MySQL HeatWave on AWS Service Guide



F82097-55



MySQL HeatWave on AWS Service Guide ,

F82097-55

Copyright © 2022, 2025, Oracle and/or its affiliates.

Contents

1 Overview

1.1 MySQL HeatWave 1
1.2 MySQL Server 1
1.3 Integration with Oracle Cloud Infrastructure (OCI) 2
1.4 Region Availability 2
1.5 Identity and Access Management 3
1.6 Security 3

2 Getting Started

2.1 Accessing MySQL HeatWave on AWS 1
2.2 Signing Up 1
2.2.1 Sign-up Overview 1
2.2.2 Sign-up Procedure 2
2.3 Signing In 5
2.4 Using the Console 5
2.4.1 Console Overview 6

3 DB Systems

3.1 About DB Systems 1
3.1.1 Supported Shapes 2
3.1.2 MySQL Server 2
3.1.2.1 MySQL Server Versions 2
3.1.2.2 Server Upgrades 5
3.1.2.3 Server Error Logging 8
3.1.2.4 Unsupported MySQL Server Features 9
3.1.2.5 MySQL Storage Engines 10
3.1.2.6 Plugins and Components 10
3.1.2.7 MySQL Enterprise Audit 11
3.1.2.8 MySQL HeatWave on AWS Service Restrictions 14
3.1.2.9 Default MySQL Privileges 14
3.1.2.10 Reserved User Names 16
3.2 Launching a Starter DB System 16

Copyright © 2022, 2025, Oracle and/or its affiliates.

3.3 Creating a DB System 17
3.4 Managing a DB System 22
3.4.1 Stopping, Starting, or Restarting a DB System 23
3.4.2 Enable or Disable High Availability 23
3.4.3 Editing a DB System 23
3.4.4 Update Networking 25
3.4.5 Update MySQL Configuration 25
3.4.6 Update MySQL Shape 26
3.4.7 Increasing DB System Storage 28
3.4.8 Update Monitoring Configuration 29
3.4.9 Upgrade MySQL Version 30
3.4.10 Deleting a DB System 30
3.5 Viewing DB System Details 31
3.5.1 MySQL DB System Details 31

4 MySQL HeatWave Clusters

4.1 Creating a MySQL HeatWave Cluster 1
4.1.1 Estimating Cluster Memory Size with MySQL HeatWave Autopilot 3
4.2 Managing a MySQL HeatWave Cluster 4
4.2.1 Starting, Stopping, or Restarting a MySQL HeatWave Cluster 4
4.2.2 Editing a MySQL HeatWave Cluster 5
4.2.3 Deleting a MySQL HeatWave Cluster 6
4.3 Viewing MySQL HeatWave Cluster Details 6
4.3.1 MySQL HeatWave Cluster Details 7
4.4 MySQL HeatWave Cluster Data Recovery 12

5 Connecting to a DB System

5.1 Connecting from the Console 1
5.2 Connecting from a Client 1
5.2.1 Connecting with MySQL Shell 2
5.2.2 Connecting with MySQL Command-Line Client 3
5.2.3 Connecting with MySQL Workbench 4
5.3 Connecting Using a PrivateLink 5
5.4 MySQL Connectors 5
5.5 Enabling Host Name Identity Verification 5

6 PrivateLink

6.1 Deploying a Query PrivateLink 1
6.1.1 Creating a Query PrivateLink 2

Copyright © 2022, 2025, Oracle and/or its affiliates.

6.1.2 Configuring IAM Policies for Endpoints for a Query PrivateLink 3
6.1.3 Creating an Endpoint for a Query PrivateLink 4
6.1.4 Connecting to a DB System With a PrivateLink 5
6.1.5 Troubleshooting a Query PrivateLink 6
6.2 Deploying an Egress PrivateLink 8
6.2.1 Configuring the Network Infrastructure for an Egress PrivateLink 9
6.2.2 Creating an Egress PrivateLink 12
6.2.3 Troubleshooting an Egress PrivateLink 13
6.3 Managing a PrivateLink 15
6.3.1 Editing a PrivateLink 16
6.3.2 Deleting a PrivateLink 16
6.3.3 Updating Authorized Principals for a Query PrivateLink 16
6.3.4 Updating Egress Endpoints for a Egress PrivateLink 17
6.3.5 Viewing PrivateLink Details 18
6.3.5.1 PrivateLink Details 19
6.4 Billing for PrivateLinks 22
6.5 PrivateLink Limitations 22

7 Importing Data

7.1 Exporting Data 1
7.1.1 About MySQL Shell 2
7.1.2 MySQL Server Compatibility 2
7.1.3 Exporting Data Using MySQL Shell 4
7.2 Importing Data 5
7.2.1 Data Import Feature 5
7.2.1.1 Importing Sample Database 5
7.2.1.2 Importing Data Using the Data Import Feature 6
7.2.1.3 Viewing Data Import Details 8
7.2.2 Bulk Ingest Feature 11
7.2.2.1 Granting Privileges to Bulk Ingest Data From Amazon S3 12
7.2.2.2 Importing Data Using the Bulk Ingest Feature 12
7.2.2.3 Bulk Ingest Limitations 13
7.2.3 Dump Loading Utility 14
7.2.3.1 Importing Data Using the Dump Loading Utility 14

8 Inbound Replication

8.1 About Inbound Replication 1
8.2 Configuring Inbound Replication 2
8.2.1 Configuring Inbound Replication over the Public Internet 2
8.2.2 Configuring Inbound Replication over an Egress PrivateLink 2

Copyright © 2022, 2025, Oracle and/or its affiliates.

8.3 Source Configuration 2
8.4 Creating a Replication User On a Source Server 3
8.5 Creating a Channel 4
8.5.1 Channel Filter Rules for Inbound Replication 9
8.6 Managing Replication Channels 10
8.6.1 Disabling or Enabling a Channel 10
8.6.2 Editing a Channel 11
8.6.3 Resuming a Channel 12
8.6.4 Resetting a Channel 12
8.6.5 Deleting a Channel 13
8.7 Viewing Channel Details 13
8.7.1 Channel Details 13
8.8 Limitations 17

9 Manage Data in MySQL HeatWave

9.1 Loading or Unloading Data into MySQL HeatWave Cluster 1
9.1.1 Unload Advisor 3
9.2 Creating Lakehouse Mapping 4
9.3 Refreshing Lakehouse Data 7
9.4 Maximum Number of Tables Loadable into a MySQL HeatWave Cluster 8

10 Running Queries

10.1 Running MySQL HeatWave Queries 1

11 Autopilot Index Advisor

11.1 Running Autopilot Index Advisor 1

12 High Availability

12.1 Overview of High Availability 1
12.2 Prerequisites 2
12.3 Enabling and Disabling High Availability 3
12.3.1 Enabling high availability on a DB System 4
12.3.2 Disabling high availability on a DB System 5
12.4 Failover 5
12.5 Switchover 6
12.5.1 Switchover Using the Console 7
12.6 Maintenance of a High Availability DB System 8
12.7 Restoring a Backup of a High Availability DB System 8

Copyright © 2022, 2025, Oracle and/or its affiliates.

12.8 Inbound Replication to a High Availability DB System 9
12.9 Limitations for High Availability 9
12.10 Billing 10

13 MySQL HeatWave AutoML

13.1 MySQL HeatWave AutoML Requirements 1
13.2 Create a MySQL HeatWave AutoML model 1
13.3 Evaluate a MySQL HeatWave AutoML model 2

14 MySQL HeatWave GenAI

14.1 MySQL HeatWave GenAI Requirements 1
14.2 Supported LLMs, Embedding Models, and Languages 2
14.2.1 View Available Models 2
14.2.2 MySQL HeatWave GenAI In-Database LLMs 3
14.2.3 MySQL HeatWave GenAI External LLMs 4
14.2.4 MySQL HeatWave GenAI In-databse Embedding Models 4
14.2.5 Supported Languages 5
14.3 MySQL HeatWave GenAI In-database Vector Store 6
14.4 MySQL HeatWave GenAI Routines 7

15 System Variables

15.1 System Variables 1

16 Events

17 Monitoring

17.1 CloudWatch Monitoring 1
17.1.1 CloudWatch Monitoring Metrics 2
17.1.2 Cost for Using CloudWatch Monitoring 7
17.2 Performance Tabs 8
17.2.1 Metrics for MySQL and MySQL HeatWave Cluster Performance 8
17.2.1.1 Details for the MySQL and MySQL HeatWave Cluster Performance Metrics 8
17.2.2 Workload Performance 9
17.2.2.1 MySQL HeatWave Workload Performance Data 9
17.3 Autopilot Shape Advisor 10
17.3.1 Autopilot Shape Advisor with MySQL HeatWave Console 11

Copyright © 2022, 2025, Oracle and/or its affiliates.

17.3.1.1 Auto Shape Prediction Data 11
17.3.2 Autopilot Shape Advisor with a MySQL Client 13

18 Backups

18.1 Creating a Backup 1
18.2 Editing a Backup 2
18.3 Viewing Backup Details 2
18.3.1 Backup Details 2
18.4 Restoring a Backup to a New DB System 4
18.5 Deleting a Backup 8
18.6 Billing and Free Quota for Backup Storage 8

19 Configuration

19.1 Creating a MySQL Configuration 1
19.2 Copying a MySQL Configuration 3
19.3 MySQL Configuration Details 6
19.4 System Initialization Variables 9
19.5 User-Configurable System Variables 9
19.6 User-Configurable Shape-Dependent System Variables 13
19.7 Service-Specific System Variables 16
19.8 Shape-Dependent System Variables 17
19.9 Session Variables 18

20 User and Group Management

20.1 Groups and Permissions 1
20.2 Groups and Policies 3
20.3 User Management 4

21 Account Management

21.1 Manage Regions 1
21.2 Service Limits 1
21.3 Billing 2
21.4 Viewing OCID of the Tenancy 3
21.5 Manage AWS Access 3
21.5.1 Creating an IAM Policy to Access an Amazon S3 Bucket 4
21.5.2 Creating an IAM Role to Access an Amazon S3 Bucket 6
21.5.3 Creating an IAM Policy to Access Amazon Bedrock LLMs 8
21.5.4 Creating an IAM Role to Access Amazon Bedrock LLMs 9

Copyright © 2022, 2025, Oracle and/or its affiliates.

21.5.5 Creating an IAM Role and Policy for CloudWatch Monitoring 10

22 Maintenance

23 Release Notes

Copyright © 2022, 2025, Oracle and/or its affiliates.

Preface and Legal Notices

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

Legal Notices

Copyright © 2022, 2025, Oracle and/or its affiliates.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software" or "commercial computer software documentation" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle, Java, and MySQL are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement 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.

Copyright © 2022, 2025, Oracle and/or its affiliates.

1 Overview

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

- MySQL HeatWave
- MySQL Server
- Integration with Oracle Cloud Infrastructure (OCI)
- Region Availability
- Identity and Access Management
- Security

1.1 MySQL HeatWave

MySQL 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, you can provision DB Systems and MySQL HeatWave Clusters.

MySQL HeatWave consists of a DB System and one or more MySQL HeatWave nodes that form a MySQL HeatWave Clusters. The DB System is responsible for cluster management, query scheduling, and returning query results. MySQL HeatWave nodes store data in memory and process analytics queries.

When a MySQL HeatWave Cluster is enabled, queries that meet certain prerequisites are automatically offloaded from the DB System to the MySQL 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 MySQL HeatWave Cluster by connecting to the DB System.

> **Tip**
>
> This *MySQL HeatWave on AWS Service Guide* covers what you need to know to set up and use DB Systems and MySQL HeatWave Clusters on AWS, mainly using the MySQL HeatWave Console. For more detailed information about using MySQL 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.

1.2 MySQL Server

The MySQL Server on MySQL HeatWave on AWS 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 Server that

Copyright © 2022, 2025, Oracle and/or its affiliates.

supports MySQL HeatWave. For MySQL Server documentation, refer to the MySQL Reference Manual .

You can choose from the latest MySQL releases when creating a DB System. See Creating a DB System.

1.3 Integration with Oracle Cloud Infrastructure (OCI)

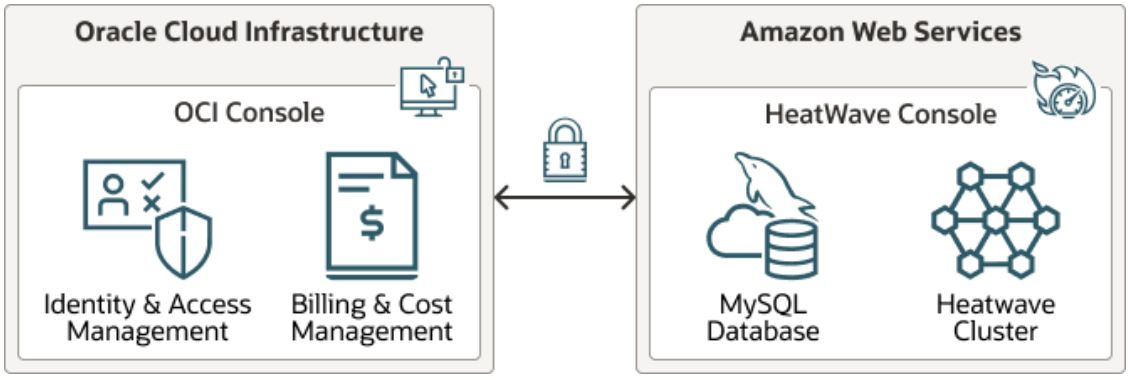
MySQL HeatWave on AWS is integrated with Oracle Cloud Infrastructure (OCI) for identity and access management, and for billing and cost management.

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



1.4 Region Availability

MySQL HeatWave on AWS is currently available in the regions listed in Table 1-1. Each AWS Region maps to an OCI region, and requires a subscription to that OCI region.

Table 1-1 OCI and AWS Regions

AWS Region name	AWS Region code	OCI Region name	OCI Region identifier
Asia Pacific (Mumbai)	ap-south-1	India West (Mumbai)	ap-mumbai-1
Asia Pacific (Seoul)	ap-northeast-2	South Korea Central (Seoul)	ap-seoul-1
Asia Pacific (Tokyo)	ap-northeast-1	Japan East (Tokyo)	ap-tokyo-1

Copyright © 2022, 2025, Oracle and/or its affiliates.

Table 1-1 (Cont.) OCI and AWS Regions

AWS Region name	AWS Region code	OCI Region name	OCI Region identifier
Europe (Frankfurt)	eu-central-1	Germany Central (Frankfurt)	eu-frankfurt-1
Europe (London)	eu-west-2	UK South (London)	uk-london-1
Mexico (Central)	mx-central-1	Mexico Central (Queretaro)	mx-queretaro-1
US East (N. Virginia)	us-east-1	US East (Ashburn)	us-ashburn-1
US West (Oregon)	us-west-2	US West (San Jose)	us-sanjose-1

Related Topics

- Subscribed Region Limits
- Enabling Host Name Identity Verification

1.5 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 MySQL 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.

1.6 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

Copyright © 2022, 2025, Oracle and/or its affiliates.

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 MySQL 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 over the public internet are encrypted. Private connections over PrivateLinks are also encrypted by default.

  For DB Systems that only accept private connections (i.e., not accessible from public IP addresses), if you need to disable encryption for your client connections (for example, due to certain client-side limitations), you can update the MySQL configuration of your DB System to set the system variable `require_secure_transport` to `OFF`, and have clients connect to the MySQL DB System through a PrivateLink.

  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

Copyright © 2022, 2025, Oracle and/or its affiliates.

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.

Copyright © 2022, 2025, Oracle and/or its affiliates.

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
- Signing Up
- Signing In
- Using the Console

2.1 Accessing MySQL HeatWave on AWS

Access MySQL HeatWave on AWS with the **MySQL HeatWave Console**. It is a browser-based interface available at https://cloud.mysql.com/.

The access requires that you have signed up for MySQL HeatWave on AWS. See Signing Up for details.

Once you have signed up, sign in to MySQL HeatWave Console. See Signing In for details. See Using the Console on how to use MySQL HeatWave Console.

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

2.2 Signing Up

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

- Sign-up Overview
- Sign-up Procedure

2.2.1 Sign-up Overview

The sign-up process for MySQL HeatWave on AWS consists of the following major steps

- Create an Oracle Cloud Free Tier Account, if you do not already have one.
- Enable the MySQL HeatWave on AWS service in your home region.

See Sign-up Procedure for detailed instructions.

Copyright © 2022, 2025, Oracle and/or its affiliates.

Free Trial

> **Note**
>
> Free trial for MySQL HeatWave on AWS is only available in your home region. If you have chosen a home region where the MySQL HeatWave on AWS service is not available, you will not be able to try MySQL HeatWave on AWS for free. Check Region Availability. Upgrade to a paid account to enable the MySQL HeatWave on AWS service in any supported regions.

Once signed up, you will receive free cloud credits you can use for up to 30 days to try out MySQL HeatWave on AWS. You can monitor your free cloud credit balance in the Oracle Cloud console; see Monitor the Credit Balance for Your Free Oracle Cloud Promotion for more information.

At the end of your free trial, your account enters a grace period, during which your usage of MySQL HeatWave on AWS are limited in the following ways:

- Your MySQL HeatWave on AWS resources are stopped.
- You cannot provision new MySQL HeatWave on AWS resources.
- You cannot restart the stopped resources, but you can view or delete them.

> **Warning**
>
> If you do not upgrade to a paid account before the end of grace period, all of your HeatWave on AWS resources will be deleted, which can result in data loss. To avoid potential service interruption, we recommend upgrading to a paid account several days before your free trial expires.

To upgrade to a paid account in the Oracle Cloud console, please refer to Managing Account Upgrades and Payment Method. The upgrade process can take up to days to complete. Oracle sends a confirmation email upon completion of the upgrade. Please allow up to 6 hours for MySQL HeatWave on AWS to recognize the completed account upgrade. Resources stopped during the grace period are not automatically restarted after upgrading to a paid account. **After upgrading, you must restart manually any stopped resources**.

See the Service Limits that apply during a free trial.

2.2.2 Sign-up Procedure

To sign up for MySQL HeatWave on AWS:

1. Create an Oracle Cloud Account (or proceed to Step 2 below if you already have one):
   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. In the verification email, click **Verify email** to verify the email address and continue the account setup process.

Copyright © 2022, 2025, Oracle and/or its affiliates.

e. Complete the **Account Information**, including **Password**, **Customer type**, **Cloud Account Name**, and **Home Region**. Click **Continue** after you are done.

> **Tip**
>
> An Oracle Cloud Account has a single region limit by default. For the **Home Region**, choose the OCI region that maps to your preferred AWS region (for which MySQL HeatWave on AWS is supported) to avoid the need for a region increase. See: Region Availability and Subscribed Region Limits for details.

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**. Your Oracle Cloud account is being set up, which might take a while. You will then be directed to the Oracle Cloud Sign In Page.

2. Sign in to the Oracle Cloud Console:

> **Note**
>
> 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 (which is the case if you have gone through Step 1 above) or have the following permissions:
>
> - `manage heatwave-registration`
> - `manage policies`
> - `manage domains`
> - `manage groups`
> - `read limits`
> - `use tenancies`
> - `inspect compartments`
>
> See Manage Identity Domain and Getting Started with Policies for more details on identity domains and policies on OCI.

a. On the Oracle Cloud Sign In page, enter your Oracle Cloud account name and click **Next**. Continue your sign in on the next page by providing your account credentials. After a successful sign in, you are directed to the **Oracle Cloud Console**.

b. Go to the MySQL HeatWave on AWS **Administration** page by doing one of the following:

- If you have gone through Steps 1 to sign up for an account, you should see on the **Get Started** tab the **Get started with HeatWave on AWS** banner. Click the **Go to service** button on it.
- If you already had an Oracle Cloud account and did not go through Step 1 before signing in to the **Oracle Cloud Console**, open the navigation menu on the top left and select **Databases**. Under **HeatWave on AWS**, click **Administration** to go to the MySQL HeatWave on AWS **Administration** page.

Copyright © 2022, 2025, Oracle and/or its affiliates.

3. Enable MySQL HeatWave on AWS service:

   On the **HeatWave on AWS Region Management** pane, if the OCI region that maps to your 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. In the **Enable HeatWave on AWS on <AWS Region>** dialog box, click **Enable**. The MySQL HeatWave on AWS Administration page shows the current status of the sign-up process. Wait for the process to finish.

   If the OCI region that maps to your preferred AWS region does NOT show **Subscribed**, follow these steps to subscribe to the desired OCI region (skip to step b if yours is already a paid account, and skip to step c if you already have sufficient region limits to subscribe to a new OCI region):

   a. Request an account upgrade

      i. On the MySQL HeatWave on AWS **Administration** page, click the **Request Upgrade** button.

      ii. On the **Upgrade** page, complete the payment information by clicking the **Add Payment Method** button and following the steps.

      iii. Unless you want to **Request a Sales Call** (click the corresponding button in that case), under **Pay as You Go**, select the account type (**Corporate** or **Individual**), agree to the **terms and conditions**, and click the **Upgrade your account** button.

      iv. From the **Upgrade Confirmation** dialog box, click **Continue**. The account upgrade might take some time.

   b. Request a region limit increase:

      i. From the OCI Console, Open the **Help** menu (), and under **Targeted help**, click **Request service limit increase**. The Support Options page opens.

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

   c. Subscribe to the preferred region and enable the MySQL HeatWave on AWS service:

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

      ii. On the **HeatWave on AWS Region Management** pane, click **Subscribe** next to the OCI region that maps to the preferred AWS region.

      iii. In the **Subscribe to <OCI Region>** dialog box, read the privacy notice, and then click **Subscribe**.

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

      v. In the **Enable HeatWave on AWS on <AWS Region>** dialog box, click **Enable**.

   > ⚠ **Caution**
   >
   > 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.

4. Start using MySQL HeatWave on AWS:

Copyright © 2022, 2025, Oracle and/or its affiliates.

On the MySQL HeatWave on AWS **Administration** page, click **Open HeatWave on AWS Console** on the **Welcome to HeatWave on AWS** banner. This will direct you to the MySQL HeatWave Console— for more information on it see Using the Console.

> **Note**
>
> The OCI user account that registered for the MySQL HeatWave on AWS service is added to the `OracleMySQLHeatwaveServiceAccountAdmin` user group for accessing MySQL HeatWave on AWS when the service is provisioned. Any other users must be added to at least one of the groups described in Groups and Permissions for access to the service.

2.3 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.
- Your user belongs to one of the user Groups described in Groups and Permissions. See also User and Groups and Management for more details.

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.

2.4 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.

Copyright © 2022, 2025, Oracle and/or its affiliates.

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

2.4.1 Console Overview

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

The profile menu at the top right corner of the MySQL HeatWave Console shows the account you are logged in to.

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

- **Home**

  This is the landing page where users are directed after signing in. It consists of the following regions:

  - The **Getting Started** region of the page provides a **Launch Starter DB System** button for Launching a Starter DB System.
  - The **Resource Summary** region shows the resource consumption. Resources listed include **Regions Subscribed**, **DB Systems**, **MySQL HeatWave Clusters**, **Backups**, **Storage**, and **PrivateLinks**.
  - The **Recent Activity** region lists the type, severity, and time for recent events.
  - The **What's New** region lists latest changes to MySQL HeatWave on AWS.

- **Resources**

  Displays the **Resources** page, where you can manage different resources like **DB Systems**, **MySQL HeatWave Clusters**, **Backups**, DB System **Configurations**, replication **Channels**, and **PrivateLinks** on different tabs.

- **Workspace**

  Displays the **Workspace** page, where you can connect to a DB System, run DB System and MySQL HeatWave queries using the **Query Editor**, manage MySQL HeatWave Cluster data, use MySQL HeatWave AutoML or MySQL HeatWave Autopilot features, and to monitor performance metrics and HeatWave workload.

  To perform any tasks on the **Workspace** page, you have to connect a DB System, after which a summary of the DB System as well as a **Disconnect** button is shown in the side-bar menu.

- **Monitoring**
  Displays the **Monitoring** page, where you can monitor, for example, the Events on the **Service Events** pane.

Copyright © 2022, 2025, Oracle and/or its affiliates.

Each of these pages of the MySQL HeatWave Console contains a sidebar menu for selecting different tabs of interest. The side-bar menu is collapsible into an icon menu by a toggle button.

Signing Out

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

Copyright © 2022, 2025, Oracle and/or its affiliates.

3
DB Systems

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

- About DB Systems
- Launching a Starter DB System
- Creating a DB System
- Managing a DB System
- Viewing DB System Details

3.1 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. 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

Copyright © 2022, 2025, Oracle and/or its affiliates.

- MySQL Server

3.1.1 Supported Shapes

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

Table 3-1 MySQL Shapes

Shape Name	ECPUs	vCPUs	Memory (GiB)
MySQL.2.16GB	1	2	16
MySQL.4.32GB	2	4	32
MySQL.8.64GB	4	8	64
MySQL.32.256GB	16	32	256

Table 3-2 MySQL HeatWave Node Shapes

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

- If you intend to use MySQL HeatWave AutoML functionality, the `MySQL HeatWave.256GB` node shape is recommended when creating a MySQL HeatWave Cluster. The `MySQL 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.
- Lakehouse is supported on the `MySQL HeatWave`.256GB shape node shape only.

3.1.2 MySQL Server

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

- MySQL Server Versions
- Server Upgrades
- Server Error Logging
- Unsupported MySQL Server Features
- MySQL Storage Engines
- Plugins and Components
- MySQL Enterprise Audit
- MySQL HeatWave on AWS Service Restrictions
- Default MySQL Privileges
- Reserved User Names

3.1.2.1 MySQL Server Versions

Copyright © 2022, 2025, Oracle and/or its affiliates.

The MySQL versioning model consists of two release tracks, Innovation and Long-Term-Support (LTS)/Bugfix, to suit your application and environment needs. Both of the release tracks are production-grade quality and supported by MySQL HeatWave Service.

- Server Versioning
- Supported MySQL Server Versions
- MySQL Versions Deprecation and Removal Policy in MySQL HeatWave Service
- MySQL Version Support Schedule in MySQL HeatWave Service

3.1.2.1.1 Server Versioning

The MySQL Server included in MySQL HeatWave on AWS uses a versioning system consisting of three numbers, an update version, and the `-cloud` suffix. Using `8.4.0-u2-cloud` as an example:

- *First number*: The major release number (`8` in the example).
- *Second number*: The minor release number (`4` in the example).
- *Third number*: The patch number (`0` in the example). This is incremented for each new release.
- u*N* : The HeatWave-specific update number (`u2` in the example). 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 its typical output:

```
mysql> SELECT @@version;
+-----------------+
| @@version       |
+-----------------+
| 8.4.0-u2-cloud  |
+-----------------+
```

3.1.2.1.2 Supported MySQL Server Versions

MySQL server versions are divided into the Innovation track and Long-Term-Support (LTS)/ BugFix tracks:

Innovation Track

If you desire access to the latest features and improvements and enjoy staying on top of the latest technologies, then MySQL innovation track is the best fit for you. It is ideal for developers and database administrators working in fast-paced environments with high levels of automated tests and modern continuous integration techniques for faster upgrade cycles. MySQL 8.1.0 is the first innovation release, and the latest innovation release is found in the 9.x.y family. If you want the latest features, improvements, and all bug fixes for the MySQL database, use the innovation release track.

LTS/BugFix Track

Copyright © 2022, 2025, Oracle and/or its affiliates.

LTS/Bugfix track contains only necessary fixes, so it is a better choice if your environment requires established database software behavior. Features can be removed and added only in the first version of a LTS release (such as 8.4.0 LTS) but not later versions (such as 8.4.1 LTS and 8.4.2 LTS) in the same LTS release.

Related Topics

- MySQL Releases: Innovation and LTS

3.1.2.1.3 MySQL Versions Deprecation and Removal Policy in MySQL HeatWave Service

New versions and updates: MySQL HeatWave on AWS begins supporting new MySQL Server versions after they are released as Generally Available (GA). Customers are advised to regularly update their DB systems on both the Innovation or LTS/Bugfix tracks to the latest available version on MySQL HeatWave on AWS. Regular updates ensure access to the latest features, bug fixes, and security patches. To update the MySQL Server version of a DB system, see Upgrade MySQL Version.

Deprecation: Each MySQL version in the LTS/Bugfix track will be deprecated about one year after its GA date while each MySQL version in the Innovation track will be deprecated about three months after its GA date. When you are running a deprecated version of MySQL Server on your DB System, the **MySQL version** on the **Details** page of your DB System is marked with the **Deprecated** label. We strongly recommend updating and testing your DB system when its MySQL server version has been deprecated.

While you cannot choose a deprecated server version when creating a new DB System , you can restore the backup of a deprecated server version into a new DB System.

Removal: About three months after the deprecation date, the MySQL server version will become unavailable. Any DB system that is still running an unavailable version will be upgraded automatically in its next maintenance window. It is upgraded to the lowest version currently available in the same track on MySQL HeatWave on AWS. For example, when version 8.4.3 LTS becomes unavailable, it will be upgraded to 8.4.4 LTS, and when version 9.2.1 Innovation becomes unavailable, it will be upgraded to 9.2.2 Innovation.

3.1.2.1.4 MySQL Version Support Schedule in MySQL HeatWave Service

Each MySQL version is released on its **Available Date** on MySQL HeatWave on AWS and is deprecated sometime before it is made unavailable.

> **Note**
>
> Dates indicated by only month and year are preliminary and will be updated to specific dates as soon as they are confirmed. The mentioned dates are exclusive to the cloud service version. For details regarding MySQL Community, Enterprise, and CGE editions, please refer to the MySQL release notes documentation.

Table 3-3 MySQL HeatWave Service 8.0 Release Schedule

Version	Available Date	Deprecation Date	Unavailable Date
8.0.31	October 7, 2022	May 2025	August 2025
8.0.32	February 23, 2023	May 2025	August 2025
8.0.33	June 7, 2023	May 2025	August 2025

Copyright © 2022, 2025, Oracle and/or its affiliates.

Table 3-3 (Cont.) MySQL HeatWave Service 8.0 Release Schedule

Version	Available Date	Deprecation Date	Unavailable Date
8.0.42	April 23, 2025	April 2026	April 2026 *
8.0.43	July 2025	April 2026 *	April 2026 *

> **Note**
>
> * MySQL 8.0 will be End-Of-Life in April 2026.

Table 3-4 MySQL HeatWave Service 8.1 to 8.4 Release Schedule

Version	Available Date	Deprecation Date	Unavailable Date
8.1.0 Innovation	July 27, 2023	May 2025	August 2025
8.2.0 Innovation	October 26, 2023	May 2025	August 2025
8.3.0 Innovation	January 24, 2024	May 2025	August 2025
8.4.0 LTS	May 22, 2024	May 2025	August 2025
8.4.3 LTS	Nov 6, 2024	October 2025	January 2026
8.4.4 LTS	May 1, 2025	January 2026	April 2026
8.4.5 LTS	May 1, 2025	April 2026	July 2026
8.4.6 LTS	July 2025	July 2026	October 2026

Table 3-5 MySQL HeatWave Service 9.0 Release Schedule

Version	Available Date	Deprecation Date	Unavailable Date
9.0.0 Innovation	July 10, 2024	May 15, 2025	August 2025
9.0.1 Innovation	Aug 28, 2024	May 15, 2025	August 2025
9.1.0 Innovation	October 23, 2024	May 15, 2025	August 2025
9.1.1 Innovation	November 20, 2024	May 15, 2025	August 2025
9.1.2 Innovation	December 18, 2024	May 15, 2025	August 2025
9.2.0 Innovation	January 22, 2025	May 15, 2025	August 2025
9.2.1 Innovation	April 2, 2025	July 2025	October 2025
9.2.2 Innovation	April 2, 2025	July 2025	October 2025
9.3.0 Innovation	May 29, 2025	Aug 2025	November 2025
9.3.1 Innovation	June 4, 2025	Sep 2025	November 2025
9.3.2 Innovation	June 25, 2025	Sep 2025	Deember 2025
9.4.0 Innovation	July 2025	October 2025	January 2026

3.1.2.2 Server Upgrades

MySQL HeatWave on AWS supports the most recent release versions of MySQL, and the following kinds of upgrades are supported:

- Patch version upgrades, such as from 9.x.*N* to 9.x.*N+1*: These upgrades must be performed manually (except when there is an automatic upgrade due to version removal). See Upgrade MySQL Version for information.

Copyright © 2022, 2025, Oracle and/or its affiliates.

- HeatWave-specific version upgrades, such as from 9.x.y. to 9.x.y-u*N*: These upgrades are performed automatically and according to the maintenance window defined on the DB System.
- Major version upgrades, such as from 8.0.x to 8.*N*.y, or from 8.x.y to 9.*N*.*M*: These upgrades must be performed manually. Only the following upgrade paths are supported:
  - Upgrade from a BugFix or LTS release to an Innovation release *before* the next LTS series.

    > **Note**
    >
    > You cannot upgrade directly from a BugFix or LTS release to an Innovation release *after* the next LTS. For example, to upgrade from MySQL 8.0 to 9.x, you must first upgrade from MySQL 8.0 to MySQL 8.4, then upgrade from MySQL 8.4 to MySQL 9.x.

  - Upgrade from a BugFix or LTS release to the next LTS series.
  - Upgrade from an Innovation release to the next LTS series.

    > **Note**
    >
    > If your major version upgrade does not fall under one of the paths above, it will fail.

  Each major version of MySQL servers is likely to introduce some new features and remove some deprecated features. You should first identify the changes between the current version and the new version, and then verify that these changes do not affect how you use the MySQL database server. Some major changes that you should be aware of before upgrading are listed in the sections to follow (see links below).

To perform a server upgrade, follow the instructions given in Upgrade MySQL Version.

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

- Upgrading MySQL 8.0 BugFix to MySQL 8.4 LTS
- Upgrading MySQL 8.4 LTS to MySQL 9.x Innovation

3.1.2.2.1 Upgrading MySQL 8.0 BugFix to MySQL 8.4 LTS

MySQL 8.4 LTS introduces some new features and removes some deprecated features as compared to MySQL 8.0 BugFix. Here are some important changes you should be aware of before upgrading your DB system from MySQL 8.0 BugFix to MySQL 8.4 LTS:

- As the `default_authentication_plugin` variable is removed in MySQL 8.4, the DB system ignores the `default_authentication_plugin` variable in the MySQL Configuration. It always defaults to `caching_sha2_password`. You can override the default authentication method of your user accounts with the `CREATE USER` and `ALTER USER` statements if needed.
- The deprecated `mysql_native_password` authentication plugin is disabled by default. Your user accounts cannot use the `mysql_native_password` authentication method.

Copyright © 2022, 2025, Oracle and/or its affiliates.

> **Note**
>
> For backward compatibility, if the `mysql_native_password` authentication method is used by any existing users, the upgrade will enable the deprecated `mysql_native_password` authentication plugin. As the `mysql_native_password` authentication has been removed in the major versions after MySQL 8.4, you should start to migrate these user accounts to use the `caching_sha2_password` authentication method.

