.

Backup Details

Use the Console to view the **Backup Details** page.

Table 12-1 Backup Details Page

Name	Description
Summary	Backup summary. See Table 12-2.
General information	General backup. See Table 12-3.
MySQL DB System snapshot	MySQL DB System snapshot. See Table 12-4.

Table 12-2 Backup Summary

Field	Description
State	The state of the backup. • **CREATING**: Resources are being reserved for the backup, and the backup is being created. Creating a backup can take several minutes. The backup is not ready to use yet. • **ACTIVE**: The backup was successfully created • **UPDATING**: The backup is in the process updating • **INACTIVE**: The backup is unavailable • **DELETING**: The backup is being deleted by a delete action in the Console • **DELETED**: The backup has been deleted and is no longer available • **FAILED**: An error condition prevented the creation or continued availability of the backup
Resource ID	The unique resource identifier assigned to the backup
Storage size	The size of the backup

Table 12-3 General information

Field	Description
Description	A description of the backup
Created	The date and time the backup was created
Last Updated	The date and time the backup was last updated
Retention in days	The number of days the user initiated backup is retained
Expiry time	The date and time when the backup expires
Backup type	The type of the backup: User initiated or scheduled

Table 12-4 MySQL DB System snapshot

Field	Description
Name	The name of the DB System used to create the backup
Description	The user-specified description of the DB System used to create the backup
MySQL version	The MySQL version used to create the backup
Shape	The resource template applied to the DB System used to create the backup
MySQL Configuration	The configuration of the DB System
Availability Zone	The physical Availability Zone of the DB System used to create the backup

Table 12-4 (Cont.) MySQL DB System snapshot

Field	Description
Maintenance window	The time when the two-hour maintenance window begins for the DB System used to create the backup
Automatic backups	The status of the automatic backups: Enabled or Disabled

Restoring a Backup to a New DB System

When you restore a backup, you create a new DB System and restore the backup to it. You can change the shape and the amount of data storage for the new DB System.

A restored DB System uses the same MySQL Administrator user name and password that was in effect when the backup was created.

DB System backups are Amazon EBS snapshots which are automatically saved to Amazon Simple Storage Service (Amazon S3).

> **Tip:**
>
> Amazon EBS snapshots are loaded asynchronously. You can use the new DB System as soon as it is active, while data continues to load in the background. If you access data that has not been loaded yet, the DB System downloads the requested data (which can therefore take longer than usual to access), then resumes asynchronous loading. If you want to download your frequently accessed data immediately, run a full-table scan such as `SELECT *` on the relevant tables.

To restore a backup to a new DB System:

1. In the MySQL HeatWave Console, select the **MySQL** tab.
2. Select the **Backups** tab. In the list of backups, find the backup you want to restore to a new DB System, and do one of the following:
   - Click on the row of the backup to highlight it, and click **Restore Backup**.
   - Click the name of the backup to open the **MySQL Backup Details** page. Click **Restore Backup**.

   The **Restore Backup to new MySQL DB System** dialog is displayed.
3. Provide basic information for the new DB System:
   - **Display Name**: Specify a display name for the new DB System or use the generated default name.
   - **Description**: Specify a description for the new DB System or use the generated description.

> **Note:**
>
> A restored DB System uses the same MySQL Administrator user name and password that was in effect when the backup was created.

4. Select the **Hardware shape**. If you are using this process to get a different shape or more storage for the DB System, choose the shape that you want the new DB System to have.
   - **Shape**: Select the shape to use for your DB System. The shape determines the resources allocated to the system. For information about shapes, see Supported Shapes. Selecting a shape for a DB System also selects the configuration associated with that shape. See Configuration.
   - **Data Storage Size (GiB)**: Specify the amount of block storage, in GiB, to allocate to the DB System. This block storage stores all data, logs, and temporary files. Binaries are not stored in this block storage. Enter a number between 50 and 16384.

     > **Note:**
     >
     > Ensure that the specified Data Storage Size is greater than or equal to the size of the backup you are restoring.

   - **Database Version**: Select the MySQL Server version you want to deploy. The latest MySQL Server version is selected by default.
5. Configure the **Availability zone**, which determines the physical location of the DB System:
   - Select **Automatically** to have the physical AWS Availability Zone selected for you.
   - Set **Manually** to select the physical AWS Availability Zone where the MySQL DB System will be created.

   > **Note:**
   >
   > A physical Availability Zone is identified by an Availability Zone ID (AZ ID). For information about AZ IDs and how to view them, see Availability Zone IDs for your AWS resources, in the *AWS RAM User Guide*.