- These new reserved words are added in MySQL 8.4: `MANUAL`, `PARALLEL`, `QUALIFY`, and `TABLESAMPLE`. You cannot use any reserved words in unquoted identifiers.
- The `AUTO_INCREMENT` option is no longer allowed in `FLOAT` and `DOUBLE` data types. Upgrade fails if you have a table that contains a `FLOAT` or `DOUBLE` column with `AUTO_INCREMENT`.
- The `SET_ANY_DEFINER` privilege introduced in MySQL 8.2.0 is granted to the administrator account and `administrator` role of any DB system running MySQL version 8.2.0 or higher.

> **Note**
>
> You cannot upgrade MySQL 8.0 to MySQL 9.x directly; you must first upgrade MySQL 8.0 to MySQL 8.4, then upgrade MySQL 8.4 to MySQL 9.x.

Related Topics

- What Is New in MySQL 8.4 since MySQL 8.0

3.1.2.2.2 Upgrading MySQL 8.4 LTS to MySQL 9.x Innovation

MySQL 9.x Innovation introduces some new features and removes some deprecated features as compared to MySQL 8.4 LTS. Here are some important changes you should be aware of before upgrading your DB system from MySQL 8.4 LTS to MySQL 9.x Innovation:

- The `mysql_native_password` authentication plugin is removed in MySQL 9.0. Your user accounts cannot use the `mysql_native_password` authentication method.

> **Note**
>
> Before upgrading, you must migrate all user accounts that use the `mysql_native_password` authentication method to the `caching_sha2_password` authentication method.

> **Note**
>
> Inbound replication channels cannot authenticate to a source server as a user that is created with the `mysql_native_password` authentication method after the target server is upgraded to 9.x. You must recreate the replication user on the source using the `caching_sha2_password` authentication method. Also, any operation that involves the `mysql_native_password` authentication method on the source server (for example, creating or changing the password of a user account that uses the `mysql_native_password` authentication method) will break the replication.

Copyright © 2022, 2025, Oracle and/or its affiliates.

- As of MySQL 9.2, the `BINLOG` keyword is restricted and can no longer be used unquoted as a label in a MySQL stored routine or stored function. You should update any affected applications accordingly, before upgrading to MySQL 9.2.

Related Topics

- What Is New in MySQL 9.0
- What Is New in MySQL 9.1
- What Is New in MySQL 9.2
- What Is New in MySQL 9.3

3.1.2.3 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/.
....
```


Copyright © 2022, 2025, Oracle and/or its affiliates.

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`, MySQL 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` (MySQL 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);
```

3.1.2.4 Unsupported MySQL Server Features

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

Copyright © 2022, 2025, Oracle and/or its affiliates.

- 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

3.1.2.5 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.

3.1.2.6 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

Copyright © 2022, 2025, Oracle and/or its affiliates.

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 Enterprise Audit

  Enables the MySQL Server to produce a log file containing an audit record of server activities. The log contents can include information such as when clients connected and disconnected, what actions they performed while connected, and which databases and tables they accessed. See MySQL Enterprise Audit for details.

3.1.2.7 MySQL Enterprise Audit

MySQL HeatWave on AWS supports the MySQL Enterprise Audit plugin for MySQL Server 8.1.0 and later. The audit plugin enables the MySQL Server to produce a log file containing an audit record of server activities. The log contents can include information such as when clients connected and disconnected, what actions they performed while connected, and which databases and tables they accessed.

You can add statistics for the time and size of the queries to detect outliers. By default, the audit plugin is disabled. You have to enable the plugin and define audit plugin filters to log auditable events for all or specific users.

To use the database audit plugin, you need to:

- Connect to the DB System
- Use the **Query Editor** on the **Workspace** page or your command-line client to perform the following operations.

Enable Audit Plugin

The audit plugin is disabled in the default MySQL configurations on MySQL HeatWave on AWS. Enable it by setting the MySQL server system variable `audit_log_disable` to `OFF` in a custom configuration for your MySQL server and then use it to create your new DB system or update an existing one (see Creating a MySQL Configuration for details on setting preferred values for User-Configurable System Variables).

Copyright © 2022, 2025, Oracle and/or its affiliates.

Granting Audit Administration Privileges

By default, the administrator user you defined while creating the DB system has the audit administration privileges. You can grant the privileges to more users (for example, to the user `User001`) by running the following command:

```
GRANT AUDIT_ADMIN ON *.* TO <User001>;
```

Defining Audit Plugin Filters

Define your audit plugin filters to log auditable events for all or specific users. See Writing Audit Log Filter Definitions for details. Here are some sample filters and the commands to define them:

- To audit all events, run the following command:

  ```
  SELECT audit_log_filter_set_filter('log_all', '{ "filter": { "log":
  true } }');
  ```

- To audit connection events only, run the following command:

  ```
  SELECT audit_log_filter_set_filter('log_conn_events', '{"filter":
  {"class": { "name": "connection" }}}');
  ```

- To view audit filters, run the following command:

  ```
  SELECT * FROM mysql_audit.audit_log_filter;
  ```

Assign the filters you created above to all or specific users for them to access the audit data. For example:

- To assign the default audit filter to log all events from any account, use the wildcard character `%`:

  ```
  SELECT audit_log_filter_set_user('%', 'log_all');
  ```

- To assign the default audit filter to log all connect events from any account, use the wildcard character `%`:

  ```
  SELECT audit_log_filter_set_user('%', 'log_conn_events');
  ```

- To assign the default audit filter to log all events from a specific user such as `user_dba`, run the following command:

  ```
  SELECT audit_log_filter_set_user('user_dba@%', 'log_all');
  ```

- To view the assigned rules, run the following command:

  ```
  SELECT * FROM mysql_audit.audit_log_user;
  ```

- To unassign the rules from the user `user_dba`, run the following command:

  ```
  SELECT audit_log_filter_remove_user('user_dba@%');
  ```


Copyright © 2022, 2025, Oracle and/or its affiliates.

Accessing and Analyzing Audit Data

Use the audit data to monitor the DB system. Use these statements to access the audit data through the functions `audit_log_read()` and `audit_log_read_bookmark()` :

- To view any new logs since you last checked, run the following command (this ensures you are always updated with the latest audit log entries):

  ```
  SELECT audit_log_read(audit_log_read_bookmark());
  ```

- To extract audit log contents starting from a particular timestamp, provide additional parameters within the `audit_log_read()` function:

  ```
  SELECT audit_log_read('{ "start": { "timestamp": "2024-01-24 12:30:00" },
  "max_array_length": 500 }');
  ```

- To view the audit data in an easier to read format, use the `JSON_PRETTY()` and `CONVERT()` functions; for example:

  ```
  SELECT JSON_PRETTY(CONVERT(audit_log_read( '{ "start": { "timestamp":
  "2024-01-24 12:30:00" }, "max_array_length": 500 }') USING UTF8MB4));
  ```

- To transform data into a tabular format, use the MySQL JSON functions. For example, you can transform a subset of the JSON name-value pairs into a structured table format, making it easier to interact with and analyze data:

  ```
  SELECT @@server_uuid as server_uuid, ts, class, event,
  login_ip,login_user,connection_id,
   status,connection_type,_client_name,_client_version,
   command,sql_command,command_status
  FROM
  JSON_TABLE
  (
    AUDIT_LOG_READ( '{ "start": {\"timestamp\": \"2024-01-16 15:33:37\"},
  \"max_array_length\": 10 }' ),
    '$[*]'
    COLUMNS
    (
      ts TIMESTAMP PATH '$.timestamp',
      class VARCHAR(20) PATH '$.class',
      event VARCHAR(80) PATH '$.event',
      login_ip VARCHAR(200) PATH '$.login.ip',
      login_user VARCHAR(200) PATH '$.login.user',
      connection_id VARCHAR(80) PATH '$.connection_id',
      status INT PATH '$.connection_data.status',
      connection_type VARCHAR(40) PATH '$.connection_data.connection_type',
      _client_name VARCHAR(80) PATH
  '$.connection_data.connection_attributes._client_name',
      _client_version VARCHAR(80) PATH
  '$.connection_data.connection_attributes._client_version',
      command VARCHAR(40) PATH '$.general_data.command',
      sql_command VARCHAR(40) PATH '$.general_data.sql_command',
      command_status VARCHAR(40) PATH '$.general_data.status'
     )) as audit_log;
  ```


Copyright © 2022, 2025, Oracle and/or its affiliates.

- To further refine the data extraction, use `WHERE` clauses (for example, `WHERE connection_type <> 'SSL'`) in the SQL statements.

Related Topics

- MySQL Enterprise Audit

3.1.2.8 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.

3.1.2.9 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`

Copyright © 2022, 2025, Oracle and/or its affiliates.

  - 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

Copyright © 2022, 2025, Oracle and/or its affiliates.

  - CREATE ROUTINE
  - CREATE TEMPORARY TABLES
  - CREATE VIEW
  - DROP
  - EVENT
  - INDEX
  - LOCK TABLES
  - REFERENCES
  - TRIGGER

3.1.2.10 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`

3.2 Launching a Starter DB System

A Starter DB System is a preconfigured MySQL HeatWave DB System with preloaded sample data that you can easily launch for testing out MySQL HeatWave on AWS. The following are included with your Starter DB System launch:

- A DB System running the latest MySQL version with 2 vCPUs, 16GiB memory, and 50 GiB storage.
- Preloaded `airportDB` and `TPCH` sample databases.
- An attached one-node MySQL HeatWave Cluster with 256 GiB memory.

Follow these steps to launch the Starter DB System:

- In the MySQL HeatWave Console, select the **Home** tab.
- Click the **Launch Starter DB System** button in the **Getting Started** region. The **Launch Starter DB System** dialog appears.
- Specify the **Administrator credentials**.
  - **Username**: Name of the user that will be granted a specific set of MySQL Server administrator privileges. See Default MySQL Privileges. The administrator username can have up to 32 characters. Certain names are reserved. See Reserved User Names.

Copyright © 2022, 2025, Oracle and/or its affiliates.

  - **Password**: The administrator password, which must be between 8 and 32 characters, and include at least one number, one uppercase letter, one lowercase letter, and one character from `,.-+*;:_!#%&/()=?><`.
  - **Confirm Password**: Reenter the administrator password.
- Click the **Launch** button. A DB system named **Starter DB System** (you can change the name later) is being launched.

You are redirected to the **Resources** tab, on which the **DB Systems** tab allows you to monitor the state of the launch operation, which may take some time to complete. The state will change from **Creating** to **Active** when the operation has completed successfully.

Click the row of the starter DB system to open the **MySQL DB System Details** pane below the list of DB Systems or click on the **Starter DB System** to open the **MySQL DB System Details** page, to review the **DB System Information**.

3.3 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 the following information:

- 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 131072 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*.

Copyright © 2022, 2025, Oracle and/or its affiliates.

> **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**
  >
  > The choices of configuration and the features of your DB system are interrelated; for example:
  >
  > – If you have chosen to create a high availability system, you will only be able to choose a configuration that supports high availability.
  >
  > – If you want to use a MySQL HeatWave Cluster, make sure you select a MySQL configuration that supports the feature (the default configuration for the hardware configuration you have chosen will support MySQL HeatWave Clusters).

- 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 Editing a 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 MySQL HeatWave on AWS.

  – The public IP addresses of application servers where client applications that will use MySQL HeatWave on AWS 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`.

Copyright © 2022, 2025, Oracle and/or its affiliates.

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

Table 3-6 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 (except for high availability DB Systems). 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.
- MySQL HeatWave Cluster configuration.
  The MySQL HeatWave Cluster shape determines the resources allocated to each MySQL HeatWave node. See Supported Shapes.

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

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

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

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

Copyright © 2022, 2025, Oracle and/or its affiliates.

> **Note**
>
> For MySQL HeatWave AutoML functionality, the `MySQL HeatWave.256GB` node shape is recommended when creating a MySQL HeatWave Cluster. The `MySQL 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 **Resources** tab.
2. On the **DB Systems** tab, click **Create MySQL DB System** to open the **Create MySQL DB System and MySQL 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. **Standalone** or **High availability**
   - **Standalone**: Select for a single-instance DB System.
   - **High availability**: Select for a three-instance High Availability DB System.
6. **Hardware configuration**: Select a suitable hardware configuration.
   - **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 131072.
7. **Preferred availability zone**: This determines the physical location of the stand-alone DB System, or the preferred location of the primary instance for a High Availability DB system:
   - **Automatic**: AWS selects the physical AWS Availability Zone.
   - **Manual**: Select a physical AWS Availability Zone.
8. **MySQL Configuration**: Select a MySQL configuration.
   - Click **Change** to change the default configuration.
   - Select a configuration from the list.
   - Click **Apply MySQL Configuration**.

> **Note**
>
> Choices of configuration might be restricted by features you have chosen for the DB system; for example, if this is a high availability system, you will only be able to change to another configuration that supports high availability (if available).

Copyright © 2022, 2025, Oracle and/or its affiliates.

9. **Database Version**: Select the MySQL Server version to deploy. The latest MySQL Server version is selected by default.
10. **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**.
11. **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.
12. **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 regularly.

      > **Note**
      >
      > For a high availability system, this is selected automatically and cannot be deselected.

    - Set a **Backup 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**.
13. **IAM roles**:
    - **Data Import role ARN**: If you want to import data with Console, specify the role ARN. See Creating an IAM Role to Access an Amazon S3 Bucket.
    - **MySQL HeatWave AutoML role ARN**: If you want to use the external AWS Bedrock LLMs for MySQL HeatWave GenAI, specify the role ARN. See Creating an IAM Role to Access Amazon Bedrock LLMs.
    - **Lakehouse role ARN**: If you want to use Lakehouse, specify the role ARN. See Creating an IAM Role to Access an Amazon S3 Bucket. You can also select the **Sample Lakehouse role**.
14. **Monitoring configuration**:
    - **Enable CloudWatch Monitoring**: Select if you want to use CloudWatch Monitoring to monitor your DB System and MySQL HeatWave Cluster, and then provide the following information:
      - Provide the **CloudWatch Monitoring role ARN** (see Creating an IAM Role and Policy for CloudWatch Monitoring).
      - Click the **Select Monitoring Configuration** button if you want to change the monitoring configuration from the default **Basic** setting. In the **Select Monitoring Configuration** pane that appears, select under **Template** one of the following

Copyright © 2022, 2025, Oracle and/or its affiliates.

template types to see the **Metrics** included for each of them (see also CloudWatch Monitoring Metrics for a list of all the metrics and some details on them):

* **Basic**: Configuration for basic monitoring.
* **Detailed**: Configuration for detailed monitoring.
* **Custom**: Customized monitoring configuration. If you have selected the **Basic** or **Detailed** configuration and add or remove any metrics, it will automatically change to a **Custom** configuration.

After making any desired changes to the monitoring configuration, click **Change** to accept it.

15. Click **Next** to proceed to the next page.
16. **Provision MySQL HeatWave Cluster**: Select whether to provision a MySQL HeatWave Cluster.
    This is only available for a standalone DB System and if the chosen MySQL configuration supports MySQL HeatWave.
    - **Basic information**: Provide a name and description for the MySQL HeatWave Cluster:
      - **Display Name**: Specify a display name for the MySQL HeatWave Cluster or use the generated default name.
      - **Description**: Specify a user-friendly description of the MySQL HeatWave Cluster.
    - **MySQL HeatWave Cluster configuration**: Specify the shape and cluster size.
      - **Shape**: Select the shape to use for the MySQL HeatWave Cluster.
      - **Cluster Size**: Specify the size of the cluster between 1 and 128 MySQL HeatWave nodes.
17. 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** pane and review the **DB System Information**.

3.4 Managing a DB System

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

- Stopping, Starting, or Restarting a DB System
- Enable or Disable High Availability
- Editing a DB System
- Update Networking
- Update MySQL Configuration
- Update MySQL Shape
- Increasing DB System Storage
- Update Monitoring Configuration
- Upgrade MySQL Version
- Deleting a DB System

Copyright © 2022, 2025, Oracle and/or its affiliates.

3.4.1 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. After any planned or unplanned restart, data is recovered into the associated MySQL HeatWave Cluster from the AWS S3 storage.

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 MySQL HeatWave Cluster. When a DB System restarts, the MySQL HeatWave Cluster also restarts, and it reloads automatically the previously-loaded data from the AWS S3 storage. See: Data Load Progress and Status Monitoring.

To start, stop, or restart a DB System:

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

3.4.2 Enable or Disable High Availability

See Enabling and Disabling High Availability for instructions.

3.4.3 Editing a DB System

Use the MySQL HeatWave Console to edit details of a DB System and various options for it.

This task requires the following:

- (For data import feature only) Permissions to assume the data import role.

Copyright © 2022, 2025, Oracle and/or its affiliates.

- (For MySQL HeatWave AutoML features only) Permissions to assume the MySQL HeatWave AutoML role.
- (For Lakehouse features only) Permissions to assume the Lakehouse role.

Do the following to edit a DB System:

1. In the MySQL HeatWave Console, select the **Resources** 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 of 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. 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. See Maintenance.
   - 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. 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.
   - 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. **IAM roles**:
   - **Data Import role ARN**: If you want to import data with Console, specify the data import role ARN. See Creating an IAM Role to Access an Amazon S3 Bucket.
   - **MySQL HeatWave AutoML role ARN**: If you want to use the external AWS Bedrock LLMs for MySQL HeatWave GenAI, specify the role ARN. See Creating an IAM Role to Access Amazon Bedrock LLMs.
   - **Lakehouse role ARN**: If you want to use Lakehouse, or want to export query results from MySQL HeatWave to your S3 bucket, specify the Lakehouse role ARN. See Creating an IAM Role to Access an Amazon S3 Bucket.
   - **CloudWatch Monitoring role ARN**: If you want to use CloudWatch Monitoring, provide the ARN (see Creating an IAM Role and Policy for CloudWatch Monitoring).
7. Click **Save**.

Copyright © 2022, 2025, Oracle and/or its affiliates.

3.4.4 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 **Resources** 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. Depending on whether you want the DB System to be accessible via the public internet, do one of the following:
   - Select **Enable inbound connectivity from allowed public IP address ranges**, and edit the **Allowed Client Addresses**.
   - Deselect **Enable inbound connectivity from allowed public IP address ranges**.
4. Click **Update** to save the changes.

3.4.5 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**
>
> You cannot change the System Initialization Variables by updating the DB System's configuration. If you try to update your DB System with a MySQL configuration containing different system initialization variable values from those your DB System's, the update will fail.

During the update, the DB System will be in the **UPDATING** state until the update is finished. If the new configuration changes any system variables that are not dynamic, the MySQL Server process, `mysqld`, will restart. If the DB System has an attached MySQL HeatWave Cluster, it will restart and reload data from the DB System.

If an update fails, the DB System will be rolled back to its original configuration. But if the rollback fails, the DB System will end up in the **UPDATING_ERROR** state.

To update the MySQL configuration for a DB System, make sure it is running, and follow these steps:

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

Copyright © 2022, 2025, Oracle and/or its affiliates.

2. On the **DB Systems** tab, in the list of DB Systems, find the DB System to update, 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, and from the **Actions** menu choose **Update MySQL Configuration**.
3. Click **Change**. The **Select MySQL Configuration for shape …** dialog box appears.
4. Select the desired MySQL configuration from a list. Select a filter for the list:
   - **Any**: Show any configuration that can be applied to your DB System
   - **Default**: Show the default configuration for the MySQL shape of your DB System.
   - **Custom**: Show all custom configurations you have created by Creating a MySQL Configuration or Copying a MySQL Configuration that are applicable to your DB System.

   Refine the list by typing into the search bar part of the names of the configurations you are interested in.

   > **Note**
   >
   > Only configurations that support the DB System's shape are shown. Also, if a DB System has a MySQL HeatWave Cluster attached, only configurations that support MySQL HeatWave are shown; to change the configuration to one that does not support MySQL HeatWave, remove the DB System's MySQL HeatWave Cluster.

   Click **Apply MySQL Configuration** to confirm your choice.
5. Click **Update** to actually apply the configuration to your DB System.

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

Related Topics

- System Initialization Variables
- User-Configurable System Variables
- User-Configurable Shape-Dependent System Variables
- Creating a MySQL Configuration
- Copying a MySQL Configuration
- View MySQL Configuration Details

3.4.6 Update MySQL Shape

Users can reduce the shape of a DB System to save cost or expand the shape to enhance the function or performance. Decisions on shape changes are aided by the Autopilot Shape Advisor, which uses the system workload to access the suitability of a shape and make recommendations on shape changes.

Changing the shape will bring the DB System offline temporarily.

Copyright © 2022, 2025, Oracle and/or its affiliates.

To update the MySQL shape or configuration for a DB System, make sure the DB System is in either the **ACTIVE** or **INACTIVE** state, and follow these steps:

1. In the MySQL HeatWave Console, select the **Resources** tab.
2. On the **DB Systems** tab, in the list of DB Systems, find the DB System to update, and do one of the following:
   - Click on the row of the DB System to highlight it, and from the **Actions** menu choose **Update MySQL Shape** .
   - Click the name of the DB System to open the **DB System Details** page, and from the **Actions** menu choose **Update MySQL Shape**.
3. On the **Update MySQL Shape** panel, provide the following information:
   - **Hardware configuration**
     - **Shape**: Select the shape you want to update your DB System to. See Supported Shapes.

       > **Note**
       >
       > If you do not choose a shape other than you current shape, you will not be able to proceed to the following steps.

   - **MySQL Configuration**: If you were previously using a default configuration, the corresponding default configuration for the new shape is displayed as the **Selected MySQL Configuration**. If you were previously using a custom configuration, you will need to select a new custom configuration or default configuration compatible with the new shape. To select or to change a configuration:
     - Click the **Change** button. The **Select MySQL Configuration panel for shape …** panel appears.
     - Select the desired MySQL configuration from a list. Select a filter:
       * **Any**: Show any configuration that can be applied to your DB System for the selected shape
       * **Default**: Show the default configuration for the selected shape of your DB System.
       * **Custom**: Show all custom configurations you have created by Creating a MySQL Configuration or Copying a MySQL Configuration that are applicable to your DB System for the selected shape.

       Refine the list by typing into the search bar part of the names of the configurations you are interested in.

       > **Note**
       >
       > Only configurations that support the selected shape are shown. Also, if your DB System has a MySQL HeatWave Cluster attached, only configurations that support MySQL HeatWave are shown; to change the configuration to one that does not support MySQL HeatWave, remove the DB System's MySQL HeatWave Cluster.

       Click **Use MySQL Configuration** to confirm your choice.

Copyright © 2022, 2025, Oracle and/or its affiliates.

4. Click **Update** to actually apply the selected shape and configuration to your DB System.

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

Related Topics

- System Initialization Variables
- User-Configurable System Variables
- User-Configurable Shape-Dependent System Variables
- Creating a MySQL Configuration
- Copying a MySQL Configuration
- View MySQL Configuration Details

3.4.7 Increasing DB System Storage

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

Increasing Storage by Updating the Data Storage Size of a DB System

Use the Console to increase the data storage size of a DB System that is in the `ACTIVE`, `STARTING_ERROR`, or `RESTARTING_ERROR` state. If the DB System is operating normally, resizing the storage will not restart the DB System, and you can continue to query it while the storage is being increased. Otherwise, resizing the storage may restart the DB System in order to return it to a normal operating state.

It is recommended that you create a backup of the DB System before updating the data storage size. See Creating a Backup.

> ⚠ **Caution**
>
> You can only increase the data storage size once every 6 hours. It is recommended that you increase your storage by an amount large enough to support your storage requirement in the next 6 hours.

Follow these steps to update the data sotrage size of a DB system:

1. In the MySQL HeatWave Console, select the **Resources** 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 131072.
4. Click **Update**.

Increasing Storage by Backup and Restore

1. Create a backup of the DB System. See Creating a Backup.

Copyright © 2022, 2025, Oracle and/or its affiliates.

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 Using 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.

3.4.8 Update Monitoring Configuration

Update the CloudWatch Monitoring configuration for a DB System.

During the update, the DB System will be in the **UPDATING** state until the update is finished.

To update the CloudWatch Monitoring configuration for a DB System follow these steps:

1. In the MySQL HeatWave Console, select the **Resources** tab.
2. On the **DB Systems** tab, in the list of DB Systems, find the DB System to update, and do one of the following:
   - Click on the row of the DB System to highlight it, and choose **Update Monitoring Configuration** from the **Actions** menu.
   - Click the name of the DB System to open the **DB System Details** page, and from the **Actions** menu choose **Update Monitoring Configuration**.
3. In the **Update Monitoring Configuration** pane:
   - Select **Enable CloudWatch Monitoring** if you want to use CloudWatch Monitoring to monitor your DB System and MySQL HeatWave Cluster, or deselect it if you want to disable the function.
   - Provide or update the **CloudWatch Monitoring role ARN** to use CloudWatch Monitoring (see Creating an IAM Role and Policy for CloudWatch Monitoring).
   - Click the **Select Monitoring Configuration** button if you want to change the metrics provided by CloudWatch Monitoring.
   - In the **Select Monitoring Configuration** pane, select under **Template** one of the following template types, and see the **Metrics** included for each of them (see also CloudWatch Monitoring Metrics for a list of all the metrics and some details on them):
     - **Basic**: Configuration for basic monitoring. This is the default configuration.
     - **Detailed**: Configuration for detailed monitoring.
     - **Custom**: Customized monitoring configuration. Customize the configuration by selecting the desired metrics.

     Click **Change** to confirm your changes, or click **Cancel** to stay with the original setting.
4. Click **Update** to apply the configuration to your DB System.

For metrics updates, there might be a brief interval during which no metrics are emitted, as the system adjusts to the new configuration.

An update to a CloudWatch role ARN and changes to the permission policy of the CloudWatch role are effective immediately.

Copyright © 2022, 2025, Oracle and/or its affiliates.

3.4.9 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.

- **Update**

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

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

  This option is only available if a more recent version is available. It shows the version number.

  More than one version might be available.

To upgrade the MySQL version:

1. In the MySQL HeatWave Console, select the **Resources** 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 following options (see detailed descriptions of the options above):
   - **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.

3.4.10 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 **Resources** 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:

Copyright © 2022, 2025, Oracle and/or its affiliates.

- 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.

3.5 Viewing DB System Details

Do the following to view DB System details:

1. In the MySQL HeatWave Console, select the **Resources** tab.
2. On the **DB Systems** tab, in the list of DB System, find the DB System you want to view details for, and do one of the following:
   - Click 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

3.5.1 MySQL DB System Details

The **MySQL DB System Details** page has the following tabs.

Table 3-7 Tabs on the MySQL DB System Details Page

Name	Description
Details	DB System details summary. See Table 3-8.
Backups	DB System backup details. See Table 3-9.
PrivateLinks	List of PrivateLinks whose targets are this DB System. Click on each PrivateLink to see Table 3-10.
Shape Advisor	Shape Advisor details. See Auto Shape Prediction Data.
Metrics	Metrics for the DB System. See Table 3-11.
Monitoring Configuration	CloudWatch Monitoring Configuration. See Table 3-12.
Events	DB System event details. See Table 3-13.

Copyright © 2022, 2025, Oracle and/or its affiliates.

Table 3-8 MySQL DB System Details Summary

Field	Description
General Information	• **Name**: The name of the DB System. • **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. • **Resource ID**: The unique resource identifier assigned to the DB System when it is created. • **Description**: The user-defined description of the DB System. • **Last Updated**: The date and time the DB System was last updated. • **Created**: The date and time the DB System was created.

Copyright © 2022, 2025, Oracle and/or its affiliates.

Table 3-8 (Cont.) MySQL DB System Details Summary

Field	Description
DB System Configuration	• **High Availability**: Whether High Availability is enabled for the DB System. • **Shape**: The resource template for the MySQL instance. For information about shapes, see Supported Shapes. • **Storage Size**: The amount of storage available to the DB System. • **vCPU count**: The number of vCPUs of the DB System. • **Memory**: The size of the RAM of the DB System. • **MySQL version**: The MySQL Server version used by the DB System. • **MySQL Configuration**: The MySQL Configuration of 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.
Networking	• **Accessibility**: Whether the DB System is accessible via the public internet (see **Allowed public IP address ranges**), PrivateLink, or console only. • **Require SSL**: Whether connections to the DB System require encryption. • **Hostname**: The host name of the DB System. The host name is a fully qualified domain name (FQDN). • **PrivateLink Hostname** Use this hostname to connect to the DB System over PrivateLink. • **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 X Protocol. • **Allowed public IP address ranges**: 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.

Copyright © 2022, 2025, Oracle and/or its affiliates.

Table 3-8 (Cont.) MySQL DB System Details Summary

Field	Description
Placement	The physical **Availability Zone(s)** where the DB System resides. • For a standalone DB System: **Availability Zone** of the DB System. • For a High Availability DB System: – **Preferred Availability Zone** for the primary instance. – **Current Availability Zone** of the primary instance. – **Secondary Availability Zones** of the secondary instances.
IAM roles	• **Data Import role ARN**. See Creating an IAM Role to Access an Amazon S3 Bucket. • **MySQL HeatWave AutoML role ARN**. See Creating an IAM Role to Access Amazon Bedrock LLMs. • **Lakehouse role ARN**. See Creating an IAM Role to Access an Amazon S3 Bucket. • **CloudWatch role ARN**. See Creating an IAM Role and Policy for CloudWatch Monitoring.

Copyright © 2022, 2025, Oracle and/or its affiliates.

Table 3-8 (Cont.) MySQL DB System Details Summary

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

Copyright © 2022, 2025, Oracle and/or its affiliates.

Table 3-8 (Cont.) MySQL DB System Details Summary

Field	Description
Backup policy	• **Automatic backups**: Is automatic backups **Enabled** or **Disabled**. • **Retention in days**: Backup retention period in days. • **Window start hour**: The time to start a backup.
Channel	• **Channel name**: Name of the inbound replication channel. • **State**: State of the inbound replication channel. See Table 8-4 for the list of possible states. • **Enabled**: Whether the inbound replication channel is enabled.

Table 3-9 MySQL DB System Backup Details

Field	Description
Name	The name of the backup
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.
Snapshot DB System ID	The ID of the DB System used to create the backup.
Snapshot DB System	The name of the DB System used to create the backup.
Type	Type of backup: **Scheduled** or **Manual**
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.

See Backups for details on MySQL DB System backups.

Copyright © 2022, 2025, Oracle and/or its affiliates.

Table 3-10 MySQL DB System PrivateLink Details

Field	Description
Name	The name of the PrivateLink. Click on it to see PrivateLink Details.
State	The state of the PrivateLink. See Table 6-4.
Type	The type of the PrivateLink, which is either **Query** or **Egress**.
Created	The date and time the backup was created.

Table 3-11 MySQL DB System Metrics

Field	Description
CloudWatch metrics	Whether CloudWatch Monitoring is enabled or disabled. Also: • If CloudWatch Monitoring is enabled, a **View CloudWatch metrics** link to the AWS CloudWatch console is displayed (you must be logged in to the AWS account to view the metrics). • If CloudWatch Monitoring is disabled, an **Update Monitoring Configuration** button is displayed for enabling CloudWatch Monitoring.
Service metrics	Service metrics for the MySQL DB System: • **CPU Utilization**: he percentage of CPU utilization on the MySQL DB System. • **Memory Utilization**: The percentage of memory utilization on the MySQL DB System. • **Disk Utilization**: The message body of the event report. • **Disk Usage** : The date and time the event report was created. • **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. • **Buffer Pool**: The buffer pool size on the MySQL DB System.

Table 3-12 MySQL DB System Monitoring Configuration

Field	Description
Metrics	List of metrics selected by the CloudWatch Monitoring configuration. The **Name** and **Description** for each metric is shown. A search box provides filtering for the list of metrics by name.

Copyright © 2022, 2025, Oracle and/or its affiliates.

Table 3-13 MySQL DB System Event Details

Field	Description
Type	The event and its status.
Severity	The severity level of the event, which can be one of • `INFO`: General information is provided. No special concerns • `WARNING`: Warning on an event that might impact the DB System's operation if no actions are taken. • `CRITICAL`: A critical event. An urgent action is required to avoid adverse impact on your DB System.
Message	The message body of the event report.
Created	The date and time the event report was created.

See Events for details on event reports.

Copyright © 2022, 2025, Oracle and/or its affiliates.

4

MySQL HeatWave Clusters

This chapter describes how to create and manage MySQL HeatWave Clusters.

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

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

MySQL HeatWave Clusters support Lakehouse. Lakehouse enables query processing on the data residing in Amazon S3. The source data is read from Amazon S3, transformed to theMySQL HeatWave format, stored in the MySQL HeatWave persistence storage layer, and then loaded to the MySQL HeatWave Cluster memory. MySQL HeatWave Lakehouse supports structured and relational data in CSV and Parquet formats. It is supported on the `MySQL HeatWave.256GB` node shape only. Lakehouse supports a maximum of 512 MySQL HeatWave nodes.

- Creating a MySQL HeatWave Cluster
- Managing a MySQL HeatWave Cluster
- Viewing MySQL HeatWave Cluster Details
- MySQL HeatWave Cluster Data Recovery

4.1 Creating a MySQL HeatWave Cluster

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

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

> **Note**
>
> Changing the number of nodes in an existing MySQL 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.

Copyright © 2022, 2025, Oracle and/or its affiliates.

To create a MySQL HeatWave Cluster, do the following:

1. In the MySQL HeatWave Console, select the **Resources** tab, and then the **MySQL HeatWave Clusters** tab.
2. Click **Create MySQL HeatWave Cluster**. The **Create MySQL HeatWave Cluster** dialog is displayed.
3. On the **Create MySQL HeatWave Cluster** dialog, provide the following information:
   - **Basic Information**
     - **Display Name**: Specify a user-friendly display name for the MySQL HeatWave Cluster.
     - **Description**: Specify a user-friendly description of the MySQL HeatWave Cluster.
   - **DB System Name**: Select a DB System from the drop down menu.
   - Select **Enable MySQL HeatWave Lakehouse** to support MySQL HeatWave Lakehouse.
   - **MySQL HeatWave Cluster Configuration**: You can select either of the following options by the toggle buttons:
     - **Manually configure cluster**. Configure the cluster then by filling in the following information.
       * **Shape**: Select a MySQL HeatWave node shape. For information about supported shapes, see Supported Shapes.

         > **Note**
         >
         > If you intend to use the MySQL HeatWave AutoML functionality, the `MySQL HeatWave.256GB` node shape is recommended when creating a MySQL HeatWave Cluster. The `MySQL 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**: Select the number of MySQL HeatWave nodes. The minimum number is 1, and the maximum number depends on the following conditions:
         * For the `Heatwave.16GB` node shape: 16.
         * For the `HeatWave.256GB` node shape:
           * If MySQL HeatWave Lakehouse is not enabled: 128.
           * If MySQL HeatWave Lakehouse is enabled: 512.
     - **Automatically configure cluster**. Configure the cluster by clicking **Estimate memory size** to estimate the required cluster size using MySQL HeatWave Autopilot. See more details in Estimating Cluster Memory Size with MySQL HeatWave Autopilot.
   - Check the MySQL HeatWave Cluster configuration in the **Summary** section. Go back to the steps above to make changes if needed.
4. Click **Create** to create the MySQL HeatWave Cluster.