6. Configure **Networking** settings:
   - **Allowed Client Addresses**: Specify public-facing client IPv4 addresses that are permitted to connect to the DB System endpoint. Addresses are specified in CIDR format; for example: `1.2.3.4/24`. Multiple addresses in CIDR format can be specified in a semicolon-separated list; for example: `1.2.3.4/24; 1.2.3.4/32`. For information about specifying IP addresses in CIDR format, see Creating a DB System.
   - **Port**: The port on which the MySQL server listens. The default is port 3306. You can specify a port number between 1024 and 65535.
   - **X Protocol Port**: The X Protocol port on which the MySQL server listens, supported by clients such as MySQL Shell. The default port is 33060. You can specify a port number between 1024 and 65535.

7. Click **Next** to proceed to creating a HeatWave Cluster.
8. Provide basic information for the new HeatWave Cluster:
   - **Display Name**: Specify a display name for the new HeatWave Cluster or use the generated default name.
   - **Description**: Specify a description for the new HeatWave Cluster.
9. Under **HeatWave Cluster Configuration**, select a node shape and number of nodes.
   - **Shape**: Select a HeatWave node shape. For information about supported shapes, see Supported Shapes.

     > **Note:**
     >
     > If you intend to use HeatWave AutoML functionality, the `HeatWave.256GB` node shape is recommended when creating a HeatWave Cluster. The `HeatWave.16GB` node shape may not have enough memory for training on large data sets. If you see error messages about this (such as `ML003024`), use the larger shape instead.

   - **Cluster Size**: The number of HeatWave nodes to create. Enter a number between 1 and 128.

     > **Tip:**
     >
     > Because the new DB System does not yet contain any data, you cannot use MySQL Autopilot to estimate the required cluster size (as described in Estimating Cluster Size with MySQL Autopilot). If you want to make an estimate, after loading the data to the DB System, you can delete the HeatWave Cluster and create a new one following the instructions in Creating a HeatWave Cluster.

10. Click **Restore** to restore the backup to the new DB System.

    You are returned to the DB Systems page where you can monitor the state of the operation, which may take some time to complete. The state will change from **Creating** to **Active** when the operation has completed successfully.

Deleting a Backup

Deleting a backup permanently deletes it.

To delete a backup:

1. In the MySQL HeatWave Console, select the **MySQL** tab.
2. Select the **Backups** tab. In the list of backups, find the backup you want to restore to a new DB System, and do one of the following:
   - Click on the row of the backup to highlight it, and click **Delete**.

- Click the name of the backup to open the **MySQL Backup Details** page. Click **Delete**.

The **Delete MySQL DB System Backup** dialog is displayed for you to confirm the deletion.

3. Click **Delete MySQL Backup** to go ahead with the deletion.

13 Configuration

The MySQL configuration comprises global system variables and session variables, which define its operation.

Creating a MySQL Configuration

Use the MySQL HeatWave Console to create a new MySQL configuration. It is not possible to change the values of any variables once a configuration has been created.

Review the default values for the MySQL global system variables, and choose preferred values. See System Initialization Variables, and User Configurable System Variables

Certain MySQL global system variables define whether a configuration can support HeatWave. See: User Configurable Shape Dependent System Variables.

1. In the MySQL HeatWave Console, select the **MySQL** tab, and then click **Configurations** to open the **Configurations** page.
2. On the **Configurations** page, click **Create MySQL Configuration** to open the **Create MySQL Configuration** dialog.
3. **Basic information**: Provide a name and description for the configuration:
   - **Display Name**: Specify a display name for the configuration or use the generated default name.
   - **Description**: Specify a user friendly description for the configuration.
4. **Select MySQL shape**: Select a shape to create a configuration for.
5. Click **Next**.
6. **Support HeatWave**: Select this option to ensure that the configuration supports HeatWave.
7. **Initialization variables**: Set preferred values for initialization variables.
   - Click **Select a variable name**, and select the variable from the drop-down list.
   - Click **Select a value**, and select the value from the drop-down list, or enter the preferred value.
8. **User variables**: Set preferred values for user variables.
   - Click **Select a variable name**, and select the variable from the drop-down list.
   - Click **Select a value**, and select the value from the drop-down list, or enter the preferred value.
   - Click **Add new variable**, and repeat these steps to create the full configuration.
9. Review all the variables, and their values. Edit the values, or if any are incorrect, click **X** to remove the variable.
10. Click **Create**.

The MySQL HeatWave Console checks the values for each variable. If any fail validation `Could not create MySQL Configuration` appears, with the name of the failed variable. Edit the value, and click **Create**.

The MySQL HeatWave Console returns to the **Configurations** page, and shows the new configuration at the top of the page.

11. Click on the configuration to view the **Details** page. This shows the variables and their values associated with the configuration.

    Click the **Associated DB Systems** tab to see all the DB Systems that use the configuration.

    Click **Delete** if no DB System uses the configuration, and it is not required.

System Initialization Variables

System initialization variables are global system variables that must be set during DB system initialization. They apply for the life span of the DB system and, once applied, you cannot change them.

Configure the following variables with the MySQL HeatWave Console. See Creating a MySQL Configuration.