You are returned to the **MySQL 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.

Copyright © 2022, 2025, Oracle and/or its affiliates.

- Estimating Cluster Memory Size with MySQL HeatWave Autopilot

4.1.1 Estimating Cluster Memory Size with MySQL HeatWave Autopilot

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

A cluster size estimate is generated using MySQL HeatWave Autopilot machine learning techniques. MySQL HeatWave 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 MySQL HeatWave Cluster size, load data into the DB System before you create a MySQL HeatWave Cluster. See Importing Data.

Prerequisites:

- The data you intend to load into the MySQL 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 MySQL 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 memory size**.

   The **Estimate memory 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.
   - **MySQL HeatWave Cluster Memory Usage (GiB)**: The estimated amount of MySQL 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 MySQL 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, MySQL HeatWave Cluster nodes required, and memory provided by the cluster.

4. To apply the cluster size estimate, click **Apply Estimated Memory Size**.

   You are returned to the **Create MySQL HeatWave Cluster** dialog where the estimate is shown and applied to the cluster.

Copyright © 2022, 2025, Oracle and/or its affiliates.

Cluster Size Estimate Table Warnings

This topic describes table warnings that may appear in the **Tables from selected schemas** pane, in the **Estimate memory 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 MySQL HeatWave.

4.2 Managing a MySQL HeatWave Cluster

- Starting, Stopping, or Restarting a MySQL HeatWave Cluster
- Editing a MySQL HeatWave Cluster
  Use the Console to edit the MySQL HeatWave Cluster. You can change the display name, description, the shape of the nodes, or the number of nodes for the MySQL HeatWave Cluster.
- Deleting a MySQL HeatWave Cluster

4.2.1 Starting, Stopping, or Restarting a MySQL HeatWave Cluster

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

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

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

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

Copyright © 2022, 2025, Oracle and/or its affiliates.

To start, stop, or restart a MySQL HeatWave Cluster:

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

4.2.2 Editing a MySQL HeatWave Cluster

Use the Console to edit the MySQL HeatWave Cluster. You can change the display name, description, the shape of the nodes, or the number of nodes for the MySQL HeatWave Cluster.

1. In the MySQL HeatWave Console, select the **Resources** tab, and then the **MySQL HeatWave Clusters** tab.
2. In the list of MySQL HeatWave Clusters, find the MySQL HeatWave Cluster you want to edit, and do one of the following:
   - Click the row of the MySQL HeatWave Cluster to highlight it, and click **Edit MySQL HeatWave Cluster** from the **Actions** menu.
   - Click the name of the MySQL HeatWave Cluster to open the **Details** page and click **Edit MySQL HeatWave Cluster** from the **Actions** menu.
3. On the **Edit MySQL HeatWave Cluster** panel, update the following information as needed:
   - **Basic information**
     - **Display Name**: Change the name of the MySQL HeatWave Cluster as needed.
     - **Description**: Change the description of the MySQL HeatWave Cluster as needed.
   - **MySQL HeatWave Cluster Configuration**: You can select either of the following options by the toggle buttons:
     - **Manually configure cluster**. Configure the cluster then by filling in the following information.
       * **Shape**: Select a MySQL HeatWave node shape. For information about supported shapes, see Supported Shapes.

         > **Note**
         >
         > If you intend to use MySQL HeatWave AutoML functionality, the `MySQL HeatWave.256GB` node shape is recommended. The `MySQL 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**: Change the number of MySQL HeatWave nodes as needed. The minimum number is 1, and the maximum number depends on the following conditions:
         * For the `Heatwave.16GB` node shape: 16.

Copyright © 2022, 2025, Oracle and/or its affiliates.

        * For the `HeatWave.256GB` node shape:
            * If MySQL HeatWave Lakehouse is not enabled: 128.
            * If MySQL HeatWave Lakehouse is enabled: 512.
    - **Automatically configure cluster**. Configure the cluster by clicking **Estimate memory size** to estimate the required cluster size using MySQL HeatWave Autopilot. See more details in Estimating Cluster Memory Size with MySQL HeatWave Autopilot.
  - Check the MySQL HeatWave Cluster configuration in the **Summary** section. Go back to the steps above to make changes if needed.
4. Click **Save**.

> **Note**
>
> The MySQL HeatWave Cluster will be offline while the configuration is being changed.

The details and configurations of the selected MySQL HeatWave Cluster is updated.

> **Note**
>
> You must reload the data into the MySQL HeatWave Cluster after changing the configuration.

4.2.3 Deleting a MySQL HeatWave Cluster

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

To delete a MySQL HeatWave Cluster:

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

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

4.3 Viewing MySQL HeatWave Cluster Details

To view MySQL HeatWave Cluster Details:

1. In the MySQL HeatWave Console, select the **Resources** tab, and then the **MySQL HeatWave Clusters** tab.

Copyright © 2022, 2025, Oracle and/or its affiliates.

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

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

- MySQL HeatWave Cluster Details

4.3.1 MySQL HeatWave Cluster Details

The **MySQL HeatWave Cluster** information page contains the following tabs:

Table 4-2 Tabs on the MySQL HeatWave Cluster Details Page

Name	Description
Details	MySQL HeatWave Cluster details. See Table 4-3.
Workload	General MySQL HeatWave Cluster workload details. See MySQL HeatWave Workload Performance Data.
Metrics	MySQL HeatWave Cluster metrics. See Details for the MySQL and MySQL HeatWave Cluster Performance Metrics.
Events	MySQL HeatWave Cluster Event details. See Table 4-6.

Copyright © 2022, 2025, Oracle and/or its affiliates.

Table 4-3 MySQL HeatWave Cluster Details

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

Copyright © 2022, 2025, Oracle and/or its affiliates.

Table 4-3 (Cont.) MySQL HeatWave Cluster Details

Field	Description
	• **Last Updated**: The date and time the MySQL HeatWave Cluster was last updated. • **Created**: The date and time the MySQL HeatWave Cluster was created.
MySQL HeatWave Cluster configuration	• **Shape**: The shape used for MySQL HeatWave Cluster nodes. See Supported Shapes. • **Node count**: The number of MySQL HeatWave Cluster nodes. • **Lakehouse**: Whether Lakehouse is **Enabled** or **Disabled**.
MySQL HeatWave node	See Table 4-5

Table 4-4 DB System Details

Field	Description
Name	The name of the DB System that the MySQL HeatWave Cluster is associated with

Copyright © 2022, 2025, Oracle and/or its affiliates.

Table 4-4 (Cont.) DB System Details

Field	Description
State	The state of the DB System that the MySQL 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.
Description	The user-defined description of the DB System

Table 4-5 MySQL HeatWave node Details

Field	Description
Node ID	The MySQL HeatWave node ID

Copyright © 2022, 2025, Oracle and/or its affiliates.

Table 4-5 (Cont.) MySQL HeatWave node Details

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

Table 4-6 MySQL HeatWave Cluster Event Details

Field	Description
Type	The event and its status.

Copyright © 2022, 2025, Oracle and/or its affiliates.

Table 4-6 (Cont.) MySQL HeatWave Cluster Event Details

Field	Description
Severity	The severity level of the event, which can be one of • `INFO`: General information is provided. No special concerns • `WARNING`: Warning on an event that might impact the DB System's operation if no actions are taken. • `CRITICAL`: A critical event. An urgent action is required to avoid adverse impact on your DB System.
Message	The message body of the event report.
Created	The date and time the event report was created.

See Events for details on event reports.

4.4 MySQL HeatWave Cluster Data Recovery

MySQL HeatWave triggers the data recovery process when:

- One or more MySQL HeatWave Cluster nodes fail and are restarted.

> **Note**
>
> MySQL HeatWave monitors MySQL HeatWave nodes status regularly and if there is no response from a node after 60 seconds, MySQL HeatWave automatically attempts to bring the node online.

- The MySQL HeatWave Cluster is restarted.
- The DB System is restarted.

MySQL HeatWave data can be recovered from the MySQL HeatWave Storage Layer, or reloaded from the DB System and Amazon S3.

- **Recover from the MySQL HeatWave Storage Layer**
  MySQL HeatWave first tries to recover data from the **MySQL HeatWave Storage Layer**, which was created when you enabled the MySQL HeatWave Cluster for the first time. To facilitate recovery, data is persisted to Storage Layer when data is loaded into the MySQL HeatWave Cluster and when data changes are propagated from the DB system to the MySQL HeatWave Cluster. Loading data from the MySQL HeatWave storage layer is faster because the data does not need to be converted to the MySQL HeatWave storage format, as is required when loading data from the DB system or Amazon S3.

> **Note**
>
> Automatically loaded tables are not recovered from the Storage Layer if the DB system is restarted; they are only recovered if the recovery was triggered by other reasons, and the DB system remains running during the recovery.

- **Reload from the DB system or Amazon S3**

Copyright © 2022, 2025, Oracle and/or its affiliates.

Data cannot be recovered from the MySQL HeatWave Storage Layer in these situations and **all** tables must be reloaded from the DB System or Amazon S3 for Lakehouse tables:

- Recovery from the MySQL HeatWave storage layer has failed due to Amazon S3 outage or network outage.
- The binary logs required to perform change propagation of the data restored from the MySQL HeatWave storage layer have been purged.
- The DB System has been upgraded and the data stored in the MySQL HeatWave storage layer is not compatible with the new MySQL version of the DB System.
- The recovery is triggered by the restart of a DB System of MySQL version 9.2.1 or earlier.

Even if some tables can be reloaded from the Storage Layer, some might not. After a DB system restart, the following types of tables need to be reloaded from the DB System or Amazon S3:

- Tables with dictionary encoded columns.
- Stale tables.
- Tables that were not captured in periodic metadata checkpoints.
- Tables where partition load or unload operations were in progress when the DB system stopped abruptly.

> **Note**
>
> Automatically loaded tables are not reloaded during the recovery from the DB System or Amazon S3.

During recovery, the MySQL HeatWave Cluster automatically reloads the data. However, if the MySQL Server is in the `SUPER_READ_ONLY` mode, you cannot load data into the MySQL HeatWave Cluster, and the recovery fails. Disable the `SUPER_READ_ONLY` mode to load data. See Resolving SUPER_READ_ONLY and OFFLINE_MODE Issue Caused by Low Free Storage Space.

See Data Load Progress and Status Monitoring on how to check the progress and status of data loading.

> **Note**
>
> When you unload a table, the data is removed from the MySQL HeatWave Cluster, and it is also removed from AWS S3 too in a background operation. An unloaded table is not recoverable into the MySQL HeatWave Cluster by the data recovery process, and must be loaded again manually.

Copyright © 2022, 2025, Oracle and/or its affiliates.

5
Connecting to a DB System

Clients and applications interact with a MySQL HeatWave Cluster by connecting to the DB System. This chapter describes how to connect to the 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 from a Client
- Connecting Using a PrivateLink
- MySQL Connectors
- Enabling Host Name Identity Verification

5.1 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 **Workspace** page to connect to a DB System, and then manage MySQL HeatWave Cluster data and use the **Query Editor** to run DB System and MySQL HeatWave queries. For more information, see Manage Data in MySQL HeatWave, 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 **Workspace** 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**.

5.2 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.

Copyright © 2022, 2025, Oracle and/or its affiliates.

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 Editing a 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 Editing a 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 MySQL client and application connections over the public internet are encrypted. Private connections over PrivateLinks are also encrypted by default. For DB Systems that only accept private connections (i.e., not accessible from public IP addresses), if you need to disable encryption for your client connections (for example, due to certain client-side limitations), you can update the MySQL configuration of your DB System to set the system variable `require_secure_transport` to `OFF`, and have clients connect to the MySQL DB System through a PrivateLink. 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).

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
- Connecting with MySQL Command-Line Client
- Connecting with MySQL Workbench

5.2.1 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.

Copyright © 2022, 2025, Oracle and/or its affiliates.

- 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 >
   ```

5.2.2 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.

Copyright © 2022, 2025, Oracle and/or its affiliates.

- 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>
   ```

5.2.3 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

Copyright © 2022, 2025, Oracle and/or its affiliates.

- **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.

5.3 Connecting Using a PrivateLink

You can connect directly to a DB System using an AWS PrivateLink. See Connecting to a DB System With a PrivateLink for details.

5.4 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

5.5 Enabling Host Name Identity Verification

MySQL HeatWave on AWS supports TLSv1.2 and requires that all MySQL client and application connections over the public internet are encrypted. Private connections over PrivateLinks are also encrypted by default. For DB Systems that only accept private connections (i.e., not accessible from public IP addresses), if you need to disable encryption for your client connections (for example, due to certain client-side limitations), you can update the MySQL configuration of your DB System to set the system variable `require_secure_transport` to `OFF`, and have clients connect to the MySQL DB System through a PrivateLink. 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).

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 supported regions:

Copyright © 2022, 2025, Oracle and/or its affiliates.

- AWS US East (N. Virginia) Region (`us-east-1`): https://cloud.mysql.com/us-east-1/aws-us-east-1-cabundle.pem
- Germany Central (Frankfurt) Region (`eu-central-1`): https://cloud.mysql.com/eu-central-1/aws-eu-central-1-cabundle.pem
- India West (Mumbai) Region (`ap-south-1`): https://cloud.mysql.com/ap-south-1/aws-ap-south-1-cabundle.pem
- Japan East (Tokyo) Region (`ap-northeast-1`): https://cloud.mysql.com/ap-northeast-1/aws-ap-northeast-1-cabundle.pem
- Mexico Central (Queretaro) Region (`mx-central-1`): https://cloud.mysql.com/mx-central-1/aws-mx-central-1-cabundle.pem
- UK South (London) Region (`eu-west-2`): https://cloud.mysql.com/eu-west-2/aws-eu-west-2-cabundle.pem
- South Korea Central (Seoul) Region (`ap-northeast-2`): https://cloud.mysql.com/ap-northeast-2/aws-ap-northeast-2-cabundle.pem
- US West (San Jose) Region (`us-west-2`): https://cloud.mysql.com/us-west-2/aws-us-west-2-cabundle.pem

> **Note**
>
> MySQL HeatWave on AWS is currently supported in the regions mentioned in Region Availability. Certificate bundles for other regions will be made available as support for MySQL HeatWave on AWS is extended to other regions.

Copyright © 2022, 2025, Oracle and/or its affiliates.

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

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


curl -o aws-ap-south-1-cabundle.pem \
https://cloud.mysql.com/ap-south-1/aws-ap-south-1-cabundle.pem


curl -o aws-eu-central-1-cabundle.pem \
https://cloud.mysql.com/eu-central-1/aws-eu-central-1-cabundle.pem


curl -o aws-ap-northeast-1-cabundle.pem \
https://cloud.mysql.com/ap-northeast-1/aws-ap-northeast-1-cabundle.pem


curl -o aws-eu-west-2-cabundle.pem \
https://cloud.mysql.com/eu-west-2/aws-eu-west-2-cabundle.pem


curl -o aws-mx-central-1-cabundle.pem \
https://cloud.mysql.com/mx-central-1/aws-mx-central-1-cabundle.pem


curl -o aws-ap-northeast-2-cabundle.pem \
https://cloud.mysql.com/ap-northeast-2/aws-ap-northeast-2-cabundle.pem


curl -o aws-us-west-2-cabundle.pem \
https://cloud.mysql.com/us-west-2/aws-us-west-2-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
```


Copyright © 2022, 2025, Oracle and/or its affiliates.

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.

Copyright © 2022, 2025, Oracle and/or its affiliates.

6 PrivateLink

PrivateLink for HeatWave on AWS allows you to configure private network paths between your MySQL DB System and your AWS account. You can use PrivateLink to ensure that all traffic to or from your DB System stays on a private network connection and does not transit the public internet.

At a high level, AWS PrivateLink works by establishing network components within two AWS accounts—your AWS Account and the HeatWave on AWS account. Within one account, an **Endpoint service** is created, and in the other account, a **VPC endpoint** is created to link to the **Endpoint service** through a uniquely identified connection. The architecture of PrivateLink and the involved components are explained in AWS PrivateLink concepts.

You can use a PrivateLink to connect your applications to your DB System, or to allow your DB System to connect to services hosted in your AWS account. MySQL HeatWave on AWS supports two types of PrivateLinks:

- **Query PrivateLinks**: For applications running in your AWS account to access a MySQL HeatWave on AWS DB System over a private connection in order to manage it or to run queries. See Deploying a Query PrivateLink for details, and see Troubleshooting a Query PrivateLink for help on common issues.
- **Egress PrivateLinks**: For your DB System to access an external service running in your AWS account—for example, to support Inbound Replication from a MySQL source server. See Deploying an Egress PrivateLink for details, and see Troubleshooting an Egress PrivateLink for help on common issues.

This chapter contains the following topics on PrivateLinks.

- Deploying a Query PrivateLink
- Deploying an Egress PrivateLink
- Managing a PrivateLink
- Billing for PrivateLinks
- PrivateLink Limitations

6.1 Deploying a Query PrivateLink

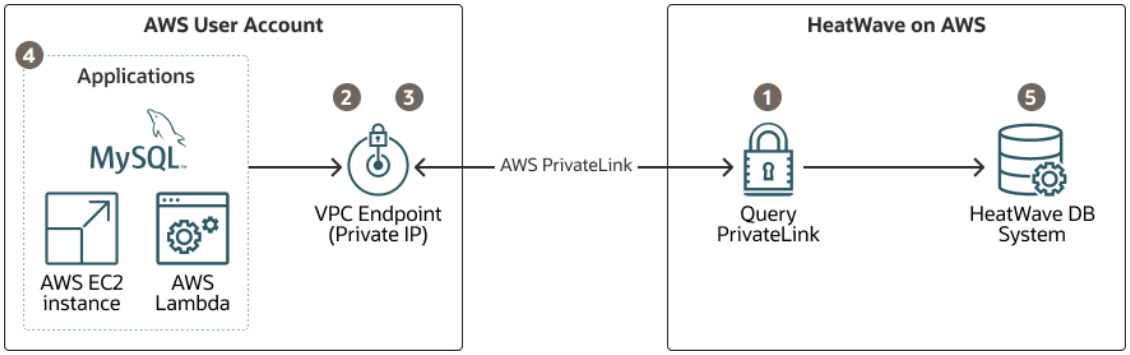
Overview

To connect from your AWS account to a MySQL HeatWave on AWS DB System over a private network path, you need to configure a Query PrivateLink inside MySQL HeatWave on AWS, and then create a VPC Endpoint inside your AWS account. Once these network components are configured, your applications connect to your DB System through the VPC Endpoint using a **Private Hostname** that is only visible within the VPC where the Endpoint exists.

Figure 6-1 illustrates the different components that make up a Query PrivateLink connection from your AWS VPC to your MySQL HeatWave on AWS DB System.

Copyright © 2022, 2025, Oracle and/or its affiliates.

Figure 6-1 Components of a Query PrivateLink



Steps for Deploying a Query PrivateLink

Follow these steps to get connected to an existing MySQL HeatWave on AWS DB System over a Query PrivateLink:

1. Create a Query PrivateLink inside HeatWave on AWS and authorize your AWS accounts to connect to the Query PrivateLink by following the steps given in Creating a Query PrivateLink.
2. Grant principals in your AWS accounts the permissions to create and delete VPC endpoints by following the instructions given in Configuring IAM Policies for Endpoints for a Query PrivateLink.
3. Create a VPC endpoint in your AWS account by following the steps given in Creating an Endpoint for a Query PrivateLink
4. Update your applications to connect to your MySQL HeatWave on AWS DB System using its private hostname, which is displayed as **Hostname** on the PrivateLink Details page and as **PrivateLink Hostname** on the MySQL DB System Details page in the HeatWave Console. See Connecting to a DB System With a PrivateLink.
5. (Optional) For security purposes, remove public connectivity to your MySQL HeatWave on AWS DB System if it is no longer needed. Follow the instructions given in Update Networking to make sure **Enable inbound connectivity from allowed public IP address ranges** is deselected.

The following sections explain the steps above in detail and provide additional information on the deployment of Query PrivateLinks.

- Creating a Query PrivateLink
- Configuring IAM Policies for Endpoints for a Query PrivateLink
- Creating an Endpoint for a Query PrivateLink
- Connecting to a DB System With a PrivateLink
- Troubleshooting a Query PrivateLink

6.1.1 Creating a Query PrivateLink

Use the MySQL HeatWave Console to create a Query PrivateLink to provide connectivity for applications running in your AWS account to MySQL HeatWave on AWS DB Systems using private IP addresses.

This task requires the following:

Copyright © 2022, 2025, Oracle and/or its affiliates.

- A DB System in the `Active` state.
- ARNs of authorized principals.

Do the following to create a Query PrivateLink:

1. In the MySQL HeatWave Console, select the **Resources** tab.
2. On the **PrivateLinks** tab, click **Create PrivateLink.**
3. Enter the following:
   - **Basic information**:
     - **Display name**: Specify a display name for the PrivateLink or use the generated default name.
     - **Description**: (Optional) Specify a description for the PrivateLink.
   - **Select PrivateLink type**: Select **Query** and click **Next** to provide the following information on the next page:
     - **Target DB System**: Select the DB System with which you want to associate the PrivateLink.
     - **ARNs of Authorized Principals**: Amazon Resource Names (ARNs) to create connections to the PrivateLink. Specify either of the following:
       * (Recommended) Your entire AWS account (identified by your AWS `ACCOUNT_ID`) in the following format:

         ```
         arn:aws:iam::<ACCOUNT_ID>:root
         ```

       * Specific principals in the following format:

         ```
         arn:aws:iam::<ACCOUNT_ID>:user/<user_id>
         ```

         ```
         arn:aws:iam::<ACCOUNT_ID>:role/<role_id>
         ```

         You can specify multiple ARNs by delimiting them by semicolons.

         Authorizing specific principals enhances security, as the authorization to create a PrivateLink is checked twice: first inside the AWS account requesting the new endpoint and then in MySQL HeatWave on AWS, to ensure that the entity requesting the endpoint is in the set of authorized principals.
4. Click **Create**.

You can see the details of the query PrivateLink displayed, including a new **Hostname** and a new **Service name**—they are needed in the next few steps for deploying a Query PrivateLink.

6.1.2 Configuring IAM Policies for Endpoints for a Query PrivateLink

Subsequent steps for the deployment of a Query PrivateLink requires the abilities to create and delete VPC endpoints in your AWS account. This section helps to ensure that the desired principals in your AWS Account have the permissions for those actions.

If you are already using AWS-managed IAM policies for permission management, check that your current AWS-managed IAM policies grant principals the permission to create and delete VPC endpoints. If not, add the appropriate managed policies to the principals you wish to

Copyright © 2022, 2025, Oracle and/or its affiliates.

authorize for VPC endpoint management. The following AWS-managed policies enable your IAM principals to create and delete endpoints:

- `NetworkAdministrator`
- `AmazonVPCFullAccess` and `AmazonRoute53FullAccess`

These AWS managed policies grant broader permissions than those strictly required for deploying PrivateLinks. For enhanced security, administer your AWS account to Grant least privilege to your IAM principals using a customer-managed policy that permits the Actions of **CreateVpcEndpoint** and **DeleteVpcEndpoint**.

You can also limit your IAM principals to only perform operations against a specific PrivateLink by specifying its service name (available on the Viewing PrivateLink Details page) with the following Condition:

```
"Condition": {
              "StringEquals": {
                 "ec2:VpceServiceName": "<privatelink-service-name>"
        }
    }
```

Or, you can restrict your principals to perform operations against all PrivateLinks in HeatWave on AWS by specifying the following service owner Condition:

```
"Condition": {
             "StringEquals": {
                 "ec2:VpceServiceOwner": "612981981079"
        }
    }
```

See Control the service names that can be specified for VPC endpoint services.

6.1.3 Creating an Endpoint for a Query PrivateLink

Use the AWS Management Console to create an endpoint in the same region and availability zone as the DB System.

This task requires the following:

- Access to AWS Management Console.
- The service name of the PrivateLink. See Viewing PrivateLink Details.
- Properly configured policies to create an endpoint. See Configuring IAM Policies for Endpoints for a Query PrivateLink.
- A VPC with subnet(s) in the same availability zone(s) as the DB System. If you are creating a PrivateLink for a Highly Available DB System, your VPC must have a subnet in each of the availability zones shown in the **Placement** field of the DB System. See Creating a VPC, and Table 3-8.

Do the following to create an endpoint:

1. Open the AWS Management Console and sign in with your credentials.
2. Switch to the same region as the DB System.
3. In the AWS Management Console home page, click **Services**, click **Networking & Content Delivery**, and then click **VPC**.

Copyright © 2022, 2025, Oracle and/or its affiliates.

4. In the navigation pane of the Console, under **Virtual private cloud**, click **Endpoints**, and then click **Create endpoint**.
5. Enter the following:
   a. **Endpoint settings**:
      i. **Name tag**: (Optional) Specify a name for the endpoint.
      ii. **Type**: Select **Endpoint services that use NLBs and GWLBs**.
   b. **Service settings**:
      i. **Service name**: Specify the service name of the PrivateLink you created in Creating a Query PrivateLink. Provide the full service name, which starts with `com.amazonaws.com...` (see Viewing PrivateLink Details on how to find it).
      ii. Click **Verify service**.
      If service verification is unsuccessful, ensure that
      - The **Service name** is entered correctly.
      - The endpoint is created in the same region as the PrivateLink.
      - The authorized principals field in the PrivateLink is correct, and your IAM permissions are configured accordingly. See Updating Authorized Principals for a Query PrivateLink.

      See Troubleshooting a Query PrivateLink for details.
   c. **VPC**:
      i. **VPC**: Select the VPC in which to create the endpoint. Only subnets that are in the same Availability Zone as the DB System will be available for selection.
      ii. Click **Additional settings**.
      iii. **Enable DNS name**: It is recommended to check this box. Checking this box configures your VPC to automatically resolve the PrivateLink hostname (visible in the HeatWave console) to the VPC endpoint's IP address. If unchecked, you need to configure DNS resolution manually in your VPC.
   d. **Subnet(s)**: Select the subnet(s) in which you wish to create the endpoint. The AWS console automatically selects the availability zone(s) in your account that match the DB System, and lists your subnet(s) in those availability zone(s). If you are creating a PrivateLink for a Highly Available DB System, ensure that you create an endpoint in each of the three displayed availability zones. If you wish to connect to this endpoint from another subnet that is not displayed in the above list, you must configure your network infrastructure to allow routing between the two subnets. See Subnets.
   e. **Security groups**: Select the security groups in your account to associate with the endpoint. These security groups must allow inbound traffic from your applications. For more information, see Control traffic to your AWS resources using security groups.
6. Click **Create endpoint**.

6.1.4 Connecting to a DB System With a PrivateLink

Connect to a DB System in MySQL HeatWave on AWS with a PrivateLink using a MySQL clients (for example, `mysql`or MySQL Shell) or MySQL Connectors.

This task requires the following:

- A DB System in MySQL HeatWave on AWS.

Copyright © 2022, 2025, Oracle and/or its affiliates.

- A correctly configured PrivateLink. See Creating a Query PrivateLink, and knowledge of its private hostname, which is displayed as **Hostname** on the PrivateLink Details page and as **PrivateLink Hostname** on the MySQL DB System Details page in the HeatWave Console.
- A MySQL client that has network access to the same VPC as the endpoint for the PrivateLink. See Creating an Endpoint for a Query PrivateLink.
- If the MySQL client is connecting to the DB System with the `VERIFY_CA` or `VERIFY_IDENTITY` connection mode, a HeatWave Certificate Authority bundle. See Enabling Host Name Identity Verification.

Connect to DB System in MySQL HeatWave on AWS as you normally would (see Connecting from a Client), but use the PrivateLink's hostname as your hostname. For example, run a `mysql` client in your AWS account with the following command:

```
mysql -h <hostname> -u <username> -p <password> --ssl-mode VERIFY_IDENTITY --
ssl-ca
      <ca-file>
```

- `<hostname>`: Specify the hostname of the PrivateLink.
- `<username>`: Specify the MySQL user name.
- `<password>`: Specify the MySQL user password.
- `<ca-file>`: Specify the name of the HeatWave Certificate Authority certificate.

You are now connected to a DB System in MySQL HeatWave on AWS over a PrivateLink. If you fail to connect, see Troubleshooting a Query PrivateLink for help.

6.1.5 Troubleshooting a Query PrivateLink

Table 6-1 lists some common issues and their resolutions when setting up a Query PrivateLink, and Troubleshooting Connectivity in your AWS Account Using the AWS Reachability Analyzer for a Query PrivateLink gives further help for toubleshooting.

> **Note**
>
> The error messages might read differently, as MySQL HeatWave on AWS gets updated.

Table 6-1 Query PrivateLink: Common Issues and Resolutions

Deployment Step	Error Summary	Steps to Resolve
Create Query PrivateLink	*CreatePrivateLink: 1 Validation Error for … principalArn*	Ensure the ARNs you have entered are formatted properly. See Identify AWS resources with ARNs.

Copyright © 2022, 2025, Oracle and/or its affiliates.

Table 6-1 (Cont.) Query PrivateLink: Common Issues and Resolutions

Deployment Step	Error Summary	Steps to Resolve
Create VPC Endpoint	AWS Console shows *Service name could not be verified*	• Ensure that you are creating the Endpoint in the same Region as the Query PrivateLink. • Check the **ARNs Authorized Principals** for the PrivateLink match a principal in your account, or authorize your entire account for simplicity. See Creating a Query PrivateLink.
Connect to Private Hostname	Application cannot connect, returning MySQL error like *Error 2003: Can't connect to MySQL server*	Validate that your account allows connectivity between your application and the VPC Endpoint. See Troubleshooting Connectivity in your AWS Account Using the AWS Reachability Analyzer for a Query PrivateLinkbelow.

Troubleshooting Connectivity in your AWS Account Using the AWS Reachability Analyzer for a Query PrivateLink

To identify any components in your account that may be preventing your Query PrivateLink from working, we recommend using the AWS Reachability Analyzer:

- Navigate to **AWS Console** > **Network Manager** > **Reachability Analyzer** and click **Create analyze path**.
- Under **Path Source**, put in an optional **Name tag** if you want, and set:
  - **Source type** as **Instances** if your application runs on an EC2 instance, or as **Network Interfaces** if your application connects through an AWS-managed network interface such as one created by AWS Lambda.
  - **Source** as the EC2 instance ID, or the interface ID of the Lambda network interface.
- Under **Path Destination**, put in an optional **Name tag**, set:
  - **Destination type** as **VPC Endpoints**.
  - **Destination** as the ID of the VPC Endpoint created in your account. See Creating an Endpoint for a Query PrivateLink.
- For **Protocol** choose **TCP**
- Add optional **Tags** if you want.
- Click **Create analyze path**, and wait for the analysis to complete. The network components that forward traffic between the Network Load Balancer and database source are displayed. If traffic is blocked, the problematic component(s) are identified.

Copyright © 2022, 2025, Oracle and/or its affiliates.

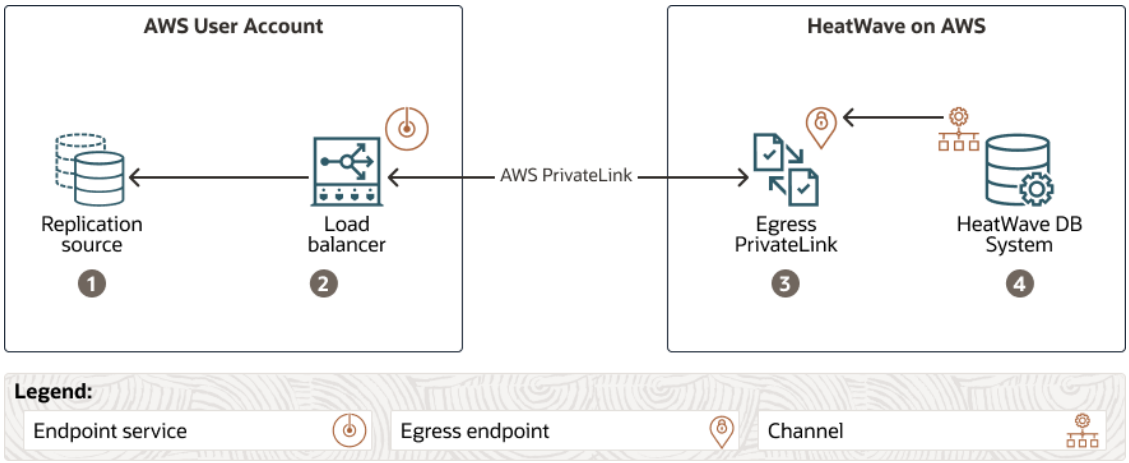
6.2 Deploying an Egress PrivateLink

Overview

Certain features like inbound replication require egress connections from your MySQL HeatWave on AWS DB System to your AWS account. To allow your DB System to connect to an endpoint in your AWS account using a private network path, you first configure your account's network infrastructure and then create an Egress PrivateLink inside MySQL HeatWave on AWS. During the creation of the Egress PrivateLink, you configure a set of Egress Endpoints that define the network parameters your DB System will use to connect to your AWS account over the Egress PrivateLink.

Figure 6-2 shows the components and steps required to configure your account's network infrastructure and create an Egress PrivateLink. It illustrates the use case of configuring inbound replication from a source in your AWS account to a MySQL HeatWave DB System using a replication Channel over an Egress PrivateLink.

Figure 6-2 Components to Configure for Inbound Replication Using an Egress PrivateLink



Steps for Deploying an Egress PrivateLink for Inbound Replication

Follow these steps to create the inbound replication scenario as illustrated in Figure 6-2.

1. Configure your replication source by:
   a. Following the instructions given in Source Configuration
   b. Creating a Replication User On a Source Server
2. Create the requisite networking resources in your AWS account by following the instructions given in Configuring the Network Infrastructure for an Egress PrivateLink.
3. Create an Egress PrivateLink by following the instructions given in Creating an Egress PrivateLink.
4. Create the Replication Channel in the MySQL HeatWave Console following the instructions in Creating a Channel, selecting **Egress PrivateLink** for **Channel over PrivateLink**.

The following sections explain some of the above-mentioned steps in details and provide additional information on the deployment of Egress PrivateLinks.

Copyright © 2022, 2025, Oracle and/or its affiliates.

- Configuring the Network Infrastructure for an Egress PrivateLink
- Creating an Egress PrivateLink
- Troubleshooting an Egress PrivateLink

6.2.1 Configuring the Network Infrastructure for an Egress PrivateLink

Before creating an Egress PrivateLink, configure the network infrastructure in your AWS account.

These are the prerequisites for this task:

- Access to your AWS account through the AWS Console.
- Knowledge of the following information regarding your service (for example, a source database that replicates to your DB System in MySQL HeatWave on AWS):
  - The VPC that is hosting the service you wish to expose to HeatWave on AWS (for example, a database instance).
  - The AWS Availability Zone IDs of that service
  - The IP address on which your service should be reached. For example, if your service is a database source such as an Amazon RDS or Aurora instance, run this command on any internet-connected system using the hostname of its primary endpoint:

    ```
    nslookup <service hostname>
    ```

To configure your service and its surrounding network infrastructure, you need to: (1) Create and configure a Target Group, (2) Create and configure a Network Load Balancer, (3) Allow your service to receive traffic from your Network Load Balancer, and (4) Create and configure an endpoint service. Here are the details for each step:

1. Create and configure a Target Group:
   - Go to **AWS Console** > **EC2** > **Target Groups** (also reachable from the console's navigation pane under **Load Balancing**), and select **Create target group**
   - Under **Basic configuration**
     - Select **IP addresses**
     - Add a **Target group name**
     - For **Protocol: Port**, select **TCP** for protocol, and enter the port number of your service (for example, the port number of a MySQL database source).
     - Under **VPC**, select the VPC in which the service is located to be included into the Target Group
   - Under **Health checks**
     - Select **TCP** for **Health check protocol**
     - Under **Advanced health check settings**, select **Override**, and enter any port number other than the port for your service (e.g., 40000).

       This setting will force the Target Group's health status to "Unhealthy". Your service will still be accessible over the PrivateLink, because the load balancer will continue to forward connections. For details, see Health checks for Network Load Balancer target groups.

Copyright © 2022, 2025, Oracle and/or its affiliates.

> **Warning**
>
> If your service is a MySQL database that acts as a replication source, depending on the its configuration, failing to adjust the health check port may result in your source blocking all traffic from the load balancer. For details, see the MySQL Server documentation pertaining to the host cache.

- Click **Next**, to go to the **Register targets** page
- Go to **Step 2** on the **Register targets** page and enter the **IP addresses** for your service (see the prerequisites above).
- Under **Ports**, make sure the port number for the service is correct.
- Click **Include as pending below**, then click **Create target group**.

> **Note**
>
> If the IP address of your service changes, you must update the Target Group's IP addresses so that the service can continue to be reachable.

2. Create and configure a Network Load Balancer:
   - Navigate to **AWS Console** > **EC2** > **Load Balancers**, and click **Create load balancer**. The **Compare and select load balancer type** page opens.
   - Under **Load balancer types**, select **Network Load Balancer** by clicking the **Create** button under its description. The **Create Network Load Balancer** page opens.
   - Enter a **Load balancer name**.
   - For **Scheme**, select **Internal**.
   - Under **Network mapping**:
     - Make sure the service's VPC is selected.
     - For **Mappings**, select the Availability Zone IDs for your service.

       > **Note**
       >
       > Your Network Load Balancer's **Network Mapping** must include **all** Availability Zone ID(s) of your target MySQL HeatWave on AWS DB System(s). For example, if your standalone DB System is placed in `use1-az1`, your **Network Mapping** must include `use1-az1`. If your High Availability DB System is placed in `use1-az2`, `use1-az3`, and `use1-az4` (see the **Placement** of your DB System), your **Network Mapping** must include `use1-az2`, `use1-az3`, and `use1-az4`.

     - For each of the Availability Zone IDs you selected, pick a private **Subnet**
   - Under **Security Groups**, click **create a new security group**. In the **Create security group** page that opens, configure the following before you click **Create security group**:
     - **Security group name**: Give a group name.
     - **Description**: Give a description.

Copyright © 2022, 2025, Oracle and/or its affiliates.

  - For **VPC**, select the service's VPC.
  - For **Inbound rules** and **Outbound rules**, keep the default configurations.
- Go back to the **Create Network Load Balancers** page and under **Security Groups**, select the security group you just created and deselect any other security groups.
- Under **Listeners and routing**, configure the **Listener** with the **Port** the service is on, and **Default action** to **Forward to** and select the target group you created above in Step 1.
- Click **Create load balancer**. The load balancer is created.
- Select in **AWS Console** > **EC2** > **Load Balancers** the load balancer you just created. A configuration page for the load balancer is opened.
- Scroll down and select the **Security** tab. Make sure **Enforce inbound rules on PrivateLink traffic** is **Off**. If it is not, click **Edit** and then deselect it.

> **Note**
>
> If your Network Load Balancer is configured in multiple Availability Zones follow the steps below to ensure the DB System can reach your service, regardless of the Availability Zone(s) in which your service is located:
>
> - Return to the load balancer configuration page by navigating to **AWS Console** > **EC2** > **Load Balancers** and selecting your load balancer.
> - Select the **Attributes** tab and click **Edit**.
> - Under **Availability Zone routing configuration** > **Load balancer targets selection policy**, select **Enable cross-zone load balancing** and click **Save changes**. See Billing for PrivateLinks.

3. Allow your service to receive traffic from your Network Load Balancer:
   - Go to the configuration page for your service (for example, for a database instance on AWS RDS, go to **AWS** > **RDS** > **Databases** > *your instance*). Under **Connectivity and security** > **Security**, click on the security group associated with the database instance. The **Security Groups** page opens.
   - Check the security group, and then select **Edit inbound rules** under **Actions**. The **Edit inbound rules** page opens.
   - Click **Add rule** and add a new rule with the following specifications:
     - Choose **Custom TCP** for **Type**.
     - Choose **Custom** and then search for and select the security group created in Step 2 above for the Network Load Balancer.
     - Enter the service instance's port number for **Port Range**.
     - Provide an optional **Description**.
   - Click **Save rules**
4. Create and configure an endpoint service:
   - Navigate to **AWS**> **VPC** > **Endpoint services**, and click **Create endpoint service**. The **Create endpoint service** page opens.
   - Configure your endpoint service with the following information:
     - **Load balancer type**: Select **Network**

Copyright © 2022, 2025, Oracle and/or its affiliates.

  - Under **Available load balancers**, select the load balancer you created in Step 2 above.
  - Under **Additional settings**:
    * Select **Acceptance required**
    * Select **IPv4** for **Supported IP address types**
    * Ensure **Enable private DNS names** is NOT selected, because it is unnecessary.
- Click **Create**. Note the **Service name** under **Details** of the VPC endpoint service you created. You will need it to configure your Egress PrivateLink.
- Navigate to **AWS** > **VPC** > **Endpoint services** and choose the endpoint service you just created, and then under the **Allow principals** tab (or under **Actions**) choose **Allow principals**. The **Allow principals** page opens.
- Under **Principals to add**, add the ARN of the MySQL HeatWave on AWS account:

  ```
  arn:aws:iam::612981981079:root
  ```

- Click **Allow principals**

6.2.2 Creating an Egress PrivateLink

Use the MySQL HeatWave Console to create an Egress PrivateLink for providing private connectivity to an external system running in your AWS account. It can be used to replicate data into your DB Systems inside MySQL HeatWave on AWS.

This task requires the following:

- A DB System in the `Active` state.
- A VPC Endpoint Service name in your AWS account that provides connectivity to your source database. See how to set up an endpoint service for your source in Configuring the Network Infrastructure for an Egress PrivateLink.
- **If you want TLS certificate identity verification for the replication channel:** The endpoint hostname for accessing the source database. If you are using, for example, an AWS RDS source, obtain the endpoint hostname using the AWS console and browse to **RDS** > **Databases** > (your source database instance) > **Connectivity and security**.

Do the following to create an Egress PrivateLink:

1. In the MySQL HeatWave Console, select the **Resources** tab.
2. On the **PrivateLinks** tab, click **Create PrivateLink.**
3. Enter the following:
   - **Basic information**:
     - **Display name**: Specify a display name for the PrivateLink or use the generated default name.
     - **Description**: (Optional) Specify a description for the PrivateLink.
   - **Select PrivateLink type**: Select **Egress** and click **Next** to finish configuring the Egress PrivateLink on the next page:
     - **Configure external endpoint service name**: Set this to the VPC endpoint service name in your AWS account to which this PrivateLink will connect. See how to set up an endpoint service for your source in Configuring the Network Infrastructure

Copyright © 2022, 2025, Oracle and/or its affiliates.

for an Egress PrivateLink. Make sure to provide the full service name, which is in the format of `com.amazonaws.vpce.<region>.vpce-svc-<id>`.

- **Configure egress endpoints**: Configure the list of endpoints for which this Egress PrivateLink is to provide connectivity. Provide the following information for each endpoint:
  - **Source Hostname**: (Optional) The endpoint of the source database on AWS. Only required if you want to support TLS certificate identity verification; leave blank otherwise.
  - **Source Port**: The port on which this egress endpoint provides connectivity.
  - **Target DB System**: Select the target DB System that is allowed to connect with this Egress PrivateLink.

> **Note**
>
> Currently, only a single egress endpoint is supported for each Egress PrivateLink. If you would like to configure more egress endpoints for replication, create a separate egress PrivateLink for each of the endpoints.

4. Click **Create**.
5. Return to the Endpoint Service page in the AWS Console (see Configuring the Network Infrastructure for an Egress PrivateLink) to explicitly accept the connection :
   - Navigate to **AWS Console** > **VPC** > **Endpoint services** and choose the endpoint service you created in Step 4 above.
   - On the **Endpoint connections** tab, choose the Endpoint ID that matches the Endpoint ID of your Egress PrivateLink (see PrivateLink Details page for the information). **Do not accept connections from any unknown Endpoint IDs**.
   - Under **Actions** choose **Accept endpoint connection request**.
   - In the **Accept endpoint connection request** dialog box that opens, type "accept" in the *accept* field, and click **Accept**.

   The **State** of the Endpoint ID becomes **Available** after some time, and the endpoint connection is now established.

6.2.3 Troubleshooting an Egress PrivateLink

Table 6-1 lists some common issues and their resolutions when setting up an Egress PrivateLink, and Troubleshooting Connectivity in your AWS Account Using the AWS Reachability Analyzer for an Egress PrivateLink gives further help for troubleshooting.

> **Note**
>
> The error messages might read differently, as MySQL HeatWave on AWS gets updated.

Copyright © 2022, 2025, Oracle and/or its affiliates.

Table 6-2 Egress PrivateLink: Common Issues and Resolutions

Deployment Step	Error Summary	Steps to Resolve
Create Egress PrivateLink	*Could not Create PrivateLink. Unable to connect to Endpoint Service.*	Ensure your Endpoint Service is in the correct Availability Zone, and you have added the HeatWave on AWS account ARN to the list of Allowed Principals. See Configuring the Network Infrastructure for an Egress PrivateLink for details.
	Could not Create PrivateLink. DB Systems <`dbsystem-ids`> do not share a single physical Availability Zone with the endpoint service. Please ensure the physical Availability Zone(s) provided by your endpoint service include `az-id1`, `az-id2`, `az-id3` for these DB System(s) and then retry your request.	Your VPC Endpoint Service, Network Load Balancer, and MySQL HeatWave on AWS DB System must share the same AWS Availability Zone ID(s). Add Network Mappings to your Network Load Balancer for all the Availability Zone ID(s) mentioned in the error summary and ensure Cross-Zone Load Balancing is enabled for your Network Load Balancer (see Configuring the Network Infrastructure for an Egress PrivateLink and Billing for PrivateLinks), or create a new DB System with the same Availability Zone ID(s) of your VPC Endpoint Service.
Create Replication Channel, when using Egress PrivateLink for Inbound Replication	*Could not create MySQL Channel. PrivateLink does not have an Egress Endpoint to provide connectivity to the channel.*	Ensure the values for **Hostname** and **Port** match exactly one of the Egress PrivateLink's Egress Endpoints. If you have left the Egress Endpoint hostname blank, use the **Default hostname** of the Egress PrivateLink.
	Channel enters **Needs Attention** state. Click on the **(i) tooltip** near the **Channel State** on the Channel Details page.	Ensure you have accepted the endpoint connection in your AWS account. See Creating an Egress PrivateLink. Validate that your account allows connectivity between your Network Load Balancer and replication source. See Configuring the Network Infrastructure for an Egress PrivateLink.
	Other messages that indicate a problem with the replication process (for example, *The Channel is not receiving transactions due to error(s): Cannot connect to MySQL server….*)	See Trouble Shooting Replication

Copyright © 2022, 2025, Oracle and/or its affiliates.

Table 6-2 (Cont.) Egress PrivateLink: Common Issues and Resolutions

Deployment Step	Error Summary	Steps to Resolve
Update Egress Endpoints	*Could not Update Egress PrivateLink. DB System(s) <dbsystem-ids> cannot be connected to Egress PrivateLink <privatelink-id> because they have Availability Zone(s) az-id1, az-id2, az-id3 and the PrivateLink was created in Availability Zone(s) az-id1, az-id4, az-id5. Please create a new PrivateLink linked to these DB System(s).*	The DB System(s) <dbsystem-id> cannot be added as an Egress Endpoint to Egress PrivateLink <privatelink-id>, because the Egress PrivateLink's Availability Zone(s) do not match the Availability Zone(s) of the DB System(s). Create a new Egress PrivateLink with this DB System(s) as an Egress Endpoint(s). See Creating an Egress PrivateLink.

Troubleshooting Connectivity in your AWS Account Using the AWS Reachability Analyzer for an Egress PrivateLink

To identify any components in your account that may be preventing your Egress PrivateLink from working, we recommend using the AWS Reachability Analyzer:

- Navigate to **AWS Console** > **Network Manager** > **Reachability Analyzer** and click **Create analyze path**.
- Under **Path Source**, put in an optional **Name tag** if you want, and set:
  - **Source type** as **Network Interfaces** or **IP Addresses**
  - **Source** as the Network Interface ID or IP address of your Network Load Balancer. See this article on how to find the Network Load Balancer's network interface IDs or IPs.
- Under **Path Destination**, put in an optional **Name tag**, set:
  - **Destination type** as **Network Interfaces** or **IP Addresses**
  - **Destination** as the Network Interface ID or IP address of your service (for example, the source database for replication).
- For **Protocol** choose **TCP**
- Add optional **Tags** if you want.
- Click **Create analyze path**, and wait for the analysis to complete. The network components that forward traffic between the Network Load Balancer and your service are displayed. If traffic is blocked, the problematic component(s) are identified.

6.3 Managing a PrivateLink

Use the MySQL HeatWave Console to edit the name or description of a PrivateLink, update its authorized principals or Egress Endpoints, delete the PrivateLink, or view its details.

- Editing a PrivateLink
- Deleting a PrivateLink
- Updating Authorized Principals for a Query PrivateLink
- Updating Egress Endpoints for a Egress PrivateLink
- Viewing PrivateLink Details

Copyright © 2022, 2025, Oracle and/or its affiliates.

6.3.1 Editing a PrivateLink

Use the MySQL HeatWave Console to edit the name or description of a PrivateLink.

This task requires the following:

- A PrivateLink in the `Active` state.

Do the following to edit a PrivateLink:

1. In the MySQL HeatWave Console, select the **Resources** tab.
2. On the **PrivateLink** tab, in the list of PrivateLinks, find the PrivateLink to edit, and do one of the following:
   - Click the row of the PrivateLink to highlight it, and click **Edit PrivateLink**.
   - Click the name of the PrivateLink to open the **PrivateLink Details** page, and click **Edit PrivateLink**.
3. Enter the following:
   - **Basic information**:
     - **Display name**: Edit the display name of the **PrivateLink**.
     - **Description**: (Optional) Edit the description for the PrivateLink.
4. Click **Save**.

6.3.2 Deleting a PrivateLink

Use the MySQL HeatWave Console to delete a PrivateLink . If you delete a DB System with a PrivateLink, the PrivateLink is deleted automatically.

This task requires the following:

- A PrivateLink in the `Active` state.

Do the following to delete a PrivateLink.

1. In the MySQL HeatWave Console, select the **Resources** tab.
2. On the **PrivateLinks** tab, in the list of PrivateLinks, find the PrivateLink to edit, and do one of the following:
   - Click the row of the PrivateLink to highlight it, and click **Delete**.
   - Click the name of the PrivateLink to open the **PrivateLink Details** page, and click **Delete**.
3. In the **Delete PrivateLink** dialog, click **Delete PrivateLink**.

Also, you can delete any resources you created in the AWS account, such as the VPC endpoint.

6.3.3 Updating Authorized Principals for a Query PrivateLink

Use the MySQL HeatWave Console to update the authorized principals of a Query PrivateLink.

This task requires the following:

- A PrivateLink in the `Active` state.

Copyright © 2022, 2025, Oracle and/or its affiliates.

Do the following to update the authorized principals of a PrivateLink:

1. In the MySQL HeatWave Console, select the **Resources** tab.
2. On the **PrivateLinks** tab, in the list of PrivateLinks, find the PrivateLink for which you want to update the authorized principals, and do one of the following:
   - Click the row of the PrivateLink to highlight it, and click **Update Authorized Principals**.
   - Click the name of the PrivateLink to open the **PrivateLink Details** page, and click **Update Authorized Principals**.
3. In the **Configure PrivateLink** section, enter the following:
   - **ARNs of Authorized Principals**: Authorize principal ARNs to create connections to the PrivateLink. You can specify more than one ARN delimited by semicolons. You can specify either of the following:
     – (Recommended) Entire AWS accounts in the following format:

       ```
       arn:aws:iam::<ACCOUNT_ID>:root
       ```

     – Specific principals in the following format:

       ```
       arn:aws:iam::<ACCOUNT_ID>:user/<user_id>
       ```

       ```
       arn:aws:iam::<ACCOUNT_ID>:role/<role_id>
       ```

       For enhanced security, authorize a specific set of principals. In this case, the authorization to create a PrivateLink is checked twice: first inside the AWS account requesting the new endpoint, and then in MySQL HeatWave on AWS to ensure that the entity requesting the endpoint is in the set of authorized principals.
4. Click **Save**.

After you have updated the authorized principals in MySQL HeatWave on AWS, configure IAM policies in your AWS account to grant specific principals the permissions to create and delete VPC endpoints. See Configuring IAM Policies for Endpoints for a Query PrivateLink.

6.3.4 Updating Egress Endpoints for a Egress PrivateLink

You need to update the Egress Endpoints if the service in your account is listening on a different port, or you want to make it accessible to your DB System with a different DNS hostname. You may also wish to connect a different DB System. Use the MySQL HeatWave Console to update the Egress Endpoints of a Egress PrivateLink. You can also add or remove an endpoint.

This task requires the following:

- A PrivateLink in the `Active` state.
- Knowledge of the new hostname or port in your AWS account, or the new DB System ID for which you want to allow connectivity.

Do the following to update, remove, or add Egress Endpoints of a PrivateLink:

1. In the MySQL HeatWave Console, select the **Resources** tab.
2. On the **PrivateLinks** tab, in the list of PrivateLinks, find the Egress PrivateLink for which you want to update the Egress endpoints, and do one of the following:

Copyright © 2022, 2025, Oracle and/or its affiliates.

- Click the row of the Egress PrivateLink to highlight it, and under **Actions**, click **Update Egress Endpoints**.
- Click the name of the Egress PrivateLink to open the **PrivateLink Details** page, and click **Update Egress Endpoints**.

3. In the **Configure PrivateLink** section, do one of the following:
   - To update the DNS hostname or port on which the PrivateLink is accessible, or the MySQL HeatWave on AWS DB System connected to this PrivateLink:
     - Under **Source**, update the following fields:
       * **Hostname**: Update the DNS hostname for which traffic will be routed over the Egress PrivateLink. For inbound replication, this is only required if you want to support TLS certificate identity verification; leave blank otherwise.
       * **Port**: Update the port on which the service in your account is listening for traffic (for inbound replication, it is the port of the source database).
     - Under **Target DB System**: Update the DB System connected to this Egress PrivateLink.
   - To remove an endpoint: Click the X beside it.
   - To add an endpoint: Click **Add Egress Endpoint**, then fill in the information.

   > **Note**
   >
   > Currently, only a single egress endpoint is supported for each DB System. If you would like to configure more egress endpoints, create a separate egress PrivateLink for each of the endpoints.

4. Click **Save**.

> **Note**
>
> Removing and adding Egress Endpoints must be done in two separate update requests..

6.3.5 Viewing PrivateLink Details

Use the MySQL HeatWave Console to view the details of a PrivateLink.

This task requires the following:

- A PrivateLink in the `Active` state.

Do the following to view the details of a PrivateLink:

1. In the MySQL HeatWave Console, select the **Resources** tab.
2. On the **PrivateLink** tab, in the list of PrivateLinks, find the PrivateLink whose details you want to view, and do one of the following:
   - Click the row of the PrivateLink to highlight it. The details of the PrivateLink appear below the list of PrivateLinks.
   - Click the name of the PrivateLink to open the **PrivateLink Details** page.

For descriptions of PrivateLink details, see PrivateLink Details.

Copyright © 2022, 2025, Oracle and/or its affiliates.

- PrivateLink Details

6.3.5.1 PrivateLink Details

The **PrivateLink Details** page has the following tabs:

Table 6-3 Tabs on the PrivateLink Details Page

Name	Description
Details	Details of the PrivateLink. See Table 6-4.
Endpoints (for Egress PrivateLinks only)	Details of the connectivity provided by this Egress PrivateLink. See Table 6-5.
DB System (for Query PrivateLinks only)	Details of the DB System for which the Query PrivateLink provides connectivity . See Table 6-6.
Events	Details of the PrivateLink events. See Table 6-7.

Copyright © 2022, 2025, Oracle and/or its affiliates.

Table 6-4 PrivateLink Details

Field	Description
General information	• **Name**: The name of the PrivateLink. • **State**: The state of the PrivateLink: – **CREATING**: The PrivateLink is being created and is not ready yet to be used – **ACTIVE**: The PrivateLink is successfully created and is ready to be used. – **UPDATING**: The PrivateLink is in the process of being updated. For example, adding a new principal, or changing metadata. – **UPDATING_ERROR**: The PrivateLink encountered an error during `UPDATE`. It continues to function with the parameters before the update was initiated. – **DELETING**: The PrivateLink is being deleted. – **DELETING_ERROR**: The PrivateLink encountered an error during `DELETE`. It may continue to function with its current set of parameters. – **DELETED**: The PrivateLink has been deleted and is no longer available. – **FAILED**: An error condition prevented creating or updating the PrivateLink. – **NEEDS_ATTENTION**: The PrivateLink needs a user action to restore connectivity. • **Resource ID**: The unique resource identifier assigned to the PrivateLink when it is created. • **Description**: The user-defined description of the PrivateLink. • **Last updated**: The date and time the PrivateLink was last updated. If the PrivateLink has never been updated, it shows the date of time of its creation. • **Created**: The date and time the PrivateLink was created. • **Link type**: The type of PrivateLink. It is either one of: – Query – Egress

Copyright © 2022, 2025, Oracle and/or its affiliates.

Table 6-4 (Cont.) PrivateLink Details

Field	Description
Connection Information	• **Service name** **For Query PrivateLink**: The service name that you need while creating an endpoint. See Creating an Endpoint for a Query PrivateLink. **For Egress PrivateLink**: The external endpoint service name in your AWS account to which this PrivateLink connects. • **Default Hostname** (for Egress PrivateLinks only): The default DNS hostname of the Egress PrivateLink. Your DB System can use this hostname to connect to services running in your AWS account (e.g., for Inbound Replication). • **Hostname** (for Query PrivateLinks only): The hostname of the target DB System in HeatWave AWS. The hostname is a fully qualified domain name (FQDN). • **Endpoint ID** (for Egress PrivateLinks only): The ID of the Endpoint created for this Egress PrivateLink. Used to determine if an incoming connection request to your Endpoint Services should be accepted or not. See Creating a Query PrivateLink. • **ARNs of authorized principals** (for Query PrivateLinks only): The principal ARNs that are authorized to create VPC endpoints for this PrivateLink. • **Port** (for Query PrivateLinks only): The port on which the PrivateLink listens for traffic using the MySQL protocol. The default value is 3306. • **Port X** (for Query PrivateLinks only): The port on which the PrivateLink listens for clients using the MySQL X Protocol, such as MySQL Shell. The default value is 33060.

Table 6-5 Endpoints (for Egress PrivateLinks Only)

Field	Description
Name	Name of the DB System for which this egress endpoint provides connectivity.
DB System State	State of the DB System for which this egress endpoint provides connectivity.
Hostname	The DNS name that the DB System in this Egress Endpoint can use to connect to your AWS account via this Egress PrivateLink. Only needed if TLS certificate identity verification is required; blank otherwise.
Port	The port on which this egress endpoint provides connectivity.

Copyright © 2022, 2025, Oracle and/or its affiliates.

Table 6-6 DB System (for Query PrivateLinks only)

Field	Description
Name	The name of the associated DB System.
State	The state of the DB System. See MySQL DB System Details.
Description	The user-defined description of the DB System.

Table 6-7 PrivateLink Event Details

Field	Description
Type	The event and its status.
Severity	The severity level of the event, which can be one of • `INFO`: General information is provided. No special concerns • `WARNING`: Warning on an event that might impact the DB System's operation if no actions are taken. • `CRITICAL`: A critical event. An urgent action is required to avoid adverse impact on your DB System.
Message	The message body of the event report.
Created	The date and time the event report was created.

6.4 Billing for PrivateLinks

There are no surcharges for using Query PrivateLinks for Highly Available DB Systems when compared to standalone DB Systems. In contrast, when creating an Egress PrivateLink for high availability target DB System(s), you are billed for three Egress Endpoints because an Egress PrivateLink is comprised of three VPC Endpoints inside MySQL HeatWave on AWS.

Also note that when Configuring the Network Infrastructure for an Egress PrivateLink, enabling Cross-Zone Load Balancing for your Network Load Balancer may incur surcharges for your AWS account.

6.5 PrivateLink Limitations

- You can only create one Query PrivateLink and one Egress PrivateLink per DB System.
- You cannot change the value of the configuration variable, `max_connect_errors` of you DB System. Changing this configuration variable results in a loss of connectivity on the Query PrivateLink.
- When you configure a user to connect to a DB System over a PrivateLink, you cannot specify public or private IP addresses in the `host_name` field of the MySQL account names. You must use wildcard (%) in the `host_name` field. See Specifying Account Names.
- Configuring AWS cross-region endpoints is not supported.

Copyright © 2022, 2025, Oracle and/or its affiliates.

7 Importing Data

You can import data to a DB System in MySQL HeatWave on AWS. If your data is present in a MySQL instance running on-premises, in other cloud vendors as managed or unmanaged services, or another MySQL HeatWave on AWS instance, you have to export the data first. See Exporting Data.

You can import data using either of the following:

- **Data import feature**: Use the data import feature in the Console to import MySQL Shell dump and text files from an Amazon S3 bucket. See Data Import Feature.
- **Bulk ingest feature**: Connect to the DB System and use the bulk ingest feature to import text files such as CSV and TSV from an Amazon S3 bucket. This method is the fastest and most efficient in terms of computing and memory consumption. This method is ideal for importing large tables. See Bulk Ingest Feature.
- **Dump loading utility**: Use the dump loading utility of MySQL Shell to import a MySQL Shell dump manually. See Importing Data Using the Dump Loading Utility.

After importing data to a DB System in MySQL HeatWave on AWS, load data into MySQL HeatWave. See Loading or Unloading Data into MySQL HeatWave Cluster.

- Exporting Data
- Importing Data

7.1 Exporting Data

Use the dump utility of MySQL Shell to export a logical dump of the data from a MySQL instance. Connect to the MySQL instance containing the data you want to dump using MySQL Shell. You can dump all schemas in the instance, a selected schema, or selected tables and views.

Use either of the following dump utility:

- `util.dumpInstance(outputUrl[, options])`: MySQL instance export utility that exports all compatible schemas to an Amazon S3 bucket, Object Storage bucket, or to local files. By default, this utility exports users, events, routines, and triggers. See Dump Utilities.
- `util.dumpSchemas(schemas, outputUrl[, options])`: MySQL schema export utility that exports selected schemas to an Amazon S3 bucket, Object Storage bucket, or to local files.
- `util.dumpTables(schema, tables, outputUrl[, options])`: MySQL table export utility that exports selected tables of a schema to an Amazon S3 bucket, Object Storage bucket, or to local files.

The source and destination MySQL instances can be in different MySQL versions.

The MySQL Shell dump utility performs compatibility checks and transformations to ensure the data can be later successfully imported and the potential data import issues can be identified and fixed as early as possible. See MySQL Server Compatibility.

- About MySQL Shell

Copyright © 2022, 2025, Oracle and/or its affiliates.

- 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.
- Exporting Data Using MySQL Shell

7.1.1 About MySQL Shell

MySQL Shell dump and load utilities are 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 dumps all compatible schemas to an Amazon S3 bucket, OCI Object Storage, or local files. By default, this utility dumps users, events, routines, and triggers. See MySQL Shell Instance and Schema Dump Utilities.
- `dumpSchemas()`: A schema export utility that dumps selected schemas to an Amazon S3 bucket, OCI Object Storage, or local files. See MySQL Shell Instance and Schema Dump Utilities.
- `dumpTables()`: A table export utility that dumps selected tables of a schema to an Amazon S3 bucket, OCI Object Storage, or local files.
- `loadDump()`: An import utility that imports schemas to a DB System. See MySQL Shell Dump Loading Utility. To import a schema to a DB System, MySQL Shell must be installed on a machine with access to the DB System. 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.

7.1.2 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

Copyright © 2022, 2025, Oracle and/or its affiliates.

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 .

Copyright © 2022, 2025, Oracle and/or its affiliates.

7.1.3 Exporting Data Using MySQL Shell

Use MySQL Shell to export data from a source MySQL Server to an Amazon S3 buket.

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:

  ```
  util.dumpSchemas("<Schema>", {s3bucketName: "<BucketName>", dryRun: true,
  ocimds: true})
  ```

  This performs a test run of the export checking for compatibility issues. The compatibility issues and remediation steps are listed in the output. See MySQL Shell Instance and Schema Dump Utilities.

To export a schema from a source MySQL Server:

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

   ```
   mysqlsh <user>@<HostNameOfSourceMySQLServer>
   ```

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

2. Export schema to an Amazon S3 bucket:

   ```
     util.dumpInstance("<Schema>",{s3bucketName: "<BucketName>", threads:
   <ThreadSize>,
           compatibility: ["strip_restricted_grants", "strip_definers",
   "ignore_missing_pks"]})
   ```

   - `<Schema>`: The name of the schema being exported. For example, `tpch`.
   - `<BucketName>`: The Amazon S3 bucket to which the dump is to be written.
   - `<ThreadSize>`: 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 DB System.
   - `compatibility`: Specify compatibility modifiers that modify the exported data for compatibility with MySQL HeatWave on AWS. See MySQL Server Compatibility.

   See Options for S3-compatible Services.

Copyright © 2022, 2025, Oracle and/or its affiliates.

7.2 Importing Data

Use either of the following features to import data from an Amazon S3 bucket:

- **Data import**: Use the data import feature in the Console to import MySQL Shell dump and text files from Amazon S3 bucket. See Data Import Feature.
- **Bulk ingest**: Connect to the DB System and use the bulk ingest feature to import text files such as CSV and TSV from Amazon S3 bucket. This method is the fastest and most efficient in terms of computing and memory consumption. This method is suitable for importing large tables. See Bulk Ingest Feature, and Connecting to a DB System.
- **Dump loading utility**: Use the dump loading utility of MySQL Shell to import a MySQL Shell dump manually. See Importing Data Using the Dump Loading Utility.

- Data Import Feature
- Bulk Ingest Feature
- Dump Loading Utility

7.2.1 Data Import Feature

Use the data import feature in MySQL HeatWave Console to import data in a variety of formats such as MySQL dump files and text files from an Amazon S3 bucket to a DB System in MySQL HeatWave on AWS in the same region.

- Importing Sample Database
- Importing Data Using the Data Import Feature
- Viewing Data Import Details

7.2.1.1 Importing Sample Database

Oracle provides on AWS sample databases for you to test out MySQL HeatWave on AWS. Use the **Import sample data** feature in MySQL HeatWave Console to import the sample database into a DB System in any region by just a few clicks.
Currently, the following sample databases are available from Oracle.

- airportdb. See Running AirportDB Queries and Additional AirportDB Queries.
- TPCH. You have to select a size for your TPCH sample database—you are suggested to choose a value that matches the expected workload on your DB system. Please also pay attention to the **Disclaimer**. See Running tpch Queries and Sample TPCH queries.

This task requires the following:

- Access to MySQL HeatWave Console.
- An established connection from the MySQL HeatWave Console to the DB System into which you want to import data.

Follow these steps to import the sample database:

1. In the MySQL HeatWave Console, click the **Workspace** tab, and then click the **Data Imports** tab.
2. Click **Import Data**.

Copyright © 2022, 2025, Oracle and/or its affiliates.

3. In the **Import data into DB System** pane, under **Source**, select **Import sample data**. A list of sample databases appear, together with information for each of them like the name of the destination schema and the size.
4. Select a sample database. You might have to make additional selections, depending on the sample database you choose.
5. Click the **Import** button. The data import operation begins.

The **Data Imports** tab is shown, on which you can view details on your import operation.

7.2.1.2 Importing Data Using the Data Import Feature

Use the data import feature in the MySQL HeatWave Console to import data from an Amazon S3 bucket to a DB System in the same region.

This task requires the following:

- Data in an Amazon S3 bucket that you want to import. If your data is present in a MySQL instance running on-premises, in other cloud vendors as managed or unmanaged services, or another MySQL HeatWave on AWS instance, you have to export the data into an Amazon S3 bucket first.
- An IAM policy to access the Amazon S3 bucket that contains the data you want to import. See Creating an IAM Policy to Access an Amazon S3 Bucket.
- An IAM role ARN that grants the DB System access to the Amazon S3 bucket that contains the data you want to import. See Creating an IAM Role to Access an Amazon S3 Bucket.
- An established connection from the MySQL HeatWave Console to the DB System into which you want to import data.

Follow these steps to import the data:

1. In the MySQL HeatWave Console, configure the DB System into which you want to import data with the IAM role ARN. See Editing a DB System.
2. Click the **Workspace** tab, and then click the **Data Imports** tab.
3. Click the **Import Data** button.
4. In the **Import data into DB System** pane, enter the following details:
   - **Basic information**:
     - **Display name**: Specify a name for the data import operation. By default, a name is generated for you in the format of `dataImportYYYYMMDDHHMMSS`.
     - **Description**: (Optional) Specify a description for the data import operation.
   - **Source**:
     - Select **Bring your own data** (unless you want to **Import sample data** instead).
     - **S3 URI**: Specify the Amazon S3 URI. The Amazon S3 bucket and DB System must be in the same region.

Copyright © 2022, 2025, Oracle and/or its affiliates.

> **Note**
>
> You are responsible for managing the Amazon S3 bucket. Once you finish the data import, it is recommended that you review the Amazon S3 bucket's access permissions and remove access by the DB System if necessary.

- **Authentication method**: Select either one of the following authentication methods by clicking its radio button:
  - **IAM role**: Use the displayed data import role ARN, which was specified while creating or editing the DB System. See Creating a DB System and Editing a DB System.
  - **User access key**: Enter the **Access key ID** and **Secret access key**. See Managing Access Keys.
- **File type**: Select either of the following file types:
  - **MySQL dump files**: Specify the **Import settings**:
    - **Character set**: Enter the character set of the import data. The default character set is `utf8mb4`. See Character Sets and Collations in MySQL.
    - **Update GTID set**: For data that contain GTIDs, to enable GTID-based replication, apply the `gtid_executed` GTID set from the source MySQL instance, as recorded in the dump metadata, to the `gtid_purged` GTID set on the target MySQL instance using one of the following options:
      - **OFF**: (Default) Do not update the GTID set. This is also the option for data that do not contain GTIDs.
      - **APPEND**: Append the `gtid_executed` GTID set from the source MySQL instance to the `gtid_purged` GTID set on the target MySQL instance.
      - **REPLACE**: Replace the `gtid_purged` GTID set on the target MySQL instance with the `gtid_executed` GTID set from the source MySQL instance.

      See the description for `gtid_purged` for more information.
    - **Maximum bytes per transaction**: The maximum number of bytes that can be loaded from a data file with a single `LOAD DATA` statement. If a data file's size exceeds this value, multiple `LOAD DATA` statements load data from the file in chunks. This setting is required when a data file is too large for the target server's limits, such as the limits defined by the server variables `group_replication_transaction_size_limit` or `max_binlog_cache_size` . For example, if you receive the error `MySQL Error 1197 (HY000): Multi-statement transaction required more than 'max_binlog_cache_size' bytes of storage` when importing a file, set this number to a value no greater than the server instance's `max_binlog_cache_size` . For details, see the description of `maxBytesPerTransaction` in Dump Loading Utility.
  - **Text files**:
    - **Names of data files in S3 bucket**: Specify the names of data files in the Amazon S3 bucket. Apart from text files, you can specify text files in compressed formats such as gzip (`.gz`) and zstd (`.zst`). You can specify ranges of files using wildcard pattern matching. To match any single

Copyright © 2022, 2025, Oracle and/or its affiliates.

character, use ?, and to match any sequence of characters, use *. For example, `data_c?`, and `/backup/replica/2021/*.tsv`.

* **File parsing settings**:
  * **Character set**: Enter the character set to import data to the target MySQL instance. It must correspond to the character set given in the dump metadata that was used when the MySQL dump was created by MySQL Shell instance dump utility, schema dump utility, or table dump utility. The character set must be permitted by the `character_set_client` system variable and supported by the MySQL instance. The default character set is `utf8mb4`. See Character Sets and Collations in MySQL.
  * **Dialect**: Select the dialect of the imported data file. You can select **CSV (Unix)**, **CSV**, or **TSV**. The default dialect is **CSV (Unix)**. The dialect selects the default values of the following parsing settings: **Fields terminated by**, **Fields enclosed by**, **Fields escaped by** and **Fields optionally enclosed**. To view the default settings per dialect, see Dialect settings.
  * **Skip rows**: Specify the number of rows that is skipped from the beginning of the imported data file or, in the case of multiple import files, at the beginning of every file included in the file list. By default, no rows are skipped.
  * **Lines terminated by**: Specify one or more characters (or an empty string) with which each of the lines are terminated in the imported data file. For example, `\r\n`. The default is as per the specified dialect.
  * **Fields terminated by**: Specify one or more characters (or an empty string) with which each of the fields are terminated in the imported data file. For example, `\t`. The default is as per the specified dialect.
  * **Fields enclosed by**: Specify a single character (or an empty string) with which the utility encloses each of the fields in the imported data file. For example, ". The default is as per the specified dialect.
  * **Fields escaped by**: Specify the character that is to begin escape sequences in the imported data file. For example, \. The default is as per the specified dialect.
  * **Fields optionally enclosed**: Select the option to enclose a field only if it has a string data type such as `CHAR`, `BINARY`, `TEXT`, or `ENUM`, and deselect the option to enclose all of the fields in the imported data file.
* **Destination**: Select the **Schema** and the **Table** in the DB System to which you want to import data.

– Click **Import**. The data import operation begins.

The **Data Imports** tab is shown, on which you can view details on your import operation..

7.2.1.3 Viewing Data Import Details

To view data import details:

1. In the MySQL HeatWave Console, select the **Workspace** tab.
2. Select the **Data Imports** tab. In the list of data imports, select the import for which you want to view the details. The **Data Import Details** pane shows its information.

Copyright © 2022, 2025, Oracle and/or its affiliates.

3. If your selected import is still in progress and you want to cancel it, click the **Cancel** button on top of the import list.

- Data Import Details

7.2.1.3.1 Data Import Details

Use the Console to view the **Data Import Details**. For more information on the details being displayed, see Importing Data Using the Data Import Feature.

Table 7-1 Tabs on the Data Import Details Page

Name	Description
Details	Data import details. See Table 7-2.
Destination	Data import destination. See Table 7-3.
Messages	Data import messages. See Table 7-4.

Table 7-2 Data Import Details

Field	Description
General information	• **Name**: The name of the data import. • **State**: The state of the data import operation: – **In Progress**: The data import is in progress – **Succeeded**: The data import has succeeded – **Failed**: The data import has failed • **Resource ID**: The unique resource identifier assigned to the data import. • **Description**: The user-defined description of the PrivateLink. • **Progress (%)**: The progress of the data import in percentage. • **Last updated**: The date and time when the **Data Import Details** were last updated. • **Created**: The date and time when the data import operation started.

Copyright © 2022, 2025, Oracle and/or its affiliates.

Table 7-2 (Cont.) Data Import Details

Field	Description
Source	Information on the data import source: • **S3 URI**: The Amazon S3 URI of the S3 bucket containing the data being imported. • **Character set**: The character set for the import data. • **Update GTID set** (for MySQL dump file imports only): For data that contain GTIDs, how the `gtid_executed` GTID set from the source MySQL instance is applied to the `gtid_purged` GTID set on the target MySQL instance. – **OFF**: (Default) Do not update the GTID set. This is also the option for data that do not contain GTIDs. – **APPEND**: Append the `gtid_executed` GTID set from the source MySQL instance to the `gtid_purged` GTID set on the target MySQL instance. – **REPLACE**: Replace the `gtid_purged` GTID set on the target MySQL instance with the `gtid_executed` GTID set from the source MySQL instance. • **Maximum bytes per transaction** (for MySQL dump file imports only): The maximum number of bytes that can be loaded from a data file with a single `LOAD DATA` statement. • **Files** (for text file imports only): The names of imported data files. • **Skip rows** (for text file imports only): The number of rows that is skipped from the beginning of the imported data file or, in the case of multiple import files, at the beginning of every file included in the file list. • **Lines terminated by** (for text file imports only): The character(s) with which each of the lines are terminated in the imported data file. • **Fields terminated by** (for text file imports only): The character(s) with which each of the fields are terminated in the imported data file. • **Fields enclosed by** (for text file imports only)The character with which the utility encloses each of the fields in the imported data file. • **Fields escaped by** (for text file imports only): The character that is to begin escape sequences in the imported data file. • **Fields optionally enclosed** (for text file imports only): Whether a field is enclosed only if it has a string data type.

Copyright © 2022, 2025, Oracle and/or its affiliates.

Table 7-3 Data Import Destination Details

Field	Description
Schema/Table	The tables being imported, listed under the schemas they are under.
Progress	The progress on the table import.
Size	The size of the table.

Table 7-4 Data Import Messages

Field	Description
Errors	Codes for any errors in the data import operation.
Warnings	Warnings on the import operation that require user attention.
Info	Information on the data import.

7.2.2 Bulk Ingest Feature

Use the bulk ingest feature to import text files in bulk from an Amazon S3 bucket to a DB System in MySQL HeatWave on AWS in the same region. Bulk import is ideal for importing large tables.

This task requires the following:

- Access to AWS Management Console.
- Access to MySQL HeatWave Console.
- A client that is connected to the DB system. See Connecting to a DB System.

Do the following to bulk ingest data to a DB System:

1. Create an IAM policy to access the Amazon S3 bucket. See Creating an IAM Policy to Access an Amazon S3 Bucket.
2. Create an IAM role that grants the DB System access to read data from the desired Amazon S3 bucket. See Creating an IAM Role to Access an Amazon S3 Bucket.
3. (Optional) If a MySQL user (and not a DB System administrator) is importing data using the bulk ingest feature, grant the `LOAD_FROM_S3` privilege. See Granting Privileges to Bulk Ingest Data From Amazon S3.
4. Create a new DB System or edit an existing DB System, and enter the data import role ARN details. See Creating a DB System and Editing a DB System.
5. Bulk ingest data from an Amazon S3 bucket. See Importing Data Using the Bulk Ingest Feature.

- Granting Privileges to Bulk Ingest Data From Amazon S3
  By default, a DB System administrator has the `LOAD_FROM_S3` privilege. However, if a user is importing data using the bulk ingest feature, the `LOAD_FROM_S3` privilege is required.
- Importing Data Using the Bulk Ingest Feature
- Bulk Ingest Limitations

Copyright © 2022, 2025, Oracle and/or its affiliates.

7.2.2.1 Granting Privileges to Bulk Ingest Data From Amazon S3

By default, a DB System administrator has the `LOAD_FROM_S3` privilege. However, if a user is importing data using the bulk ingest feature, the `LOAD_FROM_S3` privilege is required.

This task requires the following:

- A running DB System. See Creating a DB System.

Do the following to grant the `LOAD_FROM_S3` privilege to a user:

1. Connect to the DB System. See Connecting to a DB System.
2. Grant the `LOAD_FROM_S3` privilege to the user:

```
GRANT LOAD_FROM_S3 ON *.* TO <User>
```

7.2.2.2 Importing Data Using the Bulk Ingest Feature

Connect to a DB System and bulk ingest data from an Amazon S3 bucket to a DB System in the same region.

This task requires the following:

- A client that is connected to the DB System. See Creating a DB System.
- Properly configured data import role ARN in the DB System. See Creating an IAM Role to Access an Amazon S3 Bucket, Creating a DB System, and Editing a DB System.
- Data, which you want to bulk ingest, in an Amazon S3 bucket. See Exporting Data.

Do the following to bulk ingest data from an Amazon S3 bucket:

1. Connect to a DB System. See Connecting to a DB System.
2. Create a new database or use an existing database to bulk ingest data. For example, the following query creates a new database, `airportdb`, and selects the database for further actions:

```
CREATE DATABASE airportdb;
USE airportdb;
```

3. Use an existing empty table or create a new table. For example, the following query creates a new table, `booking`:

```
CREATE TABLE IF NOT EXISTS `booking`
   (`booking_id` int NOT NULL AUTO_INCREMENT,
    `flight_id` int NOT NULL,
    `seat` char(4) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci
DEFAULT NULL,
    `passenger_id` int NOT NULL,
    `price` decimal(10,2) NOT NULL,
     PRIMARY KEY (`booking_id`)) ENGINE=InnoDB AUTO_INCREMENT=55099799
DEFAULT CHARSET=utf8mb4
        COLLATE=utf8mb4_unicode_ci COMMENT='Verifying bulk ingest';
```


Copyright © 2022, 2025, Oracle and/or its affiliates.

4. Bulk ingest data from an Amazon S3 bucket into the table you created using the `LOAD DATA` query with the `BULK` algorithm:

```
LOAD DATA FROM S3 '<Amazon-S3-URL>' INTO TABLE <Table-name>
        COLUMNS TERMINATED BY '\t' LINES TERMINATED BY '\n' ALGORITHM=BULK;
```

   - `<Amazon-S3-URL>`: Specify the Amazon S3 bucket URL present in the same region as the DB System in the following format:

     ```
     s3-<Region>://<Bucketname>/<FilenameOrPrefix>
     ```

     - `<Region>`: The AWS region that contains the Amazon S3 bucket to load.
     - `<Bucketname>`: The name of the Amazon S3 bucket that contains the data to load.
     - `<FilenameOrPrefix>`: The filename or prefix of one or more text files to load.
   - `<Table-name>`: Specify an already existing empty table or create a new table.
   - Specify the column and line terminators used by the in the data files you are loading. The statement above assumes the column terminator is `\t` and the line terminator is `\n`.

   It is recommended to split the data into multiple files to improve the performance of bulk ingest. For example, the following query loads data from an Amazon S3 bucket that is split across 25 files. The files are named, `booking.tsv1`, `booking.tsv2`, ..., `booking.tsv25`:

```
LOAD DATA FROM S3 's3-us-east-1://mysql-heatwave-data-us-east-1/airportdb/
booking.tsv.'
        COUNT 25 IN PRIMARY KEY ORDER INTO TABLE booking
        COLUMNS TERMINATED BY '\t' LINES TERMINATED BY '\n' ALGORITHM=BULK;
```

   See Bulk Ingest Data to MySQL Server.

7.2.2.3 Bulk Ingest Limitations

The following are some limitations to the Bulk Ingest Feature of MySQL HeatWave on AWS when using the `LOAD DATA FROM S3` statement with `ALGORITHM=BULK`:

- The target table must be empty. The state of the table should be as though it has been freshly created. If the table has instantly added or dropped column, run the `TRUNCATE TABLE` statement before running `LOAD DATA FROM S3` with `ALGORITHM=BULK`.
- The target table must not be partitioned.
- The target table must not contain secondary indexes.
- The target table must be in a file-per-table tablespace—it must not be in a shared tablespace.
- The target table must have the default row format `ROW_FORMAT=DYNAMIC`. If needed, use `ALTER TABLE` to change the table's row format after running `LOAD DATA FROM S3` with `ALGORITHM=BULK`.
- The target table must contain a primary key.
- Prefix indexing for the primary key is not supported.
- The target table must not contain virtual or stored generated columns.

Copyright © 2022, 2025, Oracle and/or its affiliates.

- The target table must not contain foreign keys.
- The target table must not contain `CHECK` constraints.
- The target table must not contain triggers.
- The target table must not be replicated to other nodes.
- The target table must not use a secondary engine. If needed, set the secondary engine after running `LOAD DATA FROM S3` with `ALGORITHM=BULK`.
- The `INFILE` and `URL` clauses are not supported for `LOAD DATA FROM S3` with `ALGORITHM=BULK`.
- `LOAD DATA FROM S3` with `ALGORITHM=BULK` locks the target table exclusively and does not allow other operations on the table when the query is running.
- Automatic rounding or truncation of the input data is not supported. `LOAD DATA FROM S3` with `ALGORITHM=BULK` will fail if the input data requires rounding or truncation in order to be loaded.
- The target table cannot be a temporary table.
- `LOAD DATA FROM S3` with `ALGORITHM=BULK` is atomic but not transactional. It commits any transaction that is already running. On failure, the `LOAD DATA FROM S3` statement is completely rolled back.
- `LOAD DATA FROM S3` with `ALGORITHM=BULK` cannot be executed when the target table is explicitly locked by a `LOCK TABLES` statement.

7.2.3 Dump Loading Utility

The dump loading utlity is an import utility that imports schemas to a DB System. See MySQL Shell Dump Loading Utility. To import a schema to a DB System, MySQL Shell must be installed on a machine with access to the DB System.

- Importing Data Using the Dump Loading Utility

7.2.3.1 Importing Data Using the Dump Loading Utility

Use the dump loading utility of MySQL Shell to load a dump from an Amazon S3 bucket to a DB System in MySQL HeatWave on AWS.

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 Using 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 are no issues when the dump files are loaded from an Amazon S3 bucket into the connected DB System:

```
util.loadDump("<Schema>", {s3bucketName: "<BucketName>", dryRun: true,
ocimds: true})
```


Copyright © 2022, 2025, Oracle and/or its affiliates.

This performs a test run of the import checking for compatibility issues. The compatibility issues and remediation steps are listed in the output. See MySQL Shell Instance and Schema Dump Utilities.

Do the following to import data using the dump loading utility:

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

   ```
   mysqlsh <Username>@<HostNameOfMySQLDBSystem>
   ```

   See Connecting with MySQL Shell .

2. Import data using the dump loading utility:

   ```
   util.loadDump("<Prefix>",{s3BucketName: "<BucketName>", threads:
   <ThreadSize>})
   ```

   - `<Prefix>`: (Optional) Adds a prefix to the files uploaded to the bucket. If you specify this option, the files are uploaded to the defined bucket with the prefix in the following format: `<BucketPrefix>/filename`, similarly to a file path. For example, if `<BucketPrefix>` is set to `test`, every file uploaded to the defined bucket, `<BucketName>`, is done so as `test/<filename>`. If you download the file, the prefix is treated as a folder in the download. For local exports, this parameter is the path to the local directory you want to export to.
   - `<BucketName>`: The an Amazon S3 bucket from which the dump is to be loaded.
   - `<ThreadSize>`: 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 DB System.

Copyright © 2022, 2025, Oracle and/or its affiliates.

8 Inbound Replication

This chapter covers how to create and manage **Channels** for implementing inbound replication.

- About Inbound Replication
- Configuring Inbound Replication
- Source Configuration
- Creating a Replication User On a Source Server
- Creating a Channel
- Managing Replication Channels
- Viewing Channel Details
- Limitations

8.1 About Inbound Replication

Inbound replication enables asynchronous replication from a MySQL-based source (an on-premises or other cloud MySQL Instance, or another MySQL HeatWave on AWS DB System) to a MySQL HeatWave on AWS DB System. It uses a replication channel configured in MySQL HeatWave on AWS to copy transactions from the source to a MySQL HeatWave on AWS DB System (replica). The channel connects the source to the replica, and copies data from the source to the replica.

Asynchronous replication means that the replica does not need to be running and connected to the source all the time. It can pick up new updates whenever it is online and reconnects.

A replica DB System can connect to only one MySQL source. The source server does not control the DB System replica and does not need to have permission to write to it. The replica uses the configured replication channel to connect to the source using a set of replication user credentials. Over the connection formed by the channel, the replica retrieves committed transactions from the source. The replica then writes those transactions to its own copy of the databases.

> **Note**
>
> Inbound replication is not a managed functionality. You are responsible for configuring and maintaining the channel, and for ensuring that the traffic between source and replica is properly configured.

Impact of the DB System Operations on the Replication Channel

The DB System operations affect the replication channel:

- If you stop the replica DB System, it stops any enabled Channels and changes its state to `Needs Attention`.
- If you delete the replica DB System, it deletes any Channels attached to it.

Copyright © 2022, 2025, Oracle and/or its affiliates.

- If a restart, upgrade, or update is performed on a replica DB System, or if a failover happens in a replica that is a high availability DB System, the Channel is temporarily suspended and in the `Needs Attention` state while any of the aforementioned operations is taking place. When the operation is completed, replication is resumed automatically.
- If a switchover happens in a replica that is a high availability DB System, the Channel is temporarily suspended and in the `Inactive` state. When the operation is completed, replication is resumed automatically.

8.2 Configuring Inbound Replication

The following sections explain how to configure inbound replication over different kinds of connections

- Configuring Inbound Replication over the Public Internet
- Configuring Inbound Replication over an Egress PrivateLink

8.2.1 Configuring Inbound Replication over the Public Internet

Follow these steps to configure inbound replication over the public internet:

1. Configure your replication source. See Source Configuration.
2. Add a replication user on the source. See Creating a Replication User On a Source Server.
3. Create a replication channel over the public internet. See Creating a Channel.
4. Enable your replication channel. See Disabling or Enabling a Channel.

8.2.2 Configuring Inbound Replication over an Egress PrivateLink

Follow these steps to configure inbound replication over an Egress PrivateLink:

1. Configure your replication source. See Source Configuration.
2. Create a replication user on the source. See Creating a Replication User On a Source Server.
3. Configure the surrounding network infrastructure on AWS. See Configuring the Network Infrastructure for an Egress PrivateLink.
4. Create an Egress PrivateLink in MySQL HeatWave on AWS and associate it with the VPC Endpoint Service and Egress Endpoints for your source database. See Creating an Egress PrivateLink.
5. Create a replication channel with your Egress PrivateLink. See Creating a Channel.
6. Enable your replication channel (if it is not enabled yet). See Disabling or Enabling a Channel.

8.3 Source Configuration

To use inbound replication in MySQL HeatWave on AWS, the source and its network connection to the replica DB System must have the following configurations.

- The minimum supported version of MySQL for a replication source is 5.7.9.
- The source cannot be running a higher version of MySQL than the DB System. The replica must be running the same version as, or a higher version of MySQL than, the source.

Copyright © 2022, 2025, Oracle and/or its affiliates.

- If you intend to encrypt the communication between source and replica, the source must be configured to use SSL. See Server-side Configuration for Encrypted Connections.
- The source and the DB System must run with the same `lower_case_table_names` system variable value. See System Initialization Variables
- The source must have binary logging enabled, with the system variable `log_bin=ON`.
- The source must use row-based binary logging, with the system variable `binlog_format=ROW`.
- The source's binary log has an extended retention period, so that the target database inside MySQL HeatWave on AWS can apply all the transactions before the log is flushed. For example, run the following command on a MySQL client as an administrator to set retention period to be 24 hours:

  ```
  call mysql.rds_set_configuration('binlog retention hours', 24);
  ```

- The source has automated backups enabled.
- *For using GTID auto-positioning*: The source must use GTIDs, with the system variables `gtid_mode=ON` and `enforce_gtid_consistency=ON`.

> **Note**
>
> GTID auto-positioning must be used for replication to a high availability DB System.

- The replication user must be present on the source server with the required privileges. See Creating a Replication User On a Source Server.
- The network must be configured to permit replication traffic between the DB System replica and the source server.

8.4 Creating a Replication User On a Source Server

The replication user for inbound replication must be present on the source server with the required privileges. The replica uses this user account when it connects to the source server.

This task requires:

- MySQL Shell or MySQL client.

Do the following to add a replication user to the source server:

1. Open MySQL Shell and connect to the MySQL source server.
2. Run the following SQL command to create a replication user, and to permit only encrypted connections for all accounts named by the statement. In this example, the username for the replication user is `rpluser001`:

   ```
   CREATE USER rpluser001 IDENTIFIED BY '<password>' REQUIRE SSL;
   ```


Copyright © 2022, 2025, Oracle and/or its affiliates.

> **Note**
>
> Inbound replication in MySQL HeatWave on AWS requires MySQL source password to contain 8 to 32 characters including at least one number, one uppercase letter, one lowercase letter, and one character from `,.-+*;:_!#%&/()=?><`

> **Note**
>
> When replicating from a source of MySQL 8.0.x or earlier (which supports the `mysql_native_password` authentication method) to a DB system on MySQL HeatWave on AWS running MySQL 9.y.z (which does not support the `mysql_native_password` authentication method), make sure the replication user on the source does NOT use the `mysql_native_password` authentication, or connections to the source will fail. For example, if your source MySQL 8.0.x Server uses the `mysql_native_password` authentication by default, override it with an authentication method supported by the replica when creating the replication user:
>
> ```
> CREATE USER rpluser001@'%' IDENTIFIED WITH 'caching_sha2_password'
> BY 'password' REQUIRE SSL;
> ```

3. Run the following command to grant the `REPLICATION SLAVE` privilege to the new replication user, `rpluser001` in this example:

   ```
   GRANT REPLICATION SLAVE on *.* to rpluser001;
   ```

The replication user is created and granted the required privileges. Note the user name and password so that you can specify them when you create the replication channel..

8.5 Creating a Channel

> **Note**
>
> In MySQL replication, a **replica** is a MySQL server that receives changes from another MySQL server (**source**) and applies the changes. In MySQL HeatWave on AWS, a Channel references the replica as the **target** DB System.

Use the MySQL HeatWave Console to create a channel. A channel connects the source (an on-premises or other cloud MySQL Instance, or other MySQL HeatWave on AWS DB System) to a target MySQL HeatWave on AWS DB System (replica), and copies data changes from the source to the target. There are two types of replication channels:

- Channels over public internet
- Channels over Egress PrivateLinks

This task requires the following:

Copyright © 2022, 2025, Oracle and/or its affiliates.

- A MySQL source server that meets the requirements in Source Configuration. You need to know both the hostname of the source and the IP address at which the source can be reached.
- A replication user on the MySQL source server. See Creating a Replication User On a Source Server. You need to know the replication user's credentials.

> **Note**
>
> To successfully execute inbound replication from Amazon RDS databases, ensure that all user accounts on your source server are migrated away from the default `mysql_native_password` authentication method before creating the replication channel, because any operation that involves the `mysql_native_password` authentication method (for example, creating or changing the password of a user account that uses the `mysql_native_password` authentication method) will break the replication.

- An active DB System as the target of the Channel. If the source server contains data, import existing data into the DB System:
  - Export the data from the MySQL source server (see Exporting Data Using MySQL Shell for instructions) and:
    - *If using GTID-based replication:* Capture the value of the source server's system variable `gtid_executed`, from which replication should begin. The value is also available as metadata in the `gtidExecuted` field in the `@.json` dump file created by MySQL Shell's dump utilities.

      > **Note**
      >
      > GTID-based replication must be used when replicating to a high availability DB System.

    - *If not using GTID-based replication:* Capture the source server's binary log coordinates (the log file name and the log position), from which replication should begin. See Obtaining the Replication Source Binary Log Coordinates on how to do that.
  - Import the data into the target DB System (see Importing Data Using the Data Import Feature for instructions), and then:
    - *If using GTID-based replication:* apply the source's `gtid_executed` GTID set to the target's `gtid_purged` GTID set (see Update GTID set on how to do that).

      > **Note**
      >
      > GTID-based replication must be used when replicating to a high availability DB System.

    - *If not using GTID-based replication:* supply the binary log coordinates to the Replication Positioning settings when you create or edit a replication channel on the MySQL HeatWave Console.
- If you want to use a PrivateLink for the replication, a properly configured Egress PrivateLink. See Deploying an Egress PrivateLink.

Copyright © 2022, 2025, Oracle and/or its affiliates.

Do the following to create a replication channel:

1. In the MySQL HeatWave Console, select the **Resources** tab, and then click **Channels** to open the **Channels** tab.
2. On the **Channels** tab, click **Create Channel** to open the **Create Channel** dialog.
3. In the **Create Channel** dialog, enter the following information:
   - **Basic information**:
     - **Display name**: Specify a display name for the channel. If you do not specify a name, one is generated for you in the format, `channelyyyyMMddHHmmssSSS`.
     - **Description**: (Optional) Specify a description of the channel and its purpose.
     - **Enabled automatically upon creation**: Specify whether the channel should start automatically after it is successfully created. By default, it is enabled. If you disable this option, the Channel will be in Inactive state and will not start replication. You have to enable the channel manually after the channel is created to start replication.
   - **Target DB system:**
     - **Select DB system**: Click on the button to select the DB System to use as the replication target.
   - **Source connection**: Configure the MySQL source from where you want to replicate the data
     - Choose the type of connection:
       * **Channel over public internet**: Choose if your source is to be accessed through the public internet, and then configure the following parameters (*this is the only option if you are replicating to a high availability DB System*):
         * **Hostname**: Specify the hostname of the MySQL source. You can either specify an IP address or a fully qualified domain name. For example, 112.123.121.12, or `server.yourdomainname.com`.
         * **Port**: (Optional) Specify the port number (between 1024 and 65535) the MySQL source listens on. The default value is 3306.
         * **Allowed Outbound addresses**: Specify the IP addresses (in the CIDR format) of the MySQL source from which the DB System replicates the data, to allow outbound connections from the target to the source. If you want to specify more than one IP address ranges, use a semicolon. For example, 10.2.3.1/32;10.3.0.0/16.
       * **Channel over PrivateLink**: Choose if you are creating a channel over an existing Egress PrivateLink, and then configure the following parameters (*this is not supported if you are replicating to a high availability DB System*):
         * **Egress PrivateLink**: Choose from the drop-down menu the Egress PrivateLink you have defined.
         * **Hostname**: Specify the **Hostname** of one of the **Endpoints** of your Egress PrivateLink. If you have left the **Hostname** of the **Endpoint** blank, then specify the **Default hostname** of your Egress PrivateLink. See Viewing PrivateLink Details on how to find the information.

Copyright © 2022, 2025, Oracle and/or its affiliates.

> **Note**
>
> If you use the **Default hostname** here, the **Verify identity** SSL mode is not supported.

  - **Port**: (Optional) Specify the port number (between 1024 and 65535) the MySQL source listens on. The default value is 3306.

- **Username**: Specify the replication username for the account that you created on the MySQL source server. The channel uses these credentials to connect to the source. See Creating a Replication User On a Source Server.
- **Password**: Specify the password for the replication user account.

> **Note**
>
> Inbound replication in MySQL HeatWave on AWS requires MySQL source password to contain 8 to 32 characters including at least one number, one uppercase letter, one lowercase letter, and one character from `,.-+*;:_!#%&/()=?><`
> You have to change the password on the MySQL source if it does not meet these requirements.

- **SSL mode**: Select the required SSL mode. The selected mode is used to populate the SSL-specific values of the connection to the MySQL Source. Select either of the following SSL modes:
  - **Disabled**: Establishes an unencrypted connection between the source and target.
  - **Required**: (Default) If the server supports encrypted connection, establishes an encrypted connection. The connection attempt fails if an encrypted connection cannot be established.
  - **Verify certificate authority**: Like the **Required** mode, establishes an encrypted connection if the server supports encrypted connections and additionally verifies the Certificate Authority certificate configured on the source against the Certificate Authority's X509 certificate (PEM). You have to upload your Certificate Authority's X509 certificate.
  - **Verify identity**: Like the **Verify certificate authority** mode, establishes an encrypted connection, verifies the Certificate Authority's certificate, and additionally verifies the source hostname, which you define in the source SSL certificate, against the hostname that you define in the **Hostname** field. You have to upload your Certificate Authority's X509 certificate.
  - **Certificate authority's X509 certificate (PEM)**: Enables you to upload the source Certificate Authority's X509 certificate in PEM format. It is displayed only when you select the **Verify certificate authority** or **Verify identity** mode. The certificate is used to verify the Certificate Authority's certificate on the source.

- **Replication positioning**: Configure the Source GTID settings:
  - **Use GTID auto-positioning** : Select this option when the system variable, `gtid_mode`, is set to `ON` on the source. It means the source server can provide the replica with GTID information for auto-positioning. For more information, see

Copyright © 2022, 2025, Oracle and/or its affiliates.

GTIDs. *This option is recommended for replicating to a standalone DB System and mandatory for replicating to a high availability DB System*.

- **Do not use GTID auto-positioning** : Select this option when the system variable, `gtid_mode`, is set to `OFF`,`OFF_PERMISSIVE`, or `ON_PERMISSIVE` on the source. It means the source server cannot provide the replica with GTID information for auto-positioning. *This option is not available for replicating to a high availability DB System*.

  Select one of the following options to convert a transaction that is anonymous to one that has GTIDs:

  * **Manually specify a UUID**: There are two options:
    * Define your own UUID by typing it into the **UUID** field.
    * Accept the generated UUID shown in the **UUID** field, or generate a new one by clicking the regenerate button.

    See GTID Format and Storage for more information on UUIDs.
  * **Same UUID as target DB system**: Select this option to use the same UUID as the target DB System.

  Also specify the binary log coordinates (see Obtaining the Replication Source Binary Log Coordinates on how to obtain them) when not using GTID auto-positioning:

  * **Binary log file name**: Specify the name of the binary log file, which contains events that describe database changes on the source. See The Binary Log.
  * **Binary log offset**: Specify the binary log offset within the binary log file from which the replica should start reading the source's binary log.

- **Replication details**: Configure details for the replication.
  - **Channel name**: (Optional) Specify the channel name. The target DB System uses this replication channel to communicate with the MySQL source. If you do not specify a name, `replication_channel` is used.
  - **Applier username**: (Optional) Specify the username of the replication applier on the target DB System. If you do not specify a username, the credentials of the DB System administrator is used.
  - **Replication delay**: Set the amount of time, in seconds, that the channel waits before applying a transaction received from the source. The default value is 0.
- **Advanced options**:
  - **Tables without primary key**: Specify how a replication channel handles the creation and alteration of tables with no primary keys on the source.
    * **Raise an error** : Raise an error when replicating a `CREATE TABLE` or `ALTER TABLE` transaction with no primary keys.
    * **Allow**: (Default) Allow replicating a `CREATE TABLE` or `ALTER TABLE` transaction with no primary keys.
    * **Generate primary key**: Allow replicating a `CREATE TABLE` or `ALTER TABLE` transaction with no primary keys, and automatically generate a new invisible primary key column while creating a table without primary keys on the target.
  - **Channel filter**: (Optional) Click **Channel filter options** to configure the type and value of replication filters for the channel. See Channel Filter Rules for Inbound Replication for details.

Copyright © 2022, 2025, Oracle and/or its affiliates.

    - **Select a filter templates**: You can select a predefined filter template that populate the **Type** and **Value** from the available list of filters that matches the source, or click **Add new filter** to create a custom filter.
    - **Select a filter type**: You can select a variety of filter types such as ignore a database or table and rewrite a database.
    - **Enter a filter value**: Enter a value required for your selected filter type.
    - Click **Add new filter** to add another filter.
- Click **Create**.

The state of the channel changes to **Creating**, and when the create is complete, the status changes to one of the following:

- **Inactive** if **Enabled automatically upon creation** is disabled.
- **Active** if the channel is enabled successfully.
- **Needs Attention** if the channel encounters any error while enabling it. The replication cannot be started.

See the next section on the channel filter rules for configuring replication filters.

- Channel Filter Rules for Inbound Replication

8.5.1 Channel Filter Rules for Inbound Replication

Each filter rule can contain only one filter type and value. However, you can add multiple filter rules and the result is the logical addition of all the rules. For example, if you create a filter rule with type, `REPLICATE_DO_DB`, and filter value, `mysql1`, and another filter rule, with type, `REPLICATE_DO_DB`, and filter value, `mysql2`, then both `mysql1` and `mysql2` databases are replicated.

Table 8-1 Channel Filter Rules

Filter Type	Filter Value	Details
`REPLICATE_DO_DB`	`database`	Replicates the specified database.
`REPLICATE_IGNORE_DB`	`database`	Restricts the replication of the specified database.
`REPLICATE_DO_TABLE`	`database.table`	Replicates the specified table in the database.
`REPLICATE_IGNORE_TABLE`	`database.table`	Restricts the replication of the specified table in the database.
`REPLICATE_WILD_DO_TABLE`	`database.table`	Replicates the table that matches the specified wildcard pattern.
`REPLICATE_WILD_IGNORE_TABLE`	`database.table`	Restricts the replication of any table that matches the specified wildcard pattern.
`REPLICATE_REWRITE_DB`	`from_database->to_database`	Translates `from_database` on the source to `to_database`.

Wildcard Patterns: `REPLICATE_WILD_DO_TABLE` and `REPLICATE_WILD_IGNORE_TABLE` support wildcards. Wildcard patterns can contain the % and _ wildcard characters. % represents 0 or more characters and _ represents any single character. For example:

Copyright © 2022, 2025, Oracle and/or its affiliates.

- A filter value of `db%.table%` in filter type `REPLICATE_WILD_DO_TABLE` replicates only those tables where the database name starts with `db` and the table name starts with `table`.
- A filter value of `db%.%` in filter type `REPLICATE_WILD_DO_TABLE` replicates all tables where the database name starts with `db%`.
- A filter value of `db_.table_` in filter type `REPLICATE_WILD_IGNORE_TABLE` ignores those tables where the database name contains three characters and starts with `db` and the table name contains six character and starts with `table`.

If you want to use the two wildcard characters literally in the database or table names in your rule, escape them with backslashes (\). For example, `my\_db.top\_90\%`.

Mandatory Filter Rules for replicating from MySQL databases running on Amazon RDS:

> **Warning**
>
> Your inbound replication will fail with errors unless these filter rules are in place.

> **Note**
>
> These rules can be selected using the per-configured filter templates. See Creating a Channel for details.

Table 8-2 Mandatory Filter Rules for replicating from MySQL databases running on Amazon RDS

Filter Type	Filter Value
REPLICATE_IGNORE_TABLE	mysql.rds_heartbeat2
REPLICATE_IGNORE_TABLE	mysql.rds_sysinfo

8.6 Managing Replication Channels

This section describes how to manage Channels using the MySQL HeatWave Console.

- Disabling or Enabling a Channel
- Editing a Channel
- Resuming a Channel
- Resetting a Channel
- Deleting a Channel

8.6.1 Disabling or Enabling a Channel

Disabling a channel pauses the replication and **enabling** a channel resumes the replication.

Copyright © 2022, 2025, Oracle and/or its affiliates.

> **Note**
>
> If the source server contains existing data, it is necessary to copy over this data to the target server before enabling the channel and starting replication for the first time. For more information, see **Choosing a Method for Data Snapshots** and Importing Data.

To disable or enable a Channel:

1. In the MySQL HeatWave Console, select the **Resources** tab, and then click **Channels** to open the **Channels** tab.
2. In the list of channels, find the channel you want to disable or enable.
3. To disable an active channel, do one of the following:
   - Select the row of the channel to highlight it, and click the **Disable** button.
   - Click the name of the channel to open the **Channel Details** page. Click the **Disable** button.

   The **Disable MySQL Channel** dialog is displayed. Click **Disable** to confirm.

   The state of the channel changes to **Updating** and then to **Inactive**.
4. To enable an inactive channel, do one of the following:
   - Select the row of the channel to highlight it, and click the **Enable** button.
   - Click the name of the channel to open the **Channel Details** page. Click the **Enable** button.

   The **Enable MySQL Channel** dialog is displayed. Click **Enable** to confirm.

   The state of the channel changes to **Updating** and then to **Active**. The final state becomes **Needs Attention** if it encounters any error and replication cannot start.

Related Topics

- Importing Data

8.6.2 Editing a Channel

Editing a channel allows you to change the configuration of the channel.

> **Note**
>
> You cannot change the target DB System of a channel. To change the target DB System, you must delete the Channel and create a new Channel. See Deleting a Channel and Creating a Channel.

To edit a Channel:

1. In the MySQL HeatWave Console, select the **Resources** tab, and then click **Channels** to open the **Channels** page.
2. On the **Channels** page, in the list of channels, find the channel you want to edit, and do one of the following:
   - Select the row of the channel to highlight it, and choose the **Edit Channel** action from the **Actions** menu.

Copyright © 2022, 2025, Oracle and/or its affiliates.

- Click the name of the channel to open the **Channel Details** page. Click the **Edit Channel** button from the **Actions** menu.

The **Edit Channel** dialog is displayed.

3. In the **Edit Channel** dialog, update the required changes. You cannot change the target DB System.
4. Click **Save** to update the changes.

8.6.3 Resuming a Channel

The channel state becomes **Needs Attention** when an error occurs in the replication channel. After fixing the error, you can resume the channel to restart the replication.

To resume a Channel:

1. In the MySQL HeatWave Console, select the **Resources** tab, and then click **Channels** to open the **Channels** tab.
2. On the **Channels** tab, in the list of channels, find the channel you want to resume, and do one of the following:
   - Select on the row of the channel to highlight it, and choose the **Resume** action from the **Actions** menu.
   - Click the name of the channel to open the **Channel Details** page. Click the **Resume** button from the **Actions** menu.

   The **Resume MySQL Channel** dialog is displayed.
3. Click **Resume** to restart the replication.

The state of the channel becomes **Resuming**. When the channel restarts successfully, the state changes to **Active**. If the error persists or other error occurs, and the replication cannot start, the state changes to **Needs Attention**.

8.6.4 Resetting a Channel

Resetting a channel removes all replication data on the channel, except the channel configuration. It is equivalent to RESET REPLICA ALL FOR CHANNEL. When there is an unrecoverable issue, the reset operation clears the records associated with replication so that the channel can have a clean start.

The target DB System drops its position in the source binary log, clears the replication metadata repositories, deletes the relay log files, and starts a new relay log file.

You can only reset an **Inactive** channel. To reset a Channel:

1. In the MySQL HeatWave Console, select the **Resources** tab, and then click **Channels** to open the **Channels** tab.
2. On the **Channels** tab, in the list of channels, find the channel you want to reset, and do one of the following:
   - Click on the row of the channel to highlight it, and choose the **Reset** action from the **Actions** menu.
   - Click the name of the channel to open the **Channel Details** page. Click the **Reset** button from the **Actions** menu.

   The **Reset MySQL Channel** dialog is displayed.
3. Click **Reset** to reset the channel.

Copyright © 2022, 2025, Oracle and/or its affiliates.

The state of the channel becomes **Resetting**, and when the reset is complete, the state changes to **Inactive**. Then, you can enable the channel to restart the replication.

8.6.5 Deleting a Channel

Deleting a Channel stops the replication and removes the channel configuration.

To delete a Channel:

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

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

The state of the channel becomes **Deleting**, and when the delete is complete, the state changes to **Deleted**. You can click the name of the **Channel** to view the details. However, you cannot perform any other operation on the deleted channel.

8.7 Viewing Channel Details

Do the following to view Channel details:

1. In the MySQL HeatWave Console, select the **Resources** tab, and then click **Channels** to open the **Channels** tab.
2. On the **Channels** tab, in the list of channels, find the channel you want to view details for, and do one of the following:
   - Click the row of the Channel to highlight it. The Channel details appear below the list of channels.
   - Click the name of the Channel to open the **Channel Details** page for the selected Channel.

For descriptions of DB System details, see Channel Details.

- Channel Details

8.7.1 Channel Details

The **Channel Details** page contains the following tabs:

Table 8-3 Tabs on the Channel Details Page

Name	Description
Details	Channel details. See Table 8-4.
Channel Event Details	Channel event details. See Table 8-5.

Copyright © 2022, 2025, Oracle and/or its affiliates.

Table 8-4 Channel Details

Field	Description
General information	• **Name**: The name of the Channel. • **State**: The state of the Channel. – **Creating**: The Channel is being created. Provisioning can take several minutes. The channel is not yet replicating data from the source to the target. – **Active**: The Channel is running and replicating successfully. – **Inactive**: This status is displayed for the following reasons: * The Channel is not replicating because the user has disabled it. * The replica is a high availability DB System and a switchover is happening in it. – **Needs Attention**: This status is displayed for the following reasons: * The Channel is not replicating due to an error. * A restart, upgrade, or update is happening to the replica DB System. * The replica is a high availability DB System and a failover is happening in it. * The target DB System is `Inactive`, that is, the Channel is created or updated while the DB System is in `Inactive` state. Pending actions are applied when the DB System is started. – **Updating**: The Channel is in the process of being updated. – **Resuming**: The Channel is in the process of being resumed. – **Resetting**: The Channel is in the process of being reset. – **Deleting**: The Channel is being deleted. – **Deleted**: The Channel has been deleted and is no longer available. – **Resuming Error**: An attempt to resume the Channel failed. – **Resetting Error**: An attempt to reset the Channel failed. – **Updating Error**: An attempt to update the Channel failed. • **Resource ID**: The unique resource identifier assigned to the Channel when it is created. • **Description**: The user-defined description of the Channel. • **Replication status**: Replication is **Enabled** or **Disabled**. • **Last updated**: The date and time the Channel was last updated.

Copyright © 2022, 2025, Oracle and/or its affiliates.

Table 8-4 (Cont.) Channel Details

Field	Description
	• **Created**: The date and time the Channel was created.
Source	• **Network connection type**: – **Public**: Channel over public internet – **Private**: Channel over Egress PrivateLink • **Hostname**: The IP address or hostname of the source MySQL server. • **Port**: The port number the source MySQL server listens for incoming connections. • **Username**: The username of the replication user. • **Allowed outbound addresses**: (For a channel over public internet) IP addresses of the MySQL source from which the DB System replicates the data. • **SSL mode**: The security state of the connection to the source. Possible values are: – **Disabled**: No encryption required between source and target. – **Required**: (default) Establishes an encrypted connection if the server supports encrypted connections. The connection attempt fails if an encrypted connection cannot be established. – **Verify certificate authority**: Like Required, but additionally verify the CA certificate configured on the source against the Certificate Authority (CA) certificate (X509 PEM file). – **Verify identity**: Like Verify Certificate Authority, but additionally verify the source's hostname, defined in the source's SSL certificate, against the hostname defined in the Hostname field. • **SSL CA certificate**: The contents of the uploaded CA X509 certificate if the SSL mode is **Verify certificate authority** or **Verify identity**. • **Use GTID auto-positioning**: Whether the source server can provide the replica with GTID information for auto-positioning. Possible values are: – **Yes** – **No** • **Source of the UUID**: The source of the UUID for converting transactions that are anonymous to ones that have GTIDs when **Use GTID auto-positioning** is **No**. Possible values are: – **Manual input**: The UUID was user-defined or was generated by the Console. A **UUID** field shows the UUID value. – **Same as target DB system**: Use target DB System's UUID.

Copyright © 2022, 2025, Oracle and/or its affiliates.

Table 8-4 (Cont.) Channel Details

Field	Description
Target	• **DB System name**: The display name of the target DB System and a link to it. • **DB System state**: The state of the target DB System. • **Channel name**: The replication channel name. This is the replication channel used by the target MySQL DB System for the communication with the MySQL source. • **Applier username**: The username of the replication applier. • **Replication delay**: The amount of time, in seconds, that the channel waits before applying a transaction received from the source. • **Table without primary key**: How the creation and alteration of tables with no primary keys on the source are handled. Possible values are: – **Raise an error** – **Allow** – **Generate primary key**: Allows replicating a `CREATE TABLE` or `ALTER TABLE` transaction with no primary keys, and automatically generates a new invisible primary key column while creating a table without primary keys on the target.
PrivateLink (for channels over Egress PrivateLinks only)	• **Name**: Name of the PrivateLink. Click on the name to see the PrivateLink Details. • **State**: State of the PrivateLink. See Summary table of the PrivateLink Details page.
Channel filters	• **Type**: The filter type. • **Value**: The value for the filter type. See Channel Filter Rules for Inbound Replication for details on the filter types and their values .

Table 8-5 Channel Event Details

Field	Description
Type	The event and its status.
Severity	The severity level of the event, which can be one of • `INFO`: General information is provided. No special concerns • `WARNING`: Warning on an event that might impact the DB System's operation if no actions are taken. • `CRITICAL`: A critical event. An urgent action is required to avoid adverse impact on your DB System.
Message	The message body of the event report.
Created	The date and time the event report was created.

Copyright © 2022, 2025, Oracle and/or its affiliates.

See Events for details on event reports.

8.8 Limitations

Inbound replication in MySQL HeatWave on AWS does not support some of the configurations that are possible for MySQL replication.

- Only row-based replication is supported. This binary log format is the default. Statement-based replication and mixed replication are not supported.
- Only asynchronous replication is supported. Semi-synchronous replication is not supported.
- Only replication from a single source is supported. Multi-source replication is not supported.
- Changes to the `mysql` schema are not replicated and cause replication to stop.
- Using a `root@localhost` or `administrator` user in the DEFINER clause of CREATE PROCEDURE and CREATE FUNCTION statements is not supported. Instead, create a dedicated user with the privileges required to run the statements, and then edit the statements with the dedicated user in the DEFINER clause.

Also see the limitations described in Inbound Replication to a High Availability DB System.

Copyright © 2022, 2025, Oracle and/or its affiliates.

9 Manage Data in MySQL HeatWave

The **Workspace** tab in the MySQL HeatWave Console contains a **Data Management** tab, which enables you to perform the following:

- **Loading data into or unloading data from a MySQL HeatWave Cluster**: Load operations are performed using *Auto Parallel Load*, which optimizes operation time and memory usage by predicting and using an optimal degree of parallelism. Data is loaded into MySQL HeatWave from the associated DB System. If you have not loaded data into the DB System, see Importing Data. Unload operations are performed using *Auto Unload*.
- **Creating Lakehouse Data Mapping**: MySQL HeatWave Lakehouse enables query processing on the data residing in an Amazon S3 bucket. The source data is read from an Amazon S3 bucket, transformed into a memory-optimized format, stored in the MySQL HeatWave persistence storage layer, and then loaded to the MySQL HeatWave Cluster memory. MySQL HeatWave Lakehouse supports structured and relational data in CSV, Parquet, Avro, and JSON formats.
- **Refreshing Lakehouse Data**: Your MySQL HeatWave Lakehouse data can be refreshed by a **Lakehouse Incremental Load**.
- Loading or Unloading Data into MySQL HeatWave Cluster
- Creating Lakehouse Mapping
- Refreshing Lakehouse Data
- Maximum Number of Tables Loadable into a MySQL HeatWave Cluster

9.1 Loading or Unloading Data into MySQL HeatWave Cluster

Use MySQL HeatWave Console to load data into or unload data from a MySQL HeatWave Cluster.

This task requires 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 MySQL HeatWave Cluster depends on the data size.

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

To load data, do the following:

Copyright © 2022, 2025, Oracle and/or its affiliates.

1. Connect to a DB System that has a MySQL HeatWave Cluster. See Connecting from the Console.
2. On the **Workspace** tab of the MySQL HeatWave Console, click the **Data Management** tab. You can view the following information about schemas and tables:
   - **Name**: The name of the schema or table.
   - Source: The source of the schema or table. For example, InnoDB.
   - **Memory Size Estimate (GiB)**: An estimate of the memory required on the MySQL HeatWave Cluster for the schema or table.
   - **Loaded**: The number of tables loaded into HeatWave. You can click the **Refresh** icon to refresh the load status.
   - **Rows Estimate**: The estimated number of rows in the table.
   - **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 MySQL HeatWave string column encoding, see Encoding String Columns, in the *MySQL HeatWave User Guide*.
   - **Predicted Load Time (s)**: The predicted load time in seconds.
3. Select the schemas and tables that you want to load by clicking the check box present in front of the schema or table name. If you select a schema, all the tables in the schema are selected. Expanding a schema using the drop-down control displays the table in the schema. The **Load Status** column provides the load status as a percentage value.
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
   - **Tables to load**: Number of tables to be loaded
   - **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 MySQL HeatWave Cluster
   - **Estimated total memory footprint**: A pie chart showing the estimated percentage of MySQL 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

Copyright © 2022, 2025, Oracle and/or its affiliates.

  - **Memory Size Estimate (GiB)**: The memory size estimate for the schema or table on the MySQL HeatWave Cluster
- **Estimated memory foot print (GiB)**: The estimated MySQL 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 MySQL HeatWave Cluster memory used.

If a load operation fails, the error message contains a **View Details** link that opens the **Load operation summary** pane, which contains the following information:

- The failed query that was executed for the load operation.
- The query results, which are displayed in a **Query Results** pane similar to the one described in Running Queries.
- Unload Advisor

9.1.1 Unload Advisor

Unloading tables from the MySQL HeatWave Cluster that have not been queried recently can free up memory for other tables and allow for a better use of available resources. To help identify unused tables in the MySQL HeatWave Cluster, the MySQL HeatWave Console provides an **Unload Advisor**, which recommends tables to be unloaded based on the query history.

To use the **Unload Advisor**:

1. Connect to the DB System for the MySQL HeatWave Cluster from which you want to unload tables. See Connecting from the Console.
2. On the **Workspace** tab of the MySQL HeatWave Console, click the **Data Management** tab.
3. Under the **Actions** menu, select **Unload Advisor**.
4. In the **Unload Advisor** dialog that appears, choose the **Time Period** for which the advisor will check for tables that have not been queried. Choices include **Past 7 days, Past 14 days**, and **Past 28 days**.
5. Click **Get Unload Advisor Recommendations**. A message is shown when the advisor recommendations are being fetched.
6. Either one of the following then appears:
   - A message saying **Unload Advisor operation completed with no recommendations**, because there are no unqueried tables for the chosen period of time.
   - An **Unload tables from HeatWave** dialog box, showing the following information:
     - The DB System involved.
     - The number of **Tables to unload** according to the advice.
     - The **Estimated unload size**.
     - A table that shows the schemas and tables to be unloaded and their sizes.

     Click **Unload Tables** to accept the advice to unload the tables. A pop-up notification appears after the unload operation is completed.

Copyright © 2022, 2025, Oracle and/or its affiliates.

9.2 Creating Lakehouse Mapping

For MySQL HeatWave Lakehouse to process data in Amazon S3, it needs to first map the data in files on S3 to a MySQL HeatWave Lakehouse table. Once the S3 data is provided, the schema for the Lakehouse table is inferred automatically by MySQL HeatWave Autopilot. After the data mapping is created, it allows the S3 data to be loaded into the Lakehouse table for querying by Lakehouse.

> **Note**
>
> To map and load data into Lakehouse, the Amazon S3 bucket in which the data is located and the DB System must be in the same region.

This task requires the following:

- A running DB system with an active MySQL HeatWave Cluster. See Creating a MySQL HeatWave Cluster.
- Properly configured Lakehouse IAM role ARN. See Creating an IAM Role to Access an Amazon S3 Bucket.
- S3 URIs for one or more files of similar data.

Do the following to create a Lakehouse data mapping:

1. Connect to a DB System that has a MySQL HeatWave Cluster with Lakehouse enabled. See Connecting from the Console.
2. Click the **Data Management** tab, and then click **Create Lakehouse Mapping**.
3. **Source**:

> **Note**
>
> You can map more than one Amazon S3 bucket file to the destination schema and table, but all the files should be of the same format (i.e., they are all either **CSV**, **Parquet**, **Avro**, or **JSON** files). Use the **Add new file** button to create multiple entries for data sources,

   - **File path type**: Select either of the following:
     - **Prefix**: Enables you to map all files present under an Amazon S3 bucket prefix to the destination schema and table.
     - **Name**: Enables you to map a specific file present in an Amazon S3 bucket to the destination schema and table.
     - **Pattern**: Enables you to map a set of files present in an Amazon S3 bucket defined by a regular expression to the destination schema and table.
   - **File path**: Depending on the **File path type** you select, specify one of the following
     - The Amazon S3 bucket prefix path, `s3://<bucket-name>/<prefix>`
     - The Amazon S3 bucket path with the file name, `s3://<bucket-name>/<prefix>/<filename>`

Copyright © 2022, 2025, Oracle and/or its affiliates.

  - The Amazon S3 bucket path containing a regular expression, `s3://<bucket-name>/<regex>` (the pattern of the regular expression follows the modified ECMAScript regular expression grammar)

- **Strict mode**: Select **Default**, **Enabled**, or **Disabled**, which controls how Lakehouse handles invalid or missing values for data-changing statements such as INSERT or UPDATE when loading data into MySQL HeatWave. This parameter overrides the **Strict mode** setting that you specify in the **File parsing settings** for **CSV** files. See Strict SQL Mode. The following shows what happens if parsing errors, formatting errors, or empty columns are found in a file during data loading, for each possible value of **Strict mode**:
  - **Default**: The situation is handled according to the MySQL server's SQL mode setting.
  - **Enabled**: An error is displayed, interrupting the operation.
  - **Disabled**: A warning is displayed, without interrupting the operation. Empty columns are automatically filled with default column values or NULL.

4. **File parsing settings**:

   - **Format**: Select the format of the file you want to map. You can select **CSV**, **Parquet**, **Avro**, or **JSON**. The default format is **CSV**. All the files that you want to map should have the same format.

     > **Note**
     >
     > See MySQL HeatWave Lakehouse Limitations for the limitations for mapping and loading files of each of the supported formats.

     If you select the **CSV** format, you can set the following:

     - **Skip rows**: Specify the number of rows that is skipped from the beginning of the data file (applicable to every file included in the file list). By default, no rows are skipped, and the maximum rows to skip is 20.
     - **Field delimiter**: Specify one or more characters to enclose fields. The maximum field delimiter length is 64 characters. The default delimiter is `,`.
     - **Record delimiter**: Specify one or more characters to delimit records. The maximum record delimiter length is 64 characters. The default delimiter is `\n`.
     - **Escape character**: Specify one or more characters to escape special characters. The escape character is `\`.
     - **Quotation marks**: Specify one or more characters to enclose fields. The default character to enclose fields is `"`.
     - **Encoding**: Specify the character set to map data. The default character set is `utf8mb4`. See Character Sets and Collations in MySQL.
     - **Date format**: See Date_format. The default date format is `auto`.
     - **Time format**: See String and Numeric Literals in Date and Time Context. The default time format is `auto`.
     - **Strict mode**: Specify whether the mapping takes place during data mapping in strict mode (**Enabled**) or non-strict mode (**Disabled**). See Strict SQL Mode
       * **Default**: The situation is handled according to the MySQL server's SQL mode setting.

Copyright © 2022, 2025, Oracle and/or its affiliates.

        - **Enabled**: If parsing errors, formatting errors, or empty columns are found in the file, an error is displayed, interrupting the operation.
        - **Disabled**: If parsing errors, formatting errors, or empty columns are found in the file, a warning is displayed, without interrupting the operation. Empty columns are automatically filled with default column values or NULL.
    - **Has header**: Specify whether the file has a header.
        - **Default**: Default value is **Disabled**
        - **Enabled**: Treat the first row of the file as the header
        - **Disabled**: Treat the file as if it does not have a header.
    - **Trim spaces**: Select whether to remove leading and trailing spaces.
        - **Enabled**: Remove leading and trailing spaces.
        - **Disabled**: Do not remove leading and trailing spaces.
5. **Destination**: Specify the destination of the Lakehouse table into which the data is to be loaded.
    - **Schema**: Name of the destination schema.
    - **Table**: Name of the destination table.
6. Click **Next**.
7. Under **Autopilot schema inference**:
    - Review the **Definitions**, **Errors**, and **Warnings**, and fix any reported issues.
    - Click the **Refresh** button to refresh the step with the latest MySQL HeatWave schema and table information, to make sure the mapping remains valid after it was first generated.
8. Under **MySQL HeatWave load**:
    - Note the **Estimated memory required**
    - The **Load into MySQL HeatWave after creating Lakehouse mapping** option is selected by default if there are no **Errors** or **Warnings**; deselect if you want.
9. Do one of the following:
    - Click **Create Lakehouse Mapping** to create the destination Lakehouse table (which remains empty after its creation, until data is loaded).
    - Click the **Back** button to go back and change your mapping settings.
    - Click the **Cancel** button to cancel the mapping.
    - Click the **Copy** button to copy the DDL statement for data mapping into your clipboard.

> **Note**
>
> Mapping data onto an existing Lakehouse table is not supported. Mapping will fail if the specified **Destination** points to a Lakehouse table that already exists. If **Destination** points to an existing InnoDB table on the target MySQL server, the mapping operation will change the destination table on the server into a Lakehouse table if the table is empty, or the operation will fail if the table is not empty.

If you have not selected **Load into MySQL HeatWave after creating Lakehouse mapping**, once the Lakehouse table is successfully created, you can load the mapped data into your

Copyright © 2022, 2025, Oracle and/or its affiliates.

MySQL HeatWave Cluster by following the instructions in Loading or Unloading Data into MySQL HeatWave Cluster.

Related Topics

- Refreshing Lakehouse Data

9.3 Refreshing Lakehouse Data

The MySQL HeatWave Lakehouse tables contain a snapshot of your data in Amazon S3 at the time you loaded the data into your MySQL HeatWave Cluster. If the data has been altered or removed subsequently on Amazon S3, or if you want to add similar data from Amazon S3 that maps to your Lakehouse tables, you can perform a **Lakehouse Incremental Load** when using MySQL 9.1.0 or later.

In a **Lakehouse Incremental Load**, it is not possible to change the parameters you used to create the Lakehouse mapping initially (for example, the file type, file path type, or file parsing settings). However, individual files for data sourcing are added or removed according to the original file prefix or file path; for example, if your **File path type** is **Prefix** and a new file similar to the original files has been added to the location specified by the Amazon S3 bucket prefix path, the data in the new file will be loaded into the Lakehouse table by the refresh.

Here are the prerequisites for refreshing a MySQL HeatWave Lakehouse table:

- What are required for Creating Lakehouse Mapping.
- The privileges to load data into your MySQL HeatWave Cluster, as described in Loading or Unloading Data into MySQL HeatWave Cluster.
- An existing Lakehouse table in your MySQL HeatWave Cluster.
- The DB System that your MySQL HeatWave Cluster is associated with must be running MySQL 9.0.0 or later.
- Changes have been made to the files under the Amazon S3 bucket that were used to create Lakehouse mapping initially, or similar files have been added to the same file path.

Do the following to refresh your Lakehouse data with a **Lakehouse Incremental Load**:

1. Connect to the DB System that is associated with the Lakehouse table to be refreshed. See Connecting from the Console.
2. Click the **Data Management** tab, and select on it the databases containing the Lakehouse tables you want to refresh. Alternatively, you can expand the table list for each database, and then select specific Lakehouse tables for refresh.
3. Click the **Refresh Lakehouse Data** button on top. The **Refresh Lakehouse Data** dialog box shows the following:
   - The **DB System** which is associated with Lakehouse tables to be refreshed.
   - The number of **Tables to refresh**.
   - The **Schemas and tables** to be refreshed.
4. Click **Refresh** to continue. A message appears when the data refresh is completed.
5. If you want to cancel the refresh before it is completed, click the **Actions** button on top and then select **Cancel Lakehouse Refresh**.

If the refresh fails, the error message contains a **View Details** link that opens the **Lakehouse refresh summary** pane, which contains the following information:

- The failed query that was executed for the refresh operation.

Copyright © 2022, 2025, Oracle and/or its affiliates.

- The query results, which are displayed in a **Query Results** pane similar to the one described in Running Queries.

Related Topics

- Lakehouse Incremental Load in the MySQL HeatWave User Guide

9.4 Maximum Number of Tables Loadable into a MySQL HeatWave Cluster

The maximum number of tables that can be loaded into a MySQL HeatWave Cluster depends on the shape of the DB system and the total number of columns of the tables to be loaded. If the tables have more columns, fewer tables can be loaded. See Table 9-1 for the approximate maximum number of columns and tables for each shape. As an illustration, the table shows the maximum number of tables with an average of 100 columns per table.

> **Tip**
>
> You can use the MySQL HeatWave Autopilot to verify that there is sufficient memory to load the tables into your MySQL HeatWave Cluster.

Table 9-1 Limits on the Number of Columns and Tables Loadable into a MySQL HeatWave Cluster

DB System Shape	Maximum number of columns (approx.)	Maximum number of tables with an average of 100 columns per table (approx.)
MySQL.2.16GB	81,900	819
MySQL.4.32GB	81,900	819
MySQL.8.64GB	81,900	819
MySQL.32.256GB	3,538,900	35,389

> **Note**
>
> The estimates assume numerical data for the tables. The maximum number of columns and tables that can be loaded is reduced with string columns, because the data dictionary encoding for them incurs more memory.

Copyright © 2022, 2025, Oracle and/or its affiliates.

10

Running Queries

The Workspace tab in the MySQL HeatWave Console provides a Query Editor for running MySQL and MySQL 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. Select the **Workspace** tab in the MySQL HeatWave Console, and then select the **Query Editor** tab.
3. Enter a query into the query editor text box.
4. Click **Run Query**.

   The **Query Results** pane displays the results 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.

Query Editor Limitations

The Query Editor has the following limitations:

- It is not intended for loading data into a DB System. More efficient methods of loading data into a DB System are available. For more information, see Importing Data.
- Only results for the last executed query are displayed.
- Multi-set results from calling stored procedures are not displayed. Only the last result is displayed.
- Results from queries with SQL code comments might not be displayed accurately.
- Long query results might be truncated.

- Running MySQL HeatWave Queries

10.1 Running MySQL HeatWave Queries

When a MySQL HeatWave Cluster is enabled and the data you want to query is loaded in MySQL HeatWave, queries that qualify are automatically offloaded from the MySQL DB System to the MySQL 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.

Copyright © 2022, 2025, Oracle and/or its affiliates.

Before running a query, you can use `EXPLAIN` to determine if the query will be offloaded to MySQL 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 MySQL 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 MySQL HeatWave queries, see Running Queries, in the MySQL HeatWave User Guide. You can also refer to the *Quickstarts* in the MySQL HeatWave User Guide, which show how to import data into a DB System, load data into MySQL HeatWave, and run queries:

- tpch Analytics Quickstart
- AirportDB Analytics Quickstart

Copyright © 2022, 2025, Oracle and/or its affiliates.

11

Autopilot Index Advisor

Autopilot Index Advisor provides secondary index suggestions to improve the performance of OLTP workloads on DB Systems running MySQL 9.0.0 or later.

Indexes can improve query performance of SELECT statements, but managing the indexes increases the overhead of INSERT, UPDATE, and DELETE statements. The goal of Autopilot Index Advisor is to generate secondary index recommendations that optimize the overall workload performance.

Autopilot Index Advisor generates recommendations to add and drop indexes. Each recommendation also includes:

- An estimate of the overall workload performance benefit
- An estimate of the overall storage footprint impact
- A DDL statement, an explanation, and an estimated storage impact for each index recommendation
- An estimate of the index creation time and the estimated performance gain of the top five queries of each create index recommendation

The workload on the DB System is obtained from the statement digest history stored in the Performance Schema. The workload needs to have a minimum set of five queries that were executed against the tables in a user schema. Autopilot Index Advisor evaluates the statements and identifies potential indexes that can help improve the performance of the workload, as well as any existing indexes that are decreasing the performance.

- Running Autopilot Index Advisor

Related Topics

- Autopilot Indexing in the HeatWave User Guide

11.1 Running Autopilot Index Advisor

To run Autopilot Index Advisor, the user must connect to the DB system and have the following MySQL privileges:

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

Perform the following steps to run Autopilot Index Advisor on the MySQL HeatWave Console:

1. Connect to the DB System. See Connecting from the Console.
2. In the MySQL HeatWave Console, click the **Workspace** tab, and then click the **Index Advisor** tab. The Advisor runs on all the user schemas on the DB System.
3. View the index suggestions in the output.

   The suggestions provide the following information per index:

   - **Index No.**
   - **Recommendation**: It is either **DROP** or **CREATE**

Copyright © 2022, 2025, Oracle and/or its affiliates.

- **Schema Name**
- **Table Name**
- **Indexed Columns**
- **Reason**: Reason for making the recommendation. Reasons for **DROP** include **Unused Index** and **Duplicate Index**, and reason for **CREATE** is **Missing Index**.
- **Performance Impact**: This is not given for a **DROP** recommendation, and is **HIGH** or **LOW** for a **CREATE** recommendation.
- **Estimated Create Time**: Estimated time for creating the recommended index (only given for **CREATE** index recommendations).

Select one or more of the suggestions by clicking on it. The bottom pane shows more information on the selected changes (click on the  icon for each of the items for more information):

- **Estimated total speedup**: A speedup estimate is provided only for **CREATE** recommendations, and only achievable if all selected **CREATE** recommendations are executed.
- **Total DB System data storage**
- **Estimated storage impact**
- **Selected recommendations**
- **Performance/storage tradeoff plot**: The storage impact for each recommendation and for all the recommendations together are plotted, with the performance impact indicated in the plot legend.

Click the **Show Affected Queries** button to open the **Queries most affected by the selected recommendations pane**, which provides the following information for each of the queries (the information is listed in table format in the upper pane, and in more details in the lower pane when a query in the table is selected):

- **Query Text**
- **Index ID**
- **Reason for recommendation**
- **Current execution time**
- **Estimated speedup**

4. Click the **Refresh** button to refresh the recommendations if you want, to get the latest recommendations according to the recent queries you ran on the DB System
5. Click the **Apply Recommendations** button to apply the recommendations you have selected. An **Apply Selected Index Recommendations** dialog box appears, in which you are asked to **Confirm** (or **Cancel**) the action.

   The recommendations for creating or dropping indexes are executed automatically.

> **Tip**
>
> For **DROP** recommendations, you can make the indexes invisible first—if you encounter any errors later, you can make the indexes visible again; if there are no errors after some time, you can drop the indexes permanently.

Copyright © 2022, 2025, Oracle and/or its affiliates.

> **Warning**
>
> If you drop any indexes that were used in the index hints `FORCE INDEX`, `USE INDEX`, or `IGNORE INDEX` for some queries, those queries will fail to execute.

Once the recommendations are applied, queries will start taking advantage of the changes for better performance. Repeat running the Autopilot Index Advisor to get new recommendation when, for example, your workload changes.

Related Topics

- Autopilot Indexing in the HeatWave User Guide

Copyright © 2022, 2025, Oracle and/or its affiliates.

12

High Availability

A high availability DB System is made up of three MySQL instances: one primary instance and two secondary instances. All data written to the primary instance is also written to the secondary instances. A high availability DB System assures that when the primary instance fails, its function is taken over by one of the secondary instances, with zero data loss and minimal downtime.

- Overview of High Availability
- Prerequisites
- Enabling and Disabling High Availability
- Failover
- Switchover
- Maintenance of a High Availability DB System
- Restoring a Backup of a High Availability DB System
- Inbound Replication to a High Availability DB System
- Limitations for High Availability
- Billing

12.1 Overview of High Availability

A high availability DB System is made up of three MySQL instances: a primary instance and two secondary instances. Each MySQL instance utilizes the same amount of block volume storage, CPU count, and RAM size defined in the chosen shape. The primary instance functions as the only read/write endpoint. All data that you write to the primary instance is copied to the secondary instances asynchronously.

All the three MySQL instances of a high availability DB System are placed in different availability zones (AZs) in the same AWS region containing the DB System.

High availability DB Systems consume more resources (CPUs, RAM, network bandwidth, etc.) than standalone DB Systems. Also, write transactions require coordination and approval by multiple MySQL instances in the high availability DB System. Hence, the throughput and latency for a high availability DB System differ from those of a standalone DB Systems.

A high availability DB System uses MySQL Group Replication to replicate data from the primary instance to the secondary instances. The replication occurs over a secure, managed, internal network. Limited information about this internal network (for example, host names and port numbers) is available in some Performance Schema tables (for example, replication_group_members), and you can neither connect to the internal network nor view any other information related to it. In a Group Replication setup, the binary logs for the different MySQL instances are managed independently from each other: even though the instances have the same data, they might have different number of binary log files with different file names and possibly different sizes.

Promotion of a Secondary Instance

Copyright © 2022, 2025, Oracle and/or its affiliates.

- **Failover**: In case the primary instance fails, MySQL HeatWave Service *automatically* promotes one of the secondary instances to function as the primary instance. This resumes availability to client applications with no data loss.
- **Switchover**: MySQL HeatWave Service allows you to *manually* promote one of the secondary instance as the primary instance. This is called a switchover.

> **Note**
>
> After a failover or switchover, the current binary log file name and position of the new primary may be different from the old primary. As the binary logs of each instance are managed independently, each transaction recorded in the binary logs may be written to a different binary log file and position in different instances.

Preferred and Current Primary Placement

- **Preferred primary placement**: While creating a high availability DB system, you can choose the AZ where you want to place the primary instance, which functions as a read/write endpoint. This is called the *preferred primary placement*, and it does not change once you have created a high availability DB system, except when you perform a switchover. The secondary instances are placed automatically in two other AZs.
- **Current primary placement**: When you create a DB system, the current placement of the primary instance is the same as the preferred placement. However, in the event of a failover, one of the secondary instances is promoted as the primary instance. The AZ of this new primary instance becomes the *current primary placement*. The preferred primary placement, which you selected while creating the DB system, remains the same. The DB System can be accessed through the same DB System host name as before, regardless of the placement of the new primary instance.

Related Topics

- Group Replication

12.2 Prerequisites

Here are some prerequisites for creating a high availability DB System.

- A high availability DB System requires a high availability compatible configuration. All default MySQL configurations that are compatible with high availability are indicated as so in the MySQL Configuration Details, If you want to create a custom configuration that supports high availability, remember to select **Support High Availability** when Creating a MySQL Configuration or Copying a MySQL Configuration.
- As the DB system high availability feature uses MySQL Group Replication, for which each table must have a primary key. If you try to create a table without a primary key in a high availability DB system, it will fail.

  If you are migrating tables to MySQL HeatWave on AWS that do not have primary keys, you must add them. Check tables for primary keys and add keys to the tables that do not have them:

  1. Use a command-line client such as the `mysql` client or MySQL Shell to check your tables for primary keys and list the ones that do not have primary keys:

     ```
     SELECT t.table_schema, t.table_name
     FROM information_schema.tables t
     ```


Copyright © 2022, 2025, Oracle and/or its affiliates.