> **Note:**
>
> Any change to the configuration of a DB system cannot change the system initialization variables.

Table 13-1 Initialization Variables

Name	Value
lower_case_table_names	• `CASE_SENSITIVE`: (Default) Table and schema name comparisons are case-sensitive and are stored as you specify them. Selecting `CASE_SENSITIVE` sets the system variable, `lower_case_table_names`, to 0. • `CASE_INSENSITIVE_LOWERCASE`: Table and schema name comparisons are not case-sensitive and stored in lowercase. Selecting`CASE_INSENSITIVE_LOWERCASE` sets the system variable, `lower_case_table_names`, to 1.

User Configurable System Variables

Configure the following global system variables with the MySQL HeatWave Console. See Creating a MySQL Configuration.

Table 13-2 User Configurable System Variables

Name	Default Value
autocommit	ON

Table 13-2 (Cont.) User Configurable System Variables

Name	Default Value
big_tables	OFF
binlog_expire_logs_seconds	3600
binlog-row-metadata	MINIMAL
binlog_row_value_options	PARTIAL_JSON
binlog_transaction_compression	OFF
completion_type	NO_CHAIN
connect_timeout	10
connection_memory_chunk_size	8912
connection_memory_limit	9223372036854775807
cte_max_recursion_depth	1000
default_authentication_plugin	CACHING_SHA2_PASSWORD
foreign_key_checks	ON
global_connection_memory_limit	9223372036854775807
global_connection_memory_tracking	OFF
group_replication_consistency	EVENTUAL
information_schema_stats_expiry	86400
innodb_buffer_pool_dump_pct	25
innodb_buffer_pool_instances	Shape dependent, see User Configurable Shape Dependent System Variables
innodb_buffer_pool_size	Shape dependent, see User Configurable Shape Dependent System Variables
innodb_ddl_buffer_size	1048576
innodb_ddl_threads	4
innodb_ft_enable_stopword	ON
innodb_ft_max_token_size	84
innodb_ft_min_token_size	3
innodb_ft_num_word_optimize	2000
innodb_ft_result_cache_limit	2000000000
innodb_ft_server_stopword_table	NULL
innodb_lock_wait_timeout	50
innodb_log_writer_threads	ON
innodb_max_purge_lag_delay	300000
innodb_max_purge_lag	0
innodb_stats_persistent_sample_pages	20
innodb_stats_transient_sample_pages	8
interactive_timeout	28800
--local-infile	OFF
mandatory_roles	Empty string
max_allowed_packet	67108864
max_binlog_cache_size	4294967296
max_connect_errors	100

Table 13-2 (Cont.) User Configurable System Variables

Name	Default Value
max_connections	Shape dependent, see User Configurable Shape Dependent System Variables
max_execution_time	0
max_heap_table_size	16777216
max_prepared_stmt_count	Shape dependent, see User Configurable Shape Dependent System Variables
mysql_firewall_mode	ON
mysqlx_connect_timeout	30
mysqlx_deflate_default_compression_level	3
mysqlx_deflate_max_client_compression_level	5
mysqlx_interactive_timeout	28800
mysqlx_lz4_default_compression_level	2
mysqlx_lz4_max_client_compression_level	8
mysqlx_max_allowed_packet	67108864
mysqlx_read_timeout	28800
mysqlx_wait_timeout	28800
mysqlx_write_timeout	60
mysqlx_zstd_default_compression_level	3
mysqlx_zstd_max_client_compression_level	11
net_read_timeout	30
net_write_timeout	60
parser_max_mem_size	18446744073709551615
regexp_time_limit	32
sort_buffer_size	262144
sql_mode	ERROR_FOR_DIVISION_BY_ZERO, NO_ENGINE_SUBSTITUTION, NO_ZERO_DATE, NO_ZERO_IN_DATE, ONLY_FULL_GROUP_BY, STRICT_TRANS_TABLES
sql_require_primary_key	OFF
sql_warnings	OFF
telemetry_log_disable	OFF
thread_pool_dedicated_listeners	OFF
thread_pool_max_transactions_limit	Shape dependent, see User Configurable Shape Dependent System Variables
thread_pool_query_threads_per_group	Shape dependent, see User Configurable Shape Dependent System Variables
time_zone	UTC
tmp_table_size	16777216

Table 13-2 (Cont.) User Configurable System Variables

Name	Default Value
transaction_isolation	REPEATABLE-READ
wait_timeout	28800

User Configurable Shape Dependent System Variables

Shape dependent global system variables are linked to, and tuned for, a specific MySQL shape. See Supported Shapes. Configure the following variables with the MySQL HeatWave Console. See Creating a MySQL Configuration.

innodb_buffer_pool_instances , innodb_buffer_pool_size , max_connections , and max_prepared_stmt_count have different default, minimum and maximum values for each shape and for HeatWave support.

thread_pool_max_transactions_limit and thread_pool_query_threads_per_group have different default values for each shape.

Table 13-3 innodb_buffer_pool_instances

Shape	Supports HeatWave	Default	Minimum	Maximum
MySQL.2.16GB	No	4	1	64
MySQL.2.16GB	Yes	4	1	64
MySQL.4.32GB	No	4	1	64
MySQL.4.32GB	Yes	4	1	64
MySQL.8.64GB	No	4	1	64
MySQL.8.64GB	Yes	4	1	64
MySQL.32.256GB	No	8	1	64
MySQL.32.256GB	Yes	4	1	64

Table 13-4 innodb_buffer_pool_size

Shape	Supports HeatWave	Default	Minimum	Maximum
MySQL.2.16GB	No	10737418240	5242880	10737418240
MySQL.2.16GB	Yes	5368709120	5242880	5368709120
MySQL.4.32GB	No	21474836480	5242880	21474836480
MySQL.4.32GB	Yes	16106127360	5242880	16106127360
MySQL.8.64GB	No	51539607552	5242880	61203283968
MySQL.8.64GB	Yes	45097156608	5242880	45097156608
MySQL.32.256GB	No	206158430208	5242880	246960619520
MySQL.32.256GB	Yes	21474836480	5242880	21474836480

Table 13-5 max_connections

Shape	Supports HeatWave	Default	Minimum	Maximum
MySQL.2.16GB	No	1000	1	100000
MySQL.2.16GB	Yes	1000	1	100000
MySQL.4.32GB	No	2000	1	100000
MySQL.4.32GB	Yes	2000	1	100000
MySQL.8.64GB	No	4000	1	100000
MySQL.8.64GB	Yes	4000	1	100000
MySQL.32.256GB	No	8000	1	100000
MySQL.32.256GB	Yes	2000	1	100000

Table 13-6 max_prepared_stmt_count

Shape	Supports HeatWave	Default	Minimum	Maximum
MySQL.2.16GB	No	16382	16382	20000
MySQL.2.16GB	Yes	16382	16382	20000
MySQL.4.32GB	No	16382	16382	40000
MySQL.4.32GB	Yes	16382	16382	40000
MySQL.8.64GB	No	16382	16382	80000
MySQL.8.64GB	Yes	16382	16382	80000
MySQL.32.256GB	No	16382	16382	160000
MySQL.32.256GB	Yes	16382	16382	40000

Table 13-7 thread_pool_max_transactions_limit

Shape	Default	Minimum	Maximum
MySQL.2.16GB	8	0	100000
MySQL.4.32GB	16	0	100000
MySQL.8.64GB	32	0	100000
MySQL.32.256GB	512	0	100000

Table 13-8 thread_pool_query_threads_per_group

Shape	Default	Minimum	Maximum
MySQL.2.16GB	16	1	4096
MySQL.4.32GB	16	1	4096
MySQL.8.64GB	16	1	4096
MySQL.32.256GB	16	1	4096

Service Specific System Variables

MySQL HeatWave on AWS defines the following global system variables. It is not possible to edit them.

`generated_random_password_length` , `query_alloc_block_size` , and `query_prealloc_size` are also session variables. See Session Variables.

Table 13-9 Service Specific Global System Variables

Name	Default Value
generated_random_password_length	20
mysqlx_document_id_unique_prefix	0
mysqlx_enable_hello_notice	ON
mysqlx_idle_worker_thread_timeout	60
mysqlx_min_worker_threads	2
query_alloc_block_size	8192
query_prealloc_size	8192

The `SHOW VARIABLES` statement with the `GLOBAL` modifier displays global system variable values:

```
mysql> SHOW GLOBAL VARIABLES;
```

To obtain the value for a specific variable, use a `LIKE` clause as shown:

```
mysql> SHOW GLOBAL VARIABLES LIKE 'max_join_size';
```

To get a list of variables whose name match a pattern, use the `%` wildcard character in a `LIKE` clause:

```
mysql> SHOW GLOBAL VARIABLES LIKE '%size%';
```

Shape Dependent System Variables

Shape dependent global system variables are linked to, and tuned for, a specific MySQL shape. See Supported Shapes. It is not possible to edit them.

`innodb_parallel_read_threads` is also a session variable. See: Session Variables.