```
  LEFT JOIN (SELECT table_schema, table_name
             FROM information_schema.statistics
             WHERE index_name = 'PRIMARY'
             GROUP BY table_schema, table_name, index_name
             ) pks
  ON t.table_schema = pks.table_schema AND t.table_name =
pks.table_name
WHERE pks.table_name IS NULL
  AND t.table_type = 'BASE TABLE'
  AND t.table_schema NOT IN ('mysql', 'sys', 'performance_schema',
'information_schema');
```

2. If primary keys are missing, add them using any of the following methods:
   - Run a command similar to the following against the table to which you want to add the invisible column and primary key:

     ```
     ALTER TABLE <Table1> ADD <my_row_id> BIGINT UNSIGNED NOT NULL
     AUTO_INCREMENT PRIMARY KEY INVISIBLE FIRST;
     ```

     The command alters the table, <*Table1*>, by adding a column, <*my_row_id*>, which is invisible and contains the primary key for the table.

     > **Note**
     >
     > Using invisible columns to add primary keys is a low-impact way to update your existing data for use with a high availability DB system. It is transparent to your applications—the new column remains hidden from `SELECT` queries, enabling your applications to continue working as previously.

   - Using MySQL Shell dump utility: See `create_invisible_pks` in MySQL Shell Dump Utility.
   - Using MySQL Shell load utility: See `createInvisiblePKs` in MySQL Shell Load Utility.

     > **Note**
     >
     > To use MySQL Shell dump and load utility, use MySQL Shell version 8.0.30 or higher.

   - Using `sql_generate_invisible_primary_key` variable: Set the variable to `ON` to add primary keys to new tables that you create. Setting the variable to `ON` does not add primary keys to already existing tables. See Generating Invisible Primary Keys.

- Automatic Backups must be enabled, and are enabled for you when you create a high availability DB System. You can choose a retention period for the automatic backups.

12.3 Enabling and Disabling High Availability

This section explains the enabling and disabling of high availability DB systems.

Copyright © 2022, 2025, Oracle and/or its affiliates.

- Enabling high availability on a DB System
- Disabling high availability on a DB System

12.3.1 Enabling high availability on a DB System

Enabling high availability can be done when you are Creating a DB System, or you can enable high availability on a standalone DB System by following the steps below, which requires the following:

- A standalone DB system in the **Active** state.
- The MySQL configuration of the DB System supports high availability. If it does not, update the MySQL configuration and select a configuration that supports high availability.
- No MySQL HeatWave Cluster is attached to the DB System. If there is, delete the MySQL HeatWave Cluster.
- Automatic backups are enabled on the DB System. If they are not, see Editing a DB System on how to enable them.

> **Note**
>
> Optionally, if there has been a high write load on the DB System since the last backup, take a manual backup of the DB System before enabling high availability to speed up the enabling operation.

- No data bulk ingestion is running. If bulk ingestion is going on, stop it, or wait until it is finished before enabling high availability.
- No PrivateLinks are attached to the DB System. If there are any, delete them. Once the operation for enabling high availability is completed, reattach the PrivateLinks to the high availability DB System.
- Any replication channel attached to the DB System must use GTID auto-positioning. If it does not, change it by editing the channel.
- Primary keys exist on every table in the database. See Prerequisites on how to check if this is satisfied, and how to add primary keys to your tables if needed.
- If you intend to configure an inbound replication channel on this DB system, you must import data before enabling high availability, and configure your channel after high availability is enabled.

To enable high availability on a standalone DB system, do the following:

1. In the MySQL HeatWave Console, select the **Resources** tab.
2. On the **DB Systems** tab, in the list of DB Systems, find the DB System for which you want to enable high availability, and do one of the following:
   a. Click the row of the DB System to highlight it, and then select from the **Actions** menu **Enable High Availability**.
   b. Click the name of the DB System to open the **MySQL DB System Details** page for the selected DB System, and then click the **Enable High Availability** button.
3. In the **Enable High Availability** dialog box, click **Enable** to confirm that you are enabling high availability, or click **Cancel**.

   If an error is returned because any of the prerequisites is not satisfied, correct the problem by following the instructions given at the beginning of the section and retry.

Copyright © 2022, 2025, Oracle and/or its affiliates.

The DB system enters the UPDATING state. The secondary instances are cloned from the primary instance and a high availability DB system is formed. The process does not cause any downtime to the DB system. If the updating process fails, the DB system returns to a standalone state. Check the service events for details of the failure.

12.3.2 Disabling high availability on a DB System

When disabling high availability on a DB system, the primary instance located in the preferred primary placement is converted to a standalone DB system and the two secondary instances are deleted.

This task requires the following:

- A running DB system with high availability enabled.
- The DB System has no PrivateLinks attached to. If it has any, delete them from the DB System. Once the operation for disabling high availability is completed, recreate the PrivateLinks for the standalone DB System.

To disable high availability for a DB system, do the following:

1. In the MySQL HeatWave Console, select the **Resources** tab.
2. On the **DB Systems** tab, in the list of DB Systems, find the DB System for which you want to disable high availability, and do one of the following:
   a. Click the row of the DB System to highlight it, and then select from the **Actions** menu **Disable High Availability**.
   b. Click the name of the DB System to open the **MySQL DB System Details** page for the selected DB System, and then click the **Disable High Availability** button.
3. In the **Disable High Availability** dialog box, click **Disable** to confirm that you are disabling high availability, or click **Cancel**.

The DB system enters the UPDATING state. If the primary instance is running in the preferred primary placement, no downtime occurs. If the primary instance is not running in the preferred primary placement, a failover occurs and this causes a short downtime.

> **Note**
>
> You can avoid the downtime by changing the preferred primary placement of the DB system to the current primary instance by performing a Switchover before disabling high availability.

If the updating process fails, the DB system returns to a high availability state. To view the reasons for the failure, navigate to the **Service Events** pane (see Events) and check the event details.

12.4 Failover

In the event of a failure of the primary instance, one of the secondary instances residing in a separate AWS availability zone (AZ), is automatically promoted as the primary instance.

Copyright © 2022, 2025, Oracle and/or its affiliates.

> **Note**
>
> After a failover, the current binary log file name and position of the new primary may be different from the old primary. As the binary logs of each instance are managed independently, each transaction recorded in the binary logs may be written to a different binary log file and position in different instances.

When you create a high availability DB System, the current placement of the primary instance is the same as the preferred placement. However, in the event of a failover, one of the secondary instances is promoted as the primary instance. The availability zone (AZ) of this new primary instance becomes the current placement. The preferred placement of the primary instance, which you selected while creating the DB system, remains the same.

The DB System host name of the high availability DB system does not change, regardless of the placement of the primary instance.

Once the issue that caused the failover has been corrected, the original primary instance returns to the DB system as a secondary instance. In case there is another failover, the original primary instance is promoted as the current primary instance.

Failover Reasons

Table 12-1 lists possible reasons for a failover.

Table 12-1 Failover Reasons

Failover	Description
Hardware	• Storage failures • Network failures • Availability Zone failures • Host failures • Out of memory issues
MySQL Server	• MySQL process stops • Operating System stops • MySQL instance or process is slow or overloaded • Replication errors

12.5 Switchover

You can *manually* promote one of the secondary instances of your high availability DB System to be the primary instance. This is called a switchover.

When you perform a switchover, the preferred placement of the DB System changes to the newly selected placement for the primary instance (that is, the current placement of the primary instance and the preferred placement become the same after the switchover). The hostname of the DB system endpoint does not change, but the endpoint is being appointed to the new primary instance, during which there will be a short period of downtime for the DB System. Your applications' database connections to DB System must be reestablished afterwards.

Copyright © 2022, 2025, Oracle and/or its affiliates.

> **Note**
>
> After a switchover, the binary log file's name and position of the new primary may be different from those of the old primary. As the binary logs of each instance are managed independently, each transaction recorded in the binary logs may be written to a different binary log file and position in different instances.

Switchover Scenarios

Here are some scenarios for performing a switchover:

- **Application testing**: To ensure your application works correctly during a failover, you can test that by performing a switchover.
- **Testing proximities of availability zones**: Even though the MySQL instances in different availability zones are connected with a low-latency network, it may be necessary to test that latency by moving primary instance from one availability zone to another to confirm the best location for your primary instance.
- Switchover Using the Console

12.5.1 Switchover Using the Console

This task requires the following:

- A running DB system with high availability enabled.

Do the following to switchover from the current primary instance to one of the secondary instances:

1. In the MySQL HeatWave Console, select the **Resources** tab.
2. On the **DB Systems** tab, in the list of DB Systems, find the high availability DB System for which you want to perform a switchover, and do one of the following:
   a. Click on the row of the DB System to highlight it, and then choose **Switchover** from the **Actions** menu.
   b. Click the name of the DB System to open the **DB System Details** page, and then click the **Switchover** button.
3. In the **Switchover to another Availability Zone** panel
   a. Select the **Preferred availability zone**: This is the availability zone in which the new primary will be located. It will also be the new preferred availability zone for your high availability DB System after the switchover. Your choices include all the availability zones your MySQL instances are now in.
   b. Check the **Summary** for your **Current placement of the primary**, **Current preferred placement of the primary**, **New preferred placement of the primary**, and **Resulting action**. Change your selection for the **Preferred availability zone** if needed.
   c. Click **Switchover** to start the switchover, or **Cancel**.

During the switchover, there will be a short period of downtime (that is not the case if the preferred availability zone selected in the same as the current primary's availability zone, which means you are only changing the preferred availability zone and no real switchover takes place). See Switchover on other effects of the switchover process.

Copyright © 2022, 2025, Oracle and/or its affiliates.

12.6 Maintenance of a High Availability DB System

Oracle maintains a high availability DB System using a rolling maintenance process that has minimal downtime. The maintenance starts within two hours of the **Maintenance window start time** that you defined for the DB system.

The rolling maintenance process involves the following steps and does not require any input from you:

1. One of the secondary nodes is removed from the replication group.
2. The maintenance process is performed on the selected node, which is returned to the replication group when the maintenance completes.

   The returned secondary is allowed to catch up with the transactions that occurred while it was being maintained, before the maintenance of the remaining secondary begins.
3. The remaining secondary node is removed from the replication group.
4. The maintenance process is performed on the selected node, which is returned to the replication group when the maintenance completes.

   The returned secondary is allowed to catch up with the transactions that occurred while it was being maintained, before the next step begins.
5. Existing connections to the primary are closed and no new connections are permitted. The primary node is removed from the group and one of the secondary nodes that have gone through maintenance is promoted to be the primary, like what happens during a Failover. It is during this step that a brief period of downtime occurs before the newly promoted primary resumes connections.

   > **Note**
   >
   > The current primary placement changes to that of the new primary. The preferred primary placement remains unchanged.

   > **Note**
   >
   > The current binary log file name and position of the new primary may be different from the old primary's. As the binary logs of each instance are managed independently, each transaction recorded in the binary logs may be written to a different binary log file and position in different instances.
6. The maintenance process is performed on the old primary node, which is returned to the replication group as a secondary node. **No switching back to the old primary node is performed, to avoid additional downtime**.

12.7 Restoring a Backup of a High Availability DB System

A backup of a high availability DB System can be restored to either a new standalone DB System or to a new high availability DB System. See Restoring a Backup to a New DB System for instructions.

The following limitations apply when restoring a backup of a high availability DB System:

Copyright © 2022, 2025, Oracle and/or its affiliates.

- The data storage size of the high availability DB System cannot be reduced, but can be increased.
- You cannot restore a backup of a standalone DB System to a high availability DB System.

12.8 Inbound Replication to a High Availability DB System

MySQL HeatWave on AWS supports Inbound Replication to a high availability DB System.

The inbound replication channel is connected to the primary instance of the high availability DB System. Transactions on the source DB System are replicated first to the primary instance, and then asynchronously to the secondary instances. During a Failover or an upgrade for the high availability DB System, replication pauses, and the replication channel is put into the `Needs Attention` state. After a secondary instance has been promoted to become the new primary, the replication channel is reconfigured automatically to connect to the new primary. The channel is then restored to the `Active` state and replication resumes.

The following limitations apply when replicating to a high availability DB System:

- GTID-based replication must be used.

12.9 Limitations for High Availability

A high availability DB system has certain limitations.

- You can access the MySQL instance that functions as the primary instance only. You cannot access the other two secondary instances directly using MySQL Shell or any other clients.
- The maximum size of transaction depends on the amount of memory of the shape and is listed in Limitations for High Availability.

Table 12-2 Maximum Size of Transaction for a High Availability DB System

Shape Memory (GiB)	Maximum Transaction Size (Bytes)
16	171,798,692
32	343,597,384
64	687,194,767
256	1,073,741,824

- The maintenance of a high availability DB system requires rolling upgrades, for which there is a brief period of downtime before the newly promoted MySQL instance resumes connections. Each MySQL instance is upgraded separately. See Maintenance of a High Availability DB System.
- MySQL HeatWave Clusters, as well as any features that require a MySQL HeatWave Cluster (for example, MySQL HeatWave Lakehouse, MySQL HeatWave GenAI, or MySQL HeatWave AutoML), are not currently supported on a high availability DB system.
- When using a Query PrivateLink endpoint, failovers and switchovers take longer than on a public endpoint.

  To reduce Query PrivateLink endpoint downtime on failovers or switchovers, configure your application by setting the value of the `connect_timeout` parameter to a small value (for example, 5 seconds) and by making it retry always upon connection failures.
- The Bulk Ingest Feature is not supported on the DB System.

Copyright © 2022, 2025, Oracle and/or its affiliates.

- Configuration changes for high availability DB Systems are not supported.
- High availability cannot be enabled or disabled on a DB System with PrivateLink(s) configured. Delete the PrivateLink(s), and then recreate them after enabling or disabling high availability.

See also the limitations described in Restoring a Backup of a High Availability DB System and Inbound Replication to a High Availability DB System.

12.10 Billing

A high availability DB system contains three MySQL instances, each of which utilizes the same amount of block volume storage, number of CPUs, and amount of RAM defined in the chosen shape.

You are charged for the three MySQL instances according to their shapes, storage sizes as well as the data transfer between Availability Zones incurred by Group Replication..

Copyright © 2022, 2025, Oracle and/or its affiliates.

13
MySQL HeatWave AutoML

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

- MySQL HeatWave AutoML Requirements
- Create a MySQL HeatWave AutoML model
- Evaluate a MySQL HeatWave AutoML model

13.1 MySQL HeatWave AutoML Requirements

MySQL HeatWave AutoML has the following requirements:

- An operational DB System with sufficient resources and a MySQL HeatWave Cluster. See the following:
  - Supported Shapes
  - Creating a DB System
  - Creating a MySQL 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 MySQL 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 MySQL HeatWave AutoML routines reside; for example:

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

13.2 Create a MySQL HeatWave AutoML model

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

Create a MySQL HeatWave AutoML model includes an advanced option to choose one or more algorithms. MySQL 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.

Copyright © 2022, 2025, Oracle and/or its affiliates.

1. Connect to the DB System. See Connecting from the Console.
2. Select the **Workspace** tab in the MySQL HeatWave Console, and then select the **Model Catalog** tab and click **Create Model**
3. In the **Create Machine Learning Model** pane that opens, provide the following information:
   - **Name**: Specify a name for the model or use the generated default name.
   - **Description**: Specify a user friendly description for the model.
   - **Training table**: Select a training dataset.
   - **Columns to include in the model**: Select the **Target column**, and then choose which columns to **Include in model**.
   - **Machine learning task**: Choose a task.
   - **Advanced**:
     - **Optimization metric**: Choose an optimization metric.
     - **Algorithms to evaluate**: Choose which algorithms MySQL HeatWave AutoML should consider to evaluate the model.
4. Click **Create**. The MySQL HeatWave Console returns to the **Model Catalog** tab, and shows the new model.

13.3 Evaluate a MySQL HeatWave AutoML model

Evaluate a MySQL 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 **Workspace** 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 the **Evaluate** tab, can click the **Evaluate Model** button.
6. In the **Select model** pane, select a model. The lower pane shows details for the model including **Training table**, **Target column**, **Description**, **Selected algorithm**, **Machine learning task**, and **Training score**.
7. Click **Evaluate**.
8. **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.
9. **Model score**:
   - Select one of the **Scoring metrics**, and click **Calculate score**. The **Results** shows the score.
10. **Explain model**:

Copyright © 2022, 2025, Oracle and/or its affiliates.

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

11. **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.

Copyright © 2022, 2025, Oracle and/or its affiliates.

14

MySQL HeatWave GenAI

MySQL HeatWave GenAI provides integrated and automated generative AI with in-database as well as external large language models (LLMs). It provides an automated, in-database vector store, and the ability to have contextual conversations in natural language, allowing you to take advantage of generative AI without AI expertise or data movement.

MySQL HeatWave GenAI in the MySQL HeatWave User Guide explains MySQL HeatWave GenAI in general details, and should be consulted when using MySQL HeatWave GenAI. The sections in this chapter explains how the MySQL HeatWave GenAI features are supported on MySQL HeatWave on AWS.

- MySQL HeatWave GenAI Requirements
- Supported LLMs, Embedding Models, and Languages
- MySQL HeatWave GenAI In-database Vector Store
- MySQL HeatWave GenAI Routines

14.1 MySQL HeatWave GenAI Requirements

Here are some requirements for using MySQL HeatWave GenAI on MySQL HeatWave on AWS

- A DB System that meets the following requirements:
  - The MySQL version is `9.0.1-u1` or higher.
  - A MySQL HeatWave Cluster has been added to the DB System, and the shape of the cluster must be HeatWave.256GB (which supports MySQL HeatWave Lakehouse).
  - MySQL HeatWave Lakehouse must be enabled on the DB System.
- **For Retrieval Augmented Generation**:
  - An Amazon S3 bucket in which the relevant documents are stored and loadable into MySQL HeatWave Lakehouse. See Creating Lakehouse Mapping.
  - A properly configured Lakehouse IAM role ARN for accessing the relevant documents in the Amazon S3 bucket. See Creating an IAM Role to Access an Amazon S3 Bucket.
- **For using external LLMs from Amazon Bedrock**:
  - Access to the external LLMs you want to use:
    - To manage model access, follow the instructions given on Add or remove access to Amazon Bedrock foundation models, starting with Step 2.
    - See the list of Amazon Bedrock LLMs supported by MySQL HeatWave on AWS in MySQL HeatWave GenAI External LLMs.
  - An IAM policy for accessing the Amazon Bedrock LLMs .
  - An IAM role for accessing the Amazon Bedrock LLMs.

See Managing a DB System and Managing a MySQL HeatWave Cluster on how to make adjustments to your DB System or MySQL HeatWave Cluster for using MySQL HeatWave GenAI.

Copyright © 2022, 2025, Oracle and/or its affiliates.

14.2 Supported LLMs, Embedding Models, and Languages

This section lists and explains the LLMs, embedding models, and languages supported by MySQL HeatWave on AWS.

- View Available Models
- MySQL HeatWave GenAI In-Database LLMs
- MySQL HeatWave GenAI External LLMs
- MySQL HeatWave GenAI In-databse Embedding Models
- Supported Languages

14.2.1 View Available Models

As of MySQL 9.3.2, once you are connected to a DB System, you can view the list of available models and information on them with the `ML_SUPPORTED_LLMS` view inside the `sys` schema of the DB System:

```
SELECT * FROM sys.ML_SUPPORTED_LLMS;
```

The output is similar to the following:

```
+---------------+------------------------------+------------------
+--------------------+--------------+
| provider       | model_id                     | availability_date |
capabilities        | default_model |
+---------------+------------------------------+------------------
+--------------------+--------------+
| HeatWave       | llama2-7b-v1                 | 2024-07-01        |
["GENERATION"]      |            0 |
| HeatWave       | llama3-8b-instruct-v1        | 2024-07-24        |
["GENERATION"]      |            0 |
| HeatWave       | llama3.1-8b-instruct-v1      | 2025-05-20        |
["GENERATION"]      |            0 |
| HeatWave       | llama3.2-1b-instruct-v1      | 2025-05-20        |
["GENERATION"]      |            0 |
| HeatWave       | llama3.2-3b-instruct-v1      | 2025-05-20        |
["GENERATION"]      |            1 |
| HeatWave       | mistral-7b-instruct-v1       | 2024-07-01        |
["GENERATION"]      |            0 |
| HeatWave       | mistral-7b-instruct-v3       | 2025-05-20        |
["GENERATION"]      |            0 |
| HeatWave       | all_minilm_l12_v2            | 2024-07-01        |
["TEXT_EMBEDDINGS"] |            0 |
| HeatWave       | multilingual-e5-small        | 2024-07-24        |
["TEXT_EMBEDDINGS"] |            1 |
| Amazon Bedrock | cohere.command-text-v14      | NULL              |
["GENERATION"]      |            0 |
| Amazon Bedrock | cohere.command-r-v1:0        | NULL              |
["GENERATION"]      |            0 |
| Amazon Bedrock | cohere.command-r-plus-v1:0   | NULL              |
```


Copyright © 2022, 2025, Oracle and/or its affiliates.

```
["GENERATION"]      |             0 |
| Amazon Bedrock | cohere.command-light-text-v14 | NULL              |
["GENERATION"]      |             0 |
| Amazon Bedrock | meta.llama3-8b-instruct-v1:0  | NULL              |
["GENERATION"]      |             0 |
| Amazon Bedrock | meta.llama3-70b-instruct-v1:0 | NULL              |
["GENERATION"]      |             0 |
+----------------+-------------------------------+-------------------
+--------------------+--------------+
10 rows in set (1.8985 sec)
```

> **Note**
>
> For the external LLMs provided by Amazon Bedrock to appear in the list, you need to satisfy the requirements **For using the external LLMs** listed in MySQL HeatWave GenAI Requirements. Only the LLMs supported in your AWS region are listed.

14.2.2 MySQL HeatWave GenAI In-Database LLMs

MySQL HeatWave GenAI supports LLMs hosted and executed within the database. When in-database LLMs are used, data does not leave your MySQL HeatWave Cluster, which enhances the security for your data.

For MySQL 9.3.2 and later, see View Available Models on how to view a list of the supported in-database LLMs.

For MySQL 9.3.1 and earlier, the following are the in-database LLMs currently supported by MySQL HeatWave on AWS:

- Supported since MySQL 9.0.1:
  - llama2-7b-v1: supports English (en).
  - llama3-8b-instruct-v1: supports English (en), French (fr), German (de), Hindi (hi), Italian (it), Portuguese (pt), Spanish (es), and Thai (th).
  - mistral-7b-instruct-v1: supports English (en), French (fr), German (de), and Spanish (es).
- Supported since MySQL 9.3.1:
  - llama3.1-8b-instruct-v1: supports English (en), French (fr), German (de), Hindi (hi), Italian (it), Portuguese (pt), Spanish (es), and Thai (th).
  - mistral-7b-instruct-v3: supports English (en), French (fr), German (de), and Spanish (es).
  - llama3.2-3b-instruct-v1: supports English (en), French (fr), German (de), Hindi (hi), Italian (it), Portuguese (pt), Spanish (es), and Thai (th). This is the default LLM used by MySQL HeatWave GenAI for content generation.
  - llama3.2-1b-instruct-v1: supports English (en), French (fr), German (de), Hindi (hi), Italian (it), Portuguese (pt), Spanish (es), and Thai (th).

Use these models with the MySQL HeatWave GenAI routines.

Copyright © 2022, 2025, Oracle and/or its affiliates.

14.2.3 MySQL HeatWave GenAI External LLMs

MySQL HeatWave GenAI supports external LLMs provided by Amazon Bedrock. When external LLMs are used, data leaves your MySQL HeatWave Cluster to be processed by Amazon Bedrock.

For MySQL 9.3.2 and later, see View Available Models on how to view a list of the supported external LLMs.

For MySQL 9.3.1 and earlier, the following are the external LLMs provided by Amazon Bedrock that are currently supported by MySQL HeatWave on AWS:

- Supported since MySQL 9.0.1:
  - cohere.command-text-v14
  - cohere.command-light-text-v14
  - meta.llama3-1-70b-instruct-v1:0
- Supported since MySQL 9.2.1:
  - cohere.command-r-v1:0
  - cohere.command-r-plus-v1:0
  - meta.llama3-70b-instruct-v1:0
  - meta.llama3-1-405b-instruct-v1:0

See Amazon Bedrock Documentation for the list of languages supported by each of the LLMs.

> **Note**
>
> Not all LLMs provided by Amazon Bedrock are available in all AWS regions. Also, MySQL HeatWave GenAI does not support cross-region inference, which means an external LLM provided by Amazon Bedrock is not supported by MySQL HeatWave GenAI in your AWS region if it is only accessible through cross-region inference in that region. See Supported foundation models in Amazon Bedrock for details.

To use these models with the MySQL HeatWave GenAI routines, you need:

- An IAM policy to access the Amazon Bedrock service
- An IAM Role to access the Amazon Bedrock service

14.2.4 MySQL HeatWave GenAI In-databse Embedding Models

For MySQL 9.3.2 and later, see View Available Models on how to view a list of the supported in-database embedding models.

For MySQL 9.3.1 and earlier, MySQL HeatWave GenAI supports the following in-database embedding models:

- `multilingual-e5-small`
  - As of MySQL 9.3.0, MySQL HeatWave GenAI uses this embedding model by default for encoding text or documents in all supported languages (in earlier versions of MySQL, MySQL HeatWave GenAI uses this embedding model by default for encoding text or documents in all supported languages *other than English*).

Copyright © 2022, 2025, Oracle and/or its affiliates.

  - As of MySQL 9.2.1, can be used for encoding text or documents in all supported languages (in earlier versions of MySQL, can be used for encoding text or documents in all supported languages *other than English*).
  - This embedding model is available as of MySQL 9.0.1.
- `all_minilm_l12_v2`:
  - In MySQL 9.2.2 and earlier versions, MySQL HeatWave GenAI uses this embedding model by default for encoding English text or documents.
  - As of MySQL 9.2.1, can be used for encoding text or documents in any supported language
  - In earlier versions of MySQL, can be used for encoding text or documents *in English only*.

Use these models with ML_EMBED_ROW and ML_EMBED_TABLE (see MySQL HeatWave GenAI routines).

14.2.5 Supported Languages

As of MySQL 9.0.1-u1, MySQL HeatWave GenAI supports natural-language communication, ingesting documents, as well as generating text-based content in multiple languages. The quality of the generated text outputs depends on the training and ability of the LLM to work with the language.

As of MySQL 9.0.0, MySQL HeatWave GenAI supports natural-language communication, ingesting documents, as well as generating text-based content in English (en).

The following is a combined list of languages supported by MySQL HeatWave GenAI (not all LLMs support all the languages; to learn which LLM supports which language, see MySQL HeatWave GenAI In-Database LLMs and Amazon Bedrock Documentation for details).:

- Arabic (ar)
- Bengali (bn)
- Burmese (my)
- Chinese (zh)
- Czech (cs)
- Dutch (nl)
- English (en)
- French (fr)
- German (de)
- Hebrew (he)
- Hindi (hi)
- Indonesian (id)
- Italian (it)
- Japanese (ja)
- Khmer (km)
- Korean (ko)
- Lao (lo)

Copyright © 2022, 2025, Oracle and/or its affiliates.

- Malay (ms)
- Persian (fa)
- Polish (pl)
- Portuguese (pt)
- Spanish (es)
- Tagalog (tl)
- Thai (th)
- Turkish (tr)
- Urdu (ur)
- Vietnamese (vi)

14.3 MySQL HeatWave GenAI In-database Vector Store

The automated, in-database vector store enables customers to use MySQL HeatWave GenAI with their business documents without moving data to a separate vector database. All the steps to create a vector store and vector embeddings are automated and executed inside the database, including discovering the documents in object storage, parsing them, generating embeddings in a highly parallel and optimized way, and inserting them into the vector store, making the HeatWave vector store efficient and easy to use. When in-database LLMs are used, data does not leave your MySQL HeatWave Cluster, thus enhancing the security for your data. The in-database vector store enables the use of Retrieval Augmented Generation (RAG), allowing the LLMs to search proprietary data with appropriate context to provide more accurate and relevant answers.

To generate embedding for a file stored in your Amazon S3 bucket, connect to your DB System, create a database, and use the heatwave_load() stored procedure routine to load the file. For example:

```
drop database if exists vector_store;
create database vector_store;
use vector_store;
DROP TABLE IF EXISTS vector_test;
SET @dl_tables = '[{
  "db_name": "vector_store",
  "tables": [{
     "table_name": "vector_test",
     "dialect": {
       "format":"pdf"
     },
     "file": [{"prefix": "s3://<bucket-name>/<prefix>/<filename>"}]
   }]
 }]';
SET @options = JSON_OBJECT('policy', 'disable_unsupported_columns',
      'external_tables',
      CAST(@dl_tables AS JSON));
CALL sys.heatwave_load(@dl_tables, @options);
select count(*) from vector_test;
```

When `format` of the loaded file is one of `pdf`, `txt`, `html`, `json`, `doc`, or `ppt`, MySQL HeatWave Lakehouse automatically creates a vector store out of the database you created, parses the

Copyright © 2022, 2025, Oracle and/or its affiliates.

loaded file, and generates an embedding for the file in the table you created. The vector store can now be used with the `ML_RAG` and `ML_RAG_TABLE` routines; see MySQL HeatWave GenAI Routines.

You can learn more about the MySQL HeatWave GenAI in-database vector store by looking at Performing a Vector Search, but note that the VECTOR_STORE_LOAD routine, which operates asynchronously, is not supported by MySQL HeatWave on AWS; use heatwave_load() stored procedure instead for loading a file.

14.4 MySQL HeatWave GenAI Routines

The MySQL HeatWave GenAI Routines are stored procedures in MySQL that provides GenAI functionalities. The MySQL HeatWave GenAI Routines supported by MySQL HeatWave on AWS are listed below. See the full descriptions of these routines and how to use them in the HeatWave User Guide by following the links below:

- ML_GENERATE. When using Amazon Bedrock LLMs, only the task type `generation` is supported; `summarization` is not supported.
- ML_GENERATE_TABLE. When using Amazon Bedrock LLMs, only the task type `generation` is supported; `summarization` is not supported.
- ML_RAG. You need to have a MySQL HeatWave GenAI In-database Vector Store to use the routine.
- ML_RAG_TABLE. You need to have a MySQL HeatWave GenAI In-database Vector Store to use the routine.
- HEATWAVE_CHAT.
- ML_EMBED_ROW.
- ML_EMBED_TABLE.

Copyright © 2022, 2025, Oracle and/or its affiliates.

15

System Variables

- System Variables

15.1 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.

Copyright © 2022, 2025, Oracle and/or its affiliates.

16 Events

Statuses of actions and operations on MySQL HeatWave on AWS resources are reported as Events.

A list of all Events is found on the **Service Events** pane, accessible from the console's **Home** page by the following means:

- The **Monitoring** tab
- The **View all ...** link in the **Recent Activity** region

The list of Events can be filtered by **Severity** or **Type**.

Events are also listed in the operation and resource **Details** pages. See Table 16-1.

Operations and Resources Monitored

Table 16-1 shows the operations and resources on MySQL HeatWave on AWS for the events, the types of events that are reported, and the locations on MySQL HeatWave Console where the **Events** tab (which shows the event reports) can be found.

Table 16-1 Events Reported on the MySQL HeatWave Console

Resources and Operations	Types of Actions or Operations that Generate Event Reports	Location of Events Tab
DB System	Create, delete, resize, start, stop, restart, update, upgrade, configuration update, shape update	MySQL DB System Details
MySQL HeatWave Clusters	Create, delete, start, stop, restart, enabling and disabling of Lakehouse plugin,	MySQL HeatWave Cluster Details
Inbound Replication Channel	Create, delete, update, resume, reset channel	Channel Details
Backups	Create, update, delete	Backup Details
MySQL Configurations	Create, update, delete	MySQL Configuration Details

Information in an Event Report

The following information is available in each event:

- **Type**: The event and its status (**In Progress**, **Succeeded**, **Failed**). For example, `Backup Delete Succeeded`, `Channel Create In Progress`.
- **Severity**: The severity level of the event, which can be one of
  - `INFO`: General information is provided. No special concerns
  - `WARNING`: Warning on an event that might impact the DB System's operation if no actions are taken.
  - `CRITICAL`: A critical event. An urgent action is required to avoid adverse impact on your DB System.

Copyright © 2022, 2025, Oracle and/or its affiliates.

- **Message**: The message body of the event report. It usually contains the information in **Type** plus the ID of the resource involved. For example, `DB System e7fe1776-0066-4508-a904-ccb87b62ed46 stop in progress`.
- **Created**: The date and time the event report was created.

The following additional information is available for each event in the **Event Details** window, which opens when you click on the event in the **Events** tab:

- **Event ID**: A unique ID of an event.
- The type of resource involved and its resource ID. For example, **MySQL Configuration** followed by **Default MySQL Configuration for MySQL.32.256GB (Supports MySQL HeatWave)** .

Copyright © 2022, 2025, Oracle and/or its affiliates.

17 Monitoring

This chapter discusses a number of features MySQL HeatWave on AWS provides for monitoring your DB Systems and MySQL HeatWave Clusters.

- CloudWatch Monitoring
- Performance Tabs
- Autopilot Shape Advisor

17.1 CloudWatch Monitoring

MySQL HeatWave on AWS supports AWS CloudWatch for detailed monitoring of your DB Systems and MySQL HeatWave Clusters.

These are the prerequisites for using the feature:

- An AWS account for receiving the metrics emitted by the monitoring service.
- Ability to create an IAM role in your AWS account.

Follow these steps to set up or reconfigure CloudWatch monitoring for your DB System (requires logging in to your AWS account):

1. Create an AWS IAM role and policy in your AWS account for CloudWatch Monitoring. See instructions here.
2. Enable CloudWatch monitoring for your DB System by one of the following ways:
   - Setting the **Monitoring configuration** when Creating a DB System.
   - Enabling CloudWatch monitoring for an existing DB System by going to the DB System Details page and select under the **Actions** menu **Update Monitoring Configuration**. Follow then Step 3 in Update Monitoring Configuration.

Verify the list of CloudWatch monitoring metrics you have set up by checking the **Monitoring Configuration** tab of your DB System (see MySQL DB System Details).

To view the metrics in your AWS CloudWatch account:

- Go to the **Metrics** tabs of your DB System (see Viewing DB System Details) and MySQL HeatWave Cluster (see Viewing MySQL HeatWave Cluster Details). Under **CloudWatch Metrics**, follow the link there to your AWS CloudWatch console.

  > **Note**
  >
  > If CloudWatch monitoring has not been enabled for the DB System, you can enable it by clicking **Update Monitoring Configuration** under **CloudWatch Metrics**, and then follow Step 3 in Update Monitoring Configuration.

- In the AWS CloudWatch console, choose on the left sidebar **Metrics > All metrics**. On the **Browse** tab and under **Custom namespaces**, choose the **OracleHeatWave** namespace and select the ID of the DB System of interest.

Copyright © 2022, 2025, Oracle and/or its affiliates.

You can also create alarms and dashboards in your AWS account for your metrics. See **Related Topics** below.

> **Note**
>
> For high availability DB Systems, metrics are only emitted for the primary instance. On a failover, the new primary is configured to start sending metrics, and the old primary is configured to stop sending metrics. During this time of reconfiguration, the metrics emitted may be inconsistent.

> **Note**
>
> During maintenance, the metric emission might be interrupted.

- CloudWatch Monitoring Metrics
- Cost for Using CloudWatch Monitoring

Related Topics

- Using Amazon CloudWatch alarms
- Using Amazon CloudWatch dashboards
- Update Monitoring Configuration

17.1.1 CloudWatch Monitoring Metrics

CloudWatch metrics can be used to monitor resources, diagnose issues, and troubleshoot your DB systems and their MySQL HeatWave Clusters. After enabling CloudWatch monitoring on a DB System, you can select the set of metrics that will be emitted to CloudWatch for the DB System and its MySQL HeatWave Cluster.

The metrics are emitted using the CloudWatch embedded metric format (EMF) logs. The EMF logs are JSON objects that embed metric data along with metadata and contextual information. You can find more information about the EMF format in EMF specification by AWS. Here is a sample EMF log snippet:

```
{
    "_aws": {
        "CloudWatchMetrics": [
            {
                "Namespace": "OracleHeatWave",
                "Dimensions": [
                    [
                        "dbSystemId"
                    ]
                ],
                "Metrics": [
                    {
                        "Name": "mysql.stats.threads_connected",
                        "Unit": "Count",
                        "StorageResolution": 60
                    }
```


Copyright © 2022, 2025, Oracle and/or its affiliates.