Table 13-10 MySQL.2.16GB Global System Variables

Name	Default Value
back_log	1000

Table 13-10 (Cont.) MySQL.2.16GB Global System Variables

Name	Default Value
innodb_page_cleaners	4
innodb_parallel_read_threads	1
innodb_read_io_threads	1
innodb_redo_log_capacity	2
replica_parallel_workers > 0	4
temptable_max_ram	1073741824
thread_pool_size	8

Table 13-11 MySQL.4.32GB Global System Variables

Name	Default Value
back_log	2000
innodb_page_cleaners	4
innodb_parallel_read_threads	2
innodb_read_io_threads	2
innodb_redo_log_capacity	4
replica_parallel_workers > 0	8
temptable_max_ram	1073741824
thread_pool_size	16

Table 13-12 MySQL.8.64GB Global System Variables

Name	Default Value
back_log	4000
innodb_page_cleaners	4
innodb_parallel_read_threads	4
innodb_read_io_threads	4
innodb_redo_log_capacity	8
replica_parallel_workers > 0	16
temptable_max_ram	2147483648
thread_pool_size	32

Table 13-13 MySQL.32.256GB Global System Variables

Name	Default Value
back_log	2000
innodb_page_cleaners	8
innodb_parallel_read_threads	32
innodb_read_io_threads	2
innodb_redo_log_capacity	8
replica_parallel_workers > 0	16

Table 13-13 (Cont.) MySQL.32.256GB Global System Variables

Name	Default Value
temptable_max_ram	1073741824
thread_pool_size	32

Session Variables

Session variables remain in effect during the session until the variable changes or the session ends. The change has no effect on other sessions. For new connections, a session variable value is initialized with the corresponding global system variable value. Many of the following session variables are also available as user configurable variables. See: User Configurable System Variables.

To assign a value to a session variable, precede the variable name with the `SESSION` or `LOCAL` keyword, or with the `@@SESSION.`, `@@LOCAL.`, or `@@` qualifier, or with no keyword or modifier. For example:

```
mysql> SET SESSION sql_mode = 'TRADITIONAL';
mysql> SET LOCAL sql_mode = 'TRADITIONAL';
mysql> SET @@SESSION.sql_mode = 'TRADITIONAL';
mysql> SET @@LOCAL.sql_mode = 'TRADITIONAL';
mysql> SET @@sql_mode = 'TRADITIONAL';
mysql> SET sql_mode = 'TRADITIONAL';
```

Table 13-14 User Settable Session Variables

Name	Default Value
autocommit	ON
big_tables	OFF
block_encryption_mode	aes-128-ecb
character_set_client	utf8mb4
character_set_connection	utf8mb4
character_set_results	utf8mb4
character_set_server	utf8mb4
collation_connection	utf8mb4_0900_ai_ci
collation_database	utf8mb4_0900_ai_ci
collation_server	utf8mb4_0900_ai_ci
completion_type	NO_CHAIN
cte_max_recursion_depth	1000
default-storage-engine	InnoDB
--default_tmp_storage_engine	InnoDB
default_week_format	0
div_precision_increment	4
end_markers_in_json	OFF

Table 13-14 (Cont.) User Settable Session Variables

Name	Default Value
eq_range_index_dive_limit	200
foreign_key_checks	ON
generated_random_password_length	20
group_concat_max_len	1024
group_replication_consistency	BEFORE_ON_PRIMARY_FAILOVER
information_schema_stats_expiry	86400
innodb_ddl_buffer_size	1048576
innodb_ddl_threads	4
innodb_ft_enable_stopword	ON
innodb_ft_user_stopword_table	NULL
innodb_lock_wait_timeout	50
innodb_parallel_read_threads	Shape dependent. See Shape Dependent System Variables.
internal_tmp_mem_storage_engine	TempTable
join_buffer_size	262144
lc_messages	en_US
lc_time_names	en_US
lock_wait_timeout	86400
long_query_time	10
max_allowed_packet	67108864
max_execution_time	0
max_heap_table_size	16777216
max_join_size	18446744073709551615
max_length_for_sort_data	4096
max_points_in_geometry	65536
max_seeks_for_key	18446744073709551615
max_sort_length	1024
mysqlx_max_allowed_packet	67108864
mysqlx_read_timeout	30
mysqlx_wait_timeout	28800
mysqlx_write_timeout	60
net_buffer_length	16384
net_read_timeout	30
net_retry_count	10
net_write_timeout	60
new	OFF
old_alter_table	OFF
optimizer_prune_level	1
optimizer_search_depth	62

Table 13-14 (Cont.) User Settable Session Variables

Name	Default Value
optimizer_switch	index_merge_sort_union=on, index_merge_intersection=on, engine_condition_pushdown=on, index_condition_pushdown=on, mrr=on, mrr_cost_based=on, block_nested_loop=on, batched_key_access=off, materialization=on, semijoin=on, loosescan=on, firstmatch=on, duplicateweedout=on, subquery_materialization_cost_based=on, use_index_extensions=on, condition_fanout_filter=on, derived_merge=on, use_invisible_indexes=off, skip_scan=on, hash_join=on, subquery_to_derived=off, prefer_ordering_index=on, hypergraph_optimizer=off, derived_condition_pushdown=on
optimizer_trace	enabled=off,one_line=off
optimizer_trace_features	greedy_search=on, range_optimizer=on, dynamic_range=on, repeated_subselect=on
optimizer_trace_limit	1
optimizer_trace_max_mem_size	1048576
optimizer_trace_offset	-1
parser_max_mem_size	18446744073709551615
print_identified_with_as_hex	OFF
pseudo_replica_mode	OFF
pseudo_slave_mode	OFF
query_alloc_block_size	8192
query_prealloc_size	8192
range_alloc_block_size	4096
range_optimizer_max_mem_size	8388608
rbr_exec_mode	STRICT
read_buffer_size	131072
read_rnd_buffer_size	262144
resultset_metadata	FULL
secondary_engine_cost_threshold	100000.000000
session_track_gtids	OFF
session_track_schema	ON
session_track_state_change	OFF
session_track_system_variables	262144
session_track_transaction_info	OFF
show_create_table_skip_secondary_engine	OFF

Table 13-14 (Cont.) User Settable Session Variables

Name	Default Value
show_create_table_verbosity	OFF
sort_buffer_size	262144
sql_auto_is_null	OFF
sql_big_selects	ON
sql_buffer_result	OFF
sql_mode	ONLY_FULL_GROUP_BY, STRICT_TRANS_TABLES, NO_ZERO_IN_DATE, NO_ZERO_DATE, ERROR_FOR_DIVISION_BY_ZERO, NO_ENGINE_SUBSTITUTION
sql_notes	ON
sql_quote_show_create	ON
sql_safe_updates	OFF
sql_select_limit	18446744073709551615
sql_warnings	OFF
thread_pool_high_priority_connection	0
time_zone	UTC
tmp_table_size	16777216
transaction_alloc_block_size	8192
transaction_isolation	REPEATABLE-READ
transaction_prealloc_size	4096
transaction_read_only	OFF
unique_checks	ON
updatable_views_with_limit	YES
use_secondary_engine	ON
wait_timeout	28800
windowing_use_high_precision	ON

14

User and Group Management

MySQL HeatWave on AWS uses predefined OCI IAM groups and policies to manage access to the MySQL HeatWave Console. Predefined groups and policies are created when the service is provisioned. Defining your own groups and policies for MySQL HeatWave on AWS is currently not supported.

An Administrator grants access to the MySQL HeatWave Console by adding users to the predefined OCI IAM groups. User management is performed in the OCI Console. The policies associated with each group determine which resources that users can access and the permissions associated with those resources. MySQL HeatWave on AWS resources include `DB Systems`, `DB System Backups`, and `HeatWave Clusters`.

> **Note:**
>
> MySQL HeatWave on AWS supports federation with third-party Identity Providers (IdPs). For more information, see Federating with Identity Providers, in the Oracle Cloud Infrastructure documentation.

Groups and Permissions

MySQL HeatWave on AWS has three predefined groups. The groups are created in the OCI **Default** identity domain. The predefined groups and associated permissions are described in the following tables.

- OracleMySQLHeatwaveDBUsers Group
- OracleMySQLHeatwaveDBAdmin Group
- OracleMySQLHeatwaveServiceAccountAdmin Group

Table 14-1 OracleMySQLHeatwaveDBUsers Group

Group Description	Resources and Permissions
OracleMySQLHeatwaveDBUsers: Members of this group can use DB Systems and HeatWave Clusters resources.	DB Systems • View Supported Shapes • View DB Systems • View DB System Details • Start DB Systems • Restart DB Systems • Update DB Systems • Stop DB Systems • Run Queries • View Query Status • Stop Queries DB System Backups • Create DB System Backups HeatWave Clusters • View HeatWave Clusters • View HeatWave Cluster Details • Create HeatWave Clusters • Start HeatWave Clusters • Stop HeatWave Clusters

Table 14-2 OracleMySQLHeatwaveDBAdmin Group

Group Description	Resources and Permissions
OracleMySQLHeatwaveDBAdmin: Members of this group can manage all aspects of DB Systems, DB System Backups, and HeatWave Clusters resources.	In addition to *OracleMySQLHeatwaveDBUsers* group permissions, this group has these permissions: DB Systems • Create DB Systems • Delete DB Systems DB System Backups • View DB System Backups • View DB System Backup Details • Delete DB System Backups HeatWave Clusters • Delete HeatWave Clusters

Table 14-3 OracleMySQLHeatwaveServiceAccountAdmin Group

Group Description	Resources and Permissions
OracleMySQLHeatwaveServiceAccountAdmin: Members of this group can manage all aspects of DB Systems, DB System Backups, and HeatWave Clusters resources.	This group has the same permissions as the *OracleMySQLHeatwaveDBAdmin* group.

> **Note:**
>
> The OCI user account that registered for the MySQL HeatWave on AWS service is added to the `OracleMySQLHeatwaveServiceAccountAdmin` group when the service is provisioned.

Groups and Policies

The predefined groups used to manage MySQL HeatWave Console access, described in Groups and Permissions, are created in the OCI **Default** identity domain. For information, see https://docs.oracle.com/en-us/iaas/Content/Identity/domains/overview.htm#the_default_domain, in the *Oracle Cloud Infrastructure Documentation*.

The predefined groups all start with `OracleMySQLHeatWave`. Policies are defined for each predefined group which enable the MySQL HeatWave on AWS resource permissions outlined in Groups and Permissions.

> **NOT_SUPPORTED:**
>
> The predefined `OracleMySQLHeatWave` groups and policies are static. You must not modify them, add to them, or remove them. If you do delete one of these groups, MySQL HeatWave on AWS will stop working. In this case, submit a My Oracle Support ticket to get the group re-added to your account, specifying " MySQL HeatWave on AWS " as the product.

Because the predefined groups are created in OCI, you can define further OCI policies using these groups to provide access to other OCI resources. For example, you can create an OCI policy that allows members of the `OracleMySQLHeatwaveDBUsers` group to access resources in an OCI tenancy. For information about OCI policies, see How Policies Work.

User Management

User management is performed in the Oracle Cloud Infrastructure (OCI) Console.

MySQL HeatWave on AWS uses predefined OCI IAM groups created in the OCI **Default** identity domain to manage access to the MySQL HeatWave Console, as described in Groups and Permissions. A MySQL HeatWave on AWS Administrator manages the users that belong to those groups.

To access the Oracle Identity Cloud Service from the Oracle Cloud Infrastructure (OCI) Console, open the navigation menu and click **Identity & Security**.

Alternatively, to access the Oracle Identity Cloud Service from the MySQL HeatWave Console:

1. Sign into the MySQL HeatWave Console as an Administrator. For sign-in instructions, see Signing In.
2. From the profile menu, select **Administration**.
3. Select **Identity Service**.

Refer to Managing Users, in the *Oracle Cloud Infrastructure Documentation*, for the following user management procedures.

- Creating a user
- Editing a user
- Resetting a user's password
- Deleting a user

Refer to Managing Groups, in the *Oracle Cloud Infrastructure Documentation*, for the following group-related user management procedures.

> **Note:**
>
> The predefined groups used by MySQL HeatWave on AWS are created in the OCI **Default** identity domain. For more information, see The Default Identity Domain, in the *Oracle Cloud Infrastructure Documentation*.

- Adding users to groups
- Removing users from groups

To add another user with Oracle Cloud Administrator permissions, see Add a User with Oracle Cloud Administrator Permissions, in the *Oracle Cloud Infrastructure Documentation*.

15
Account Management

This chapter discusses region management, service limits, and billing for MySQL HeatWave on AWS.

Manage Regions

The AWS region is shown at the top of the MySQL HeatWave Console. To change regions, click on ***AWS region name*** , and select the preferred region.

To add a new region:

1. Click on ***AWS region name*** , and select **Manage Regions**.
2. Subscribe to an OCI region.
   a. Click **Subscribe** next to the OCI region that maps to the preferred AWS region.

      If a region limit increase is required, do the following to request an increase:

      i. From the **Limit increase needed** dialog box, click **Open Service Options**.

      ii. Follow these steps: Requesting a Limit Increase to the Subscribed Region Count.

      iii. When the region limit increase is complete, click **Subscribe** next to the OCI region that maps to the preferred AWS region.

   b. From the **Subscribe to New Region** dialog box, read the privacy notice, and then click **Subscribe**.
3. Enable the MySQL HeatWave on AWS service.
   a. Click **Enable** next to the preferred AWS region.
   b. From the **Enable Service** dialog box, click **Enable**.

Service Limits

A service limit is the quota or allowance set on a resource. MySQL HeatWave on AWS default service limits are shown below. Service limits are per region unless explicitly specified.

To request a service limit increase, submit a My Oracle Support ticket, specifying " MySQL HeatWave on AWS " as the product.

Table 15-1 Service Limits

Resource	Default Limit
MySQL Database Block Storage	10 TB
MySQL Database Manual Backups	40 (for all DB System instances)
MySQL.32.256GB instances	2
MySQL.8.64GB instances	1

Table 15-1 (Cont.) Service Limits

Resource	Default Limit
MMySQL.4.32GB instances	1
MySQL.2.16GB instances	4

Billing

For information about managing service costs, see Billing, Cost Management, and Payments Overview.

Billing for MySQL HeatWave on AWS is managed in the OCI Console.

To access billing information in the Oracle Cloud Infrastructure (OCI) Console, open the navigation menu and select **Billing & Cost Management**.

Alternatively, to access billing information from the MySQL HeatWave Console,:

1. Sign into the MySQL HeatWave Console as an Account Administrator.
2. From the profile menu, select **Administration**.
3. Select **Billing**.

16 Maintenance

This section describes the maintenance of DB Systems and HeatWave.

Essential patching and maintenance is an automatic process that may include patching the underlying operating system, updating the MySQL Server version, and updating underlying hardware. Maintenance is initiated during the maintenance window defined when creating a DB System. The maintenance window start day and time can be viewed on the **DB System Details** page in the MySQL HeatWave on AWS Console. The maintenance window is a two hour period during which maintenance is initiated. The time required to apply patches and updates may extend beyond the maintenance window and require DB System restarts. See Viewing DB System Details.

To change a DB System's Maintenance Window start day or start time, see Edit DB System.

When maintenance is performed, your DB System's status changes to `UPDATING` and the DB System may be unavailable for a short time while the maintenance completes.

> **Note:**
>
> DB Systems maintenance is performed when the DB System is in an `ACTIVE` state or an `INACTIVE` state, and the DB System is returned to the same state after the maintenance completes.

17

Release Notes

Release notes for MySQL HeatWave on AWS

Release Notes for MySQL HeatWave on AWS (2023-08-08, General Availability)

- You can now increase the data storage size of an active DB System online. When the DB System is active and healthy, updating the data storage size of the DB System does not restart the DB System, and you can continue to query it while the storage is being increased. You get elasticity without compromising uptime or performance.
  For more information, see Increasing DB System Storage.

Release Notes for MySQL HeatWave on AWS (2023-07-31, General Availability)

- MySQL HeatWave on AWS now supports auto error recovery. Whenever a HeatWave node fails because of a hardware or a software issue, the cluster becomes unhealthy and error recovery is triggered. During the recovery process, HeatWave automatically attempts to bring the node online and reload data that was previously loaded. This reduces manual intervention and improves service uptime.
  For more information, see HeatWave Cluster Failure and Recovery.

Release Notes for MySQL HeatWave on AWS (2023-07-13, General Availability)

- MySQL HeatWave on AWS is now available in the AWS Europe (Frankfurt) and Europe (London) region. See Region Availability.
- Fast data reload is now available when you pause and resume HeatWave on AWS. Besides storing the HeatWave formatted data in-memory, HeatWave also stores the formatted data in HeatWave storage layer on AWS S3. This enables reload of data in constant time regardless of data size. This capability is now available when you pause and resume the HeatWave cluster. You can now pause the HeatWave cluster when you don't need it to save cost, and resume the cluster very fast when you want to use HeatWave for query acceleration.

Release Notes for MySQL HeatWave on AWS (2023-06-08, General Availability)

- MySQL HeatWave on AWS is now available in the AWS Asia Pacific (Mumbai) region. See Region Availability.

Release Notes for MySQL HeatWave on AWS (2023-05-04, General Availability)

- MySQL HeatWave on AWS is now available in the AWS Asia Pacific (Tokyo) region. See Region Availability.

Release Notes for MySQL HeatWave on AWS (2023-04-27, General Availability)

- HeatWave Scale-out Data Management on AWS S3
  MySQL HeatWave on AWS now provides an optimized storage layer built on AWS S3 to store the HeatWave in-memory hybrid columnar representation of the data. This allows data to be reloaded to each HeatWave node independently and in parallel. This significantly improves the service uptime and performance of operations such as error recovery, maintenance, and system restart. See HeatWave Architectural Features.
- MySQL Autopilot: Auto Error Recovery from MySQL failure
  With Auto Error Recovery, now when MySQL fails and restarts, the HeatWave Cluster automatically restarts, identifies the tables which were loaded prior to the failure, and reloads those tables automatically from MySQL. This reduces the intervention by the user and also improves service uptime.
- Auto reload of data in HeatWave Cluster after MySQL upgrade
  HeatWave now automatically reloads data from MySQL InnoDB after a MySQL node restarts due to maintenance upgrades or planned restarts. With auto-reload capability, there are no more manual steps after maintenance or a restart operation. This reduces the operational overhead and improves service availability.
- MySQL Autopilot for OLTP - Auto Shape Prediction
  Auto shape prediction collects the most recent query execution metrics and uses advanced machine learning models to predict the MySQL database instance shape for optimal transactional processing performance. Auto shape prediction continuously monitors OLTP workload to provide a suggestion that will adapt to evolving workload patterns, allowing the MySQL DB System to maintain the best OLTP price performance over time. See Autopilot Shape Advisor.

  Monitor MySQL statistics such as buffer pool usage, workload activity and access patterns, and recommendations for the optimal MySQL shape for the workload with MySQL HeatWave Console.
- MySQL Configuration
  Use MySQL HeatWave Console to view and configure user, system, initialization, and service-specific variables for the MySQL DB System shape with or without HeatWave support. Select a default configuration or create a custom configuration for the DB system. See and Configuration.
- Automatic Backup
  MySQL HeatWave on AWS now support automatic backups. Choose a time for automatic backups during the creation of a MySQL DB System. The retention period can be between 1 and 35 days. The default retention period is 7 days.

Scheduled backups are deleted when the DB System is deleted. See Creating a DB System.