```
                ]
            }
        ],
        "Timestamp": 1750074660311
    },
    "dbSystemId": "example-dbsystem-id",
    "mysql.stats.threads_connected": 1
}
```

Here are explanations for some of the objects in an EMF log:

- **Namespace:** Metrics are emitted to the **OracleHeatWave** namespace in CloudWatch. The EMF logs are streamed to the `/OracleHeatWave/metrics` log group.
- **Dimensions**:
  - **dbSystemId**: Specifies the unique ID of the DB System.
  - **resourceType**: Specifies the resource type the metric originated from. It can be one of:
    - `mysql`: DB System
    - `heatWave`: MySQL HeatWave Cluster
  - **heatWaveNode**: Specifies the MySQL HeatWave nodeindex the metric originated from.
- **Metrics**: Table 17-1 list the metrics for monitoring your DB Systems and MySQL HeatWave Clusters.

Table 17-1 CloudWatch Metrics for MySQL HeatWave on AWS

Metric	Unit	Description	Dimensions	Monitoring Configuration Templates the Metric Is In
heatwave.auto_ml_operation_count	Count	The number of AutoML operations that have been executed. This number resets on MySQL HeatWave restart.	– dbSystemId	– Detailed
heatwave.change_propagation_lag	Count	The number of transactions that are not propagated to MySQL HeatWave yet.	– dbSystemId	– Detailed
heatwave.dataset_bytes	Bytes	The amount of data loaded in the MySQL HeatWave cluster.	– dbSystemId	– Detailed
heatwave.heap_usage	Bytes	The size of the MySQL HeatWave data dictionary, which consumes heap space on the DB system nodes.	– dbSystemId	– Detailed
heatwave.lakehouse_total_loaded_bytes	Bytes	The total size of all Lakehouse tables loaded into MySQL HeatWave.	– dbSystemId	– Detailed

Copyright © 2022, 2025, Oracle and/or its affiliates.

Table 17-1 (Cont.) CloudWatch Metrics for MySQL HeatWave on AWS

Metric	Unit	Description	Dimensions	Monitoring Configuration Templates the Metric Is In
heatwave.loaded_tables_available	Count	The total size of all Lakehouse tables loaded into MySQL HeatWave.	– dbSystemId	– Detailed
heatwave.loaded_tables_unusable	Count	The number of tables loaded into MySQL HeatWave that are currently not available for use.	– dbSystemId	– Detailed
heatwave.health	Count	MySQL HeatWave Cluster health status.	– dbSystemId	– Basic – Detailed
heatwave.load_progress	Percent	Progress of data load intoMySQL HeatWave Cluster memory.	– dbSystemId	– Basic – Detailed
heatwave.query_offload_count	Count	The number of statements executed against the DB System and were executed on MySQL HeatWave Cluster.	– dbSystemId	– Basic – Detailed
mysql.inno.buffer_pool.pages_dirty	Count	Buffer pages currently dirty.	– dbSystemId	– Detailed
mysql.inno.buffer_pool.pages_free	Count	Buffer pages currently free.	– dbSystemId	– Detailed
mysql.inno.buffer_pool.pages_total	Count	Total buffer pool size in pages.	– dbSystemId	– Detailed
mysql.stats.aborted_clients	Count	The number of connections that were aborted because the client died without closing the connection properly.	– dbSystemId	– Detailed
mysql.stats.aborted_connects	Count	The number of failed attempts to connect to the MySQL server.	– dbSystemId	– Detailed
mysql.stats.connection.total	Count	Cumulative count of total connections created.	– dbSystemId	– Detailed
mysql.stats.created.tmp_disk_tables	Count	The number of internal on-disk temporary tables created by the server while executing statements.	– dbSystemId	– Detailed
mysql.stats.created.tmp_tables	Count	The number of internal temporary tables created by the server while executing statements.	– dbSystemId	– Detailed
mysql.stats.handler.delete	Count	The number of times that rows have been deleted from tables.	– dbSystemId	– Detailed

Copyright © 2022, 2025, Oracle and/or its affiliates.

Table 17-1 (Cont.) CloudWatch Metrics for MySQL HeatWave on AWS

Metric	Unit	Description	Dimensions	Monitoring Configuration Templates the Metric Is In
mysql.stats.handler.read_first	Count	The number of times the first entry in an index was read.	– dbSystemId	– Detailed
mysql.stats.handler.read_key	Count	The number of requests to read a row based on a key.	– dbSystemId	– Detailed
mysql.stats.handler.read_last	Count	The number of requests to read the last key in an index.	– dbSystemId	– Detailed
mysql.stats.handler.read_next	Count	The number of requests to read the next row in key order.	– dbSystemId	– Detailed
mysql.stats.handler.read_prev	Count	The number of requests to read the previous row in key order.	– dbSystemId	– Detailed
mysql.stats.handler.read_rnd	Count	The number of requests to read a row based on a fixed position.	– dbSystemId	– Detailed
mysql.stats.handler.read_rnd_next	Count	The number of requests to read the next row in the data file.	– dbSystemId	– Detailed
mysql.stats.handler.update	Count	The number of requests to update a row in a table.	– dbSystemId	– Detailed
mysql.stats.handler.write	Count	The number of requests to insert a row in a table.	– dbSystemId	– Detailed
mysql.stats.select_full_join	Count	The number of joins that perform table scans because they do not use indexes.	– dbSystemId	– Detailed
mysql.stats.select_full_range_join	Count	The number of joins that used a range search on a reference table.	– dbSystemId	– Detailed
mysql.stats.select_range	Count	The number of joins that used ranges on the first table.	– dbSystemId	– Detailed
mysql.stats.select_range_check	Count	The number of joins without keys that check for key usage after each row.	– dbSystemId	– Detailed
mysql.stats.select_scan	Count	The number of joins that did a full scan of the first table.	– dbSystemId	– Detailed
mysql.stats.sort_merge_passes	Count	The number of merge passes that the sort algorithm has had to do.	– dbSystemId	– Detailed
mysql.stats.sort_range	Count	The number of sorts that were done using ranges.	– dbSystemId	– Detailed

Copyright © 2022, 2025, Oracle and/or its affiliates.

Table 17-1 (Cont.) CloudWatch Metrics for MySQL HeatWave on AWS

Metric	Unit	Description	Dimensions	Monitoring Configuration Templates the Metric Is In
mysql.stats.sort_rows	Count	The number of sorted rows.	– dbSystemId	– Detailed
mysql.stats.sort_scan	Count	The number of sorts that were done by scanning the table.	– dbSystemId	– Detailed
mysql.stats.threads_connected	Count	The number of current connections to the DB system.	– dbSystemId	– Basic – Detailed
mysql.stats.threads_running	Count	The number of connections actively executing statements against the DB system.	– dbSystemId	– Basic – Detailed
system.cpu.utilization	Percent	The CPU utilization for the DB system host.	– dbSystemId – resourceType	– Basic – Detailed
system.dbvolume.bytes_read	Bytes	The number of bytes read from the DB System volume during the period.	– dbSystemId – resourceType	
system.dbvolume.bytes_write	Bytes	The number of bytes written to the DB System volume during the period.	– dbSystemId – resourceType	
system.dbvolume.read_operations	Count	The number of read operations for the DB volume during the period.	– dbSystemId – resourceType	
system.dbvolume.size	Bytes	The maximum amount of space allocated to the DB System during the interval.	– dbSystemId – resourceType	– Basic – Detailed
system.dbvolume.usage	Bytes	The maximum amount of space used during the interval.	– dbSystemId – resourceType	– Basic – Detailed
system.dbvolume.utilization	Percent	The total space utilization of the DB System volume.	– dbSystemId – resourceType	– Basic – Detailed
system.dbvolume.write_operations	Count	The number of write operations for the DB volume during the period.	– dbSystemId – resourceType	
system.memory.utilization	Percent	Memory utilization for the DB System host.	– dbSystemId – resourceType – heatWaveNode	– Basic – Detailed
system.memory.size	Bytes	The total amount of memory allocated during the selected interval.	– dbSystemId – resourceType – heatWaveNode	– Basic – Detailed

Copyright © 2022, 2025, Oracle and/or its affiliates.

Table 17-1 (Cont.) CloudWatch Metrics for MySQL HeatWave on AWS

Metric	Unit	Description	Dimensions	Monitoring Configuration Templates the Metric Is In
system.memory.usage	Bytes	The maximum amount of memory used during the selected interval.	– dbSystemId – resourceType – heatWaveNode	– Basic – Detailed
system.network.bytes_received	Bytes	The number of bytes received by the DB system through the network during the period.	– dbSystemId – resourceType	
system.network.bytes_sent	Bytes	The number of bytes transmitted for the DB system through the network during the period.	– dbSystemId – resourceType	

How metrics are sent:

- Metrics are emitted every 60 seconds.
- During maintenance or actions against the DB System (for example, restart, update), metric emissions might be interrupted.
- Metrics for highly available DB Systems are only emitted for the primary instance; secondary instances do not emit metrics. In case of a loss of quorum when both secondaries are unreachable (for example, due to network partitioning), no metrics will be sent until majority is restored for the DB System.

Related Topics

- Update Monitoring Configuration

17.1.2 Cost for Using CloudWatch Monitoring

There are no surcharges for using CloudWatch monitoring in MySQL HeatWave on AWS. However, enabling CloudWatch monitoring will incur charges on your AWS account. Metrics are emitted using CloudWatch logs in the embedded metric format (EMF), for which costs are incurred. **Users should consult the Amazon CloudWatch Pricing for details on CloudWatch-related charges**. Here is a breakdown of the charges to your AWS account incurred by enabling CloudWatch monitoring on MySQL HeatWave on AWS:

- Log ingestion costs: Metrics are sent to your AWS CloudWatch account as log events in EMF format, under the `/OracleHeatWave/metrics` log group.
- Log archival costs: By default, the retention period on the `/OracleHeatWave/metrics` log group is 30 days. You can change the retention period for this log group in your AWS account.
- Metric costs: Each metric incurs charges in Amazon CloudWatch. Metric charges are prorated by the hour and metered only when metrics are sent to CloudWatch.

Copyright © 2022, 2025, Oracle and/or its affiliates.

> **Note**
>
> For metrics with multiple dimensions, CloudWatch counts each unique combination of dimensions as a separate metric and charged it as so. For example if you enable monitoring on a DB System with an attached MySQL HeatWave Cluster and choose `system.memory.usage` as a metric, you will be paying for *MySQL HeatWave Cluster size* + 1 metrics in CloudWatch, because the chosen metric will monitor the memory usage on the associated DB System as well as all MySQL HeatWave nodes in your cluster.

17.2 Performance Tabs

The **Workspace** tab in the MySQL HeatWave Console has a **Performance** group of tabs, which provides **Metrics** for MySQL and MySQL HeatWave performance and **HeatWave Workload** data. An active MySQL HeatWave Cluster is required for those data to be available.

- Metrics for MySQL and MySQL HeatWave Cluster Performance
- Workload Performance

17.2.1 Metrics for MySQL and MySQL HeatWave Cluster Performance

To view the **Metrics** for MySQL and MySQL HeatWave Cluster:

1. Connect to the DB System. See Connecting from the Console.
2. In the MySQL HeatWave Console, select the **Workspace** tab.
3. The **Metrics** tab shows performance data for the DB System and its MySQL HeatWave Cluster.

   Click the **Refresh** icon to refresh the performance data.

For descriptions of the **Metrics**, see Details for the MySQL and MySQL HeatWave Cluster Performance Metrics.

- Details for the MySQL and MySQL HeatWave Cluster Performance Metrics

17.2.1.1 Details for the MySQL and MySQL HeatWave Cluster Performance Metrics

The **Metrics** tab provides the following performance data for your DB System and MySQL HeatWave Cluster:

- **CloudWatch metrics**

  It shows if CloudWatch Monitoring has been enabled and, if it is, provides a link to **View CloudWatch metrics** with your AWS account.

- **HeatWave**
  - **Dataset Size**

    The size of the dataset loaded on the MySQL HeatWave Cluster.
  - **Data Dictionary Size**

    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*.
  - **Cluster Memory Utilization**

Copyright © 2022, 2025, Oracle and/or its affiliates.

The percentage of total MySQL 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**.

- **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 Utilization**

    The percentage of disk space used 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.

17.2.2 Workload Performance

To view MySQL HeatWave workload performance data:

1. Connect to the DB System. See Connecting from the Console.
2. In the MySQL HeatWave Console, select the **Workspace** tab.
3. The **HeatWave Workload** tab shows workload performance data for MySQL HeatWave and for the MySQL node.

   Click the **Refresh** icon to refresh the performance data.

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

- MySQL HeatWave Workload Performance Data

17.2.2.1 MySQL HeatWave Workload Performance Data

The **HeatWave Workload** tab provides the following performance data for your MySQL HeatWave Clusters:

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.

Copyright © 2022, 2025, Oracle and/or its affiliates.

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.

17.3 Autopilot Shape Advisor

Use the Auto Shape Prediction feature in MySQL HeatWave Autopilot for MySQL HeatWave on AWS to analyze the workload and assess the suitability of the current MySQL shape.

> **Note**
>
> The Auto Shape Prediction feature is available prior to MySQL 8.0.32, but is enabled by default only for MySQL 8.0.32 and later.

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 activities in a five-minute interval, or if buffer pool usage is growing, Auto Shape Prediction cannot make a prediction for that interval.

The Auto Shape Prediction looks at buffer pool usage, workload activity, access patterns, and CPU statistics to estimate the required buffer pool (for MySQL version 8.0.32 and later) and CPU sizes (for MySQL 8.2.0-u2 and later). The system administrator should then make sure the DB system shape can accommodate the recommended buffer pool size and number of CPU cores. If the current buffer pool or CPU size is smaller than the predicted requirements, although MySQL can usually manage the workload while keeping the system stable, the performance of the DB System might suffer due to elevated disk I/O or insufficient CPU cores for processing the workload.

> **Tip**
>
> **For DB systems using MySQL versions earlier than 8.2.0-u2**: 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 of the collected feature data for 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 past predictions.

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

For MySQL 8.0.32 and later, there are two ways to access the Autopilot Shape Advisor: with the MySQL HeatWave Console or with a MySQL client. Prior to MySQL 8.0.32, use a MySQL client.

Copyright © 2022, 2025, Oracle and/or its affiliates.

> **Note**
>
> The SQL output by a MySQL client can only provide some hints. The MySQL HeatWave Console can provide detailed and graphical information, and can recommend an improved shape by its specific shape name. See: Autopilot Shape Advisor with MySQL HeatWave Console .

- Autopilot Shape Advisor with MySQL HeatWave Console
- Autopilot Shape Advisor with a MySQL Client

17.3.1 Autopilot Shape Advisor with MySQL HeatWave Console

To view the Autopilot Shape Advisor:

1. Connect to the DB System of interest. See Connecting from the Console.
2. In the MySQL HeatWave Console, click the **Workspace** tab, and then click the **Shape Advisor** tab. It shows the Auto Shape Prediction data for the DB System.

   Click the **Refresh** icon beside the selected DB system to refresh the performance data.
3. To see performance data for another DB System, click the **Disconnect** button on the sidebar and connect to another DB System.

For descriptions of Auto Shape Prediction data, see:

- Auto Shape Prediction Data

17.3.1.1 Auto Shape Prediction Data

The **Autopilot Shape Advisor** tab shows data for the last seven days and contains the following information:

- **Summary**

  This summarizes the basic information of your DB system and MySQL HeatWave Cluster including their names, their states, and the shape and size of the cluster.
- **Recommendation**

  This section contains the **MySQL Shape Prediction** and the recommended **Action**. You need to select a time period in the drop-down list on the right, upon which the prediction will be based. The choices in the drop-down list include:

  – **Past 12 hours**

  – **Past day**

  – **Past 3 days**

  – **Past week**

  The time period can also be chosen by using the horizontal slider controls at the bottom of the page.

  MySQL Shape Prediction

  – **Current**

    The current shape size, with **Shape Name**, **Memory Size**, **CPU Count**, and **Buffer Pool Size**.

Copyright © 2022, 2025, Oracle and/or its affiliates.

- **Recommended**

  The recommended shape size, with **Shape Name**, **Memory Size**, **CPU Count**, and **Buffer Pool Size**.

Action

The prediction is based on the selected time period, and there are five possible cases:

- **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.

After the recommendation, **Steps to take** provides the instructions to follow for adopting the recommended shape.

- **Shape Prediction Insights**

  This sections contains the following information:

  - **Average Statistics**
    - **Buffer Pool**

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

      A percentage value calculated for the selected time period.
    - **CPU Utilization**

      Average CPU utilization calculated for the selected time period.
  - **Recent Statistics**

    Use the left and right slider controls to adjust the time period for these time graphs:
    - **Buffer Pool**

      A detailed graph that shows the **Buffer Pool Size** and **Recommended Buffer Pool Size** for the selected time period.
    - **CPU Count**

      A detailed graph that shows the CPU count predictions for downsize cases for the selected time period.
    - **Buffer Pool Hit Rate**

      A detailed graph that shows the **Buffer Pool Hit Rate** for the selected time period.
    - **CPU Utilization**

      A detailed graph that shows the history of **CPU Utilization** for the selected time period.

    Workload markers on the time series graphs indicate recent activities:
    - Green markers: The workload is running and stable. This allows good predictions.

Copyright © 2022, 2025, Oracle and/or its affiliates.

* Orange markers: The workload is running, but is not stable. This makes predictions difficult.
* No markers: No recent activities, and no predictions.

17.3.2 Autopilot Shape Advisor with a MySQL Client

Auto Shape Prediction records predictions in the `shape_predictions` table in the `mysql_autopilot` schema on the MySQL server. 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 some hints. The MySQL HeatWave Console provides 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 45 minutes:

```
mysql> SELECT * FROM mysql_autopilot.shape_predictions LIMIT 10;
+----------------------------+------------+-------------+---------------
+----------------+---------------------+-------------------------
+-----------------------------------+
| prediction_time            | bp_size_gb | bp_hit_rate | cpu_core_count |
cpu_utilization | predicted_bp_size_gb | predicted_cpu_core_count |
prediction_outcome                |
+----------------------------+------------+-------------+---------------
+----------------+---------------------+-------------------------
+-----------------------------------+
| 2023-12-11 14:34:43.566972 |         43 |        NULL |              8
|       0.0188317 |                NULL |                     NULL | EMPTY
FEATURE DATA                |
| 2023-12-11 14:39:43.630568 |         43 |        NULL |              8
|        0.809462 |                NULL |                     NULL | NOT
ENOUGH FEATURE DATA SNAPSHOTS |
| 2023-12-11 14:44:43.682870 |         43 |           1 |              8
|        0.917938 |                NULL |                     NULL | FEATURE
DATA IS NOT STABLE        |
| 2023-12-11 14:49:43.691465 |         43 |           1 |              8
|        0.517613 |                NULL |                     NULL | FEATURE
DATA IS NOT STABLE        |
| 2023-12-11 14:54:43.707952 |         43 |       0.875 |              8
|       0.0378884 |                NULL |                     NULL | FEATURE
DATA IS NOT STABLE        |
| 2023-12-11 14:59:43.724345 |         43 |        NULL |              8
|      0.00403875 |                NULL |                     NULL | LOW
ACTIVITY                    |
| 2023-12-11 15:04:43.734345 |         43 |           1 |              8
|        0.120723 |                12.3 |                     4    |
DOWNSIZE                   |
| 2023-12-11 15:09:43.744345 |         43 |           1 |              8
|        0.110723 |                12.5 |                     4    |
```


Copyright © 2022, 2025, Oracle and/or its affiliates.

```
DOWNSIZE                          |
| 2023-12-11 15:14:43.754345 |         43 |           1 |              8
|        0.120723 |                 12.4 |                       4    |
DOWNSIZE                          |
| 2023-12-11 15:19:43.764345 |         43 |           1 |              8
|        0.130723 |                 12.2 |                       4    |
DOWNSIZE                          |
+---------------------------+------------+-------------+----------------
+-----------------+---------------------+-------------------------
+-----------------------------------+
```

The `shape_predictions` table has these columns:

- `prediction_time`: The timestamp for the prediction. A prediction is attempted every five minutes.
- `bp_size_gb`: The current buffer pool size in GB.
- `bp_hit_rate`: The current buffer pool hit rate.
- `cpu_core_count`: The number of current available CPU cores.
- `cpu_utilization`: The CPU utilization (on the scale of 0 to 1, with 1 meaning full utilization).
- `predicted_bp_size_gb`: The predicted buffer pool size for optimal performance with this workload in GB.
- `predicted_cpu_core_count`: The predicted CPU core count.
- `prediction_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 with a MySQL client, follow these steps:

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

   ```
   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 a statement like the following on 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 predictions have stabilized.
4. When the predictions have stabilized, make a note of the maximum values of `predicted_bp_size_gb` and `predicted_cpu_core_count`, and the associated actions suggested in `prediction_outcome`.

Copyright © 2022, 2025, Oracle and/or its affiliates.

5. 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.

Copyright © 2022, 2025, Oracle and/or its affiliates.

18 Backups

This section describes how to create, restore, and manage MySQL DB System backups.

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
- Editing a Backup
- Viewing Backup Details
- Restoring a Backup to a New DB System
- Deleting a Backup
- Billing and Free Quota for Backup Storage

18.1 Creating a Backup

Backups can be created automatically or manually.

For how to enable automatic backups for a DB system, see Creating a DB System and Editing a DB System.

Use the Console to create a manual backup:

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

Copyright © 2022, 2025, Oracle and/or its affiliates.

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**.

18.2 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 **Resources** 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.

18.3 Viewing Backup Details

To view backup details:

1. In the MySQL HeatWave Console, select the **Resources** 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, which contains the following tabs.

18.3.1 Backup Details

Use the Console to view the **Backup Details** page, which contains the following tabs.

Table 18-1 Tabs on the Backup Details Page

Name	Description
Details	Backup Details. See Table 18-2.
Backup Event Details	Backup event details. See Table 18-3.

Copyright © 2022, 2025, Oracle and/or its affiliates.

Table 18-2 Backup Details

Field	Description
General information	• **Name**: The name of the backup. • **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. • **Description**: A description of the backup. • **Last Updated**: The date and time the backup was last updated. • **Created**: The date and time the backup was created.
Backup configuration	• **Storage size**: The size of the backup. • **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.
DB System snapshot	• **Resource ID**: The unique resource identifier of the DB System used to create the backup. • **Name**: The name of the DB System. • **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. • **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.

Copyright © 2022, 2025, Oracle and/or its affiliates.

Table 18-3 Backup Event Details

Field	Description
Type	The event and its status.
Severity	The severity level of the event, which can be one of • `INFO`: General information is provided. No special concerns. • `WARNING`: Warning on an event that might impact the DB System's operation if no actions are taken. • `CRITICAL`: A critical event. An urgent action is required to avoid adverse impact on your DB System.
Message	The message body of the event report.
Created	The date and time the event report was created.

See Events for details on event reports.

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

> **Tip**
>
> DB System backups are Amazon EBS snapshots saved to Amazon Simple Storage Service (Amazon S3). 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 **Resources** 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 opened, with the **Data source** information (**Backup ID** and **Backup name**) 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.

Copyright © 2022, 2025, Oracle and/or its affiliates.

- **Description**: Specify a description for the new DB System or use the generated description.

> **Note**
>
> No administrator user name and password are requested, because the restored DB System uses the same administrator user name and password that were in effect when the backup was created.

4. **Standalone** or **High availability**
   - **Standalone**: Select for restoring the backup to a single-instance DB System.
   - **High availability**: Select for restoring the backup to a three-instance High Availability DB System.

> **Note**
>
> You cannot restore a backup of a standalone DB System to a high availability DB System.

5. Select the **Hardware configuration**. 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**
>
> – When restoring a backup of a high availability DB System, the **Data Storage Size** cannot be smaller than its value in the backed-up DB System.
>
> – Ensure that the specified **Data Storage Size** is not smaller than the size of the backup you are restoring.

6. **Preferred availability zone**: This determines the physical location of the standalone DB System, or the preferred location of the primary instance for a High Availability DB System:
   - **Automatic**: AWS selects the physical AWS Availability Zone.
   - **Manual**: Select a physical AWS Availability Zone.

Copyright © 2022, 2025, Oracle and/or its affiliates.

> **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*.

7. **MySQL Configuration**: Select a MySQL configuration.
   - Click **Change** to change the default configuration.
   - Select a configuration from the list.
   - Click **Apply MySQL Configuration**.

   > **Note**
   >
   > Choices of configuration might be restricted by features you have chosen for the DB system; for example, if this is a high availability system, you will only be able to change to another configuration that supports high availability (if available).

8. **MySQL version**: Select the MySQL Server version to deploy.
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. Configure **Networking** settings. Select **Enable inbound connectivity from allowed public IP address ranges** if you want the DB System to be accessible via the public internet, and provide the following information:
    - **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. The X Protocol is supported by clients such as MySQL Shell. The default port is 33060. You can 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.

      > **Note**
      >
      > For a high availability system, this is selected automatically and cannot be deselected.

    - Set a **Backup retention period** between 1 and 35 days.

Copyright © 2022, 2025, Oracle and/or its affiliates.

- Select **Configure backup window** to set a preferred time to start a backup and set the **Backup start time**.

12. **IAM roles**:
    - **Data Import role ARN**: If you want to import data with Console, specify the role ARN. See Creating an IAM Role to Access an Amazon S3 Bucket.
    - **MySQL HeatWave AutoML role ARN**: If you want to use the external AWS Bedrock LLMs for MySQL HeatWave GenAI, specify the role ARN. See Creating an IAM Role to Access Amazon Bedrock LLMs.
    - **Lakehouse role ARN**: If you want to use Lakehouse, specify the role ARN with one of the two options.
        - **Custom Lakehouse role**: Choose and then specify the **Lakehouse role ARN**. See Creating an IAM Role to Access an Amazon S3 Bucket.
        - **Sample Lakehouse role**: Choose to use a sample **Lakehouse role ARN** for accessing sample Lakehouse data provided by MySQL HeatWave on AWS.
13. Click **Next** to proceed to the next page for creating a MySQL HeatWave Cluster.
14. Select **Provision MySQL HeatWave Cluster** to create a MySQL HeatWave Cluster if desired, and then:

> **Note**
>
> Provisioning a MySQL HeatWave Cluster is not supported when restoring a backup to a high availability DB System.

    - Provide **Basic information** for the new MySQL HeatWave Cluster:
        - **Display Name**: Specify a display name for the new MySQL HeatWave Cluster or use the generated default name.
        - **Description**: Specify a description for the new MySQL HeatWave Cluster.
    - Under **MySQL HeatWave Lakehouse**, select **Enable MySQL HeatWave Lakehouse** if you want the feature to be supported.
    - Under **MySQL HeatWave Cluster Configuration**, select a node shape and cluster size.
        - **Shape**: Select a MySQL HeatWave node shape. For information about supported shapes, see Supported Shapes.

> **Note**
>
> If you intend to use MySQL HeatWave AutoML functionality, the `MySQL HeatWave.256GB` node shape is recommended when creating a MySQL HeatWave Cluster. The `MySQL 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.

        - **Node count**: The number of MySQL HeatWave nodes to create. Enter a number between 1 and 128.

Copyright © 2022, 2025, Oracle and/or its affiliates.

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

15. Check the **Summary** on the DB System. Click **Back** to go back and make changes.
16. Click **Restore** to restore the backup to the new DB System.

    You can go to the **Resources** > **DB Systems** tab to monitor the state of the restore operation, which may take some time to complete. The state will change from **Creating** to **Active** when the operation has completed successfully.

18.5 Deleting a Backup

Deleting a backup permanently deletes it.

To delete a backup:

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

18.6 Billing and Free Quota for Backup Storage

The backup storage billing depends on the DB system state and the data storage size that you defined while creating a DB system or updating the storage size. You can see a DB System's state and data storage size by Viewing DB System Details.

The way you are billed for backup storage depends on the **DB System State**:

- **Active/Inactive**: You are billed only when the backups' total size exceeds the free quota described below.
- **Failed/Deleted**: There is no free storage available and you are billed for all backup storage.

Free Quota for DB Systems

You get free storage when the DB system is in the **Active** or **Inactive** state. The size of the free storage equals to the data storage size of the DB system. For example: For a DB system with 500 GiB of storage size, the free quota for backup storage is 500 GiB.

Backups

Copyright © 2022, 2025, Oracle and/or its affiliates.

Let us assume you have two active DB systems with data storage size of 50 GiB and 100 GiB in the tenancy in region 1. You are entitled to a maximum of 150 GiB free backup storage, which is the sum of the data storage size of the two DB systems in region 1.

Now you created 4 manual backups:

- Backup 1 - 30 GiB
- Backup 2 - 50 GiB
- Backup 3 - 80 GiB
- Backup 4 - 85 GiB

The total manual backup size is 30 + 50 + 80 + 85 GiB = 245 GiB. Let us assume the automatic backups have a total size of 50 GiB.

You are billed for region 1 as follows:

Billed backup = Total backup size including manual and automatic - free storage, which is (245+50) - 150 = 145 GiB.

> **Note**
>
> If you have tenancies in two regions, each region is billed separately following the billing method as described above.

Copyright © 2022, 2025, Oracle and/or its affiliates.

19 Configuration

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

- Creating a MySQL Configuration
- Copying a MySQL Configuration
- MySQL Configuration Details
- System Initialization Variables
- User-Configurable System Variables
- User-Configurable Shape-Dependent System Variables
- Service-Specific System Variables
- Shape-Dependent System Variables
- Session Variables

19.1 Creating a MySQL Configuration

Use the MySQL HeatWave Console to create a new MySQL configuration.

> **Note**
>
> It is not possible to change the values of any variables once a configuration has been created. However, you can create a custom configuration by copying and modifying an existing configuration. You can then update your DB system's configuration using your custom configuration.

Follow these steps to create your MySQL configuration:

1. In the MySQL HeatWave Console, select the **Resources** tab, and then click the **Configurations** tab.
2. On the **Configurations** tab, click **Create MySQL Configuration** to open the **Create MySQL Configuration** dialog.
3. On the **General Information** page of the **Create MySQL Configuration** dialog, provide the following information:
   - **Basic information**:
     - **Display Name**: Specify a display name for the configuration or use the generated default name.
     - **Description**: Specify a simple description for the configuration.
   - **Select MySQL shape**: Select a MySQL shape to create a configuration for.
4. Click **Next**. On the **Configuration Variables** page of the dialog, provide the following information:

Copyright © 2022, 2025, Oracle and/or its affiliates.

- **Configuration validation**: Select one of these features to ensure that this MySQL Configuration supports it:
  - **None**: No feature validation will be performed
  - **MySQL HeatWave Clusters**: The configuration will be validated for MySQL HeatWave Cluster support.
  - **High Availability**: The configuration will be validated for high availability support.
- **Initialization variables**: Set preferred values for (and only for) initialization variables whose values you want to be different from the default setting.
  - 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. If no value is selected or entered, the value in the base configuration you are copying (shown in the value box) is used.
  - Click **X** to remove a variable you have selected by the steps above, if you change your mind and no longer want to set a value for it.
  - Click the information icon to show the default value of the variable you have selected. A **Learn More** link for more information on the selected variable also appears.
  - Click the setting icon to reset a selected variable to its default value, if it has been set to something else. If the selected variable has MySQL version-specific default values, it is set to the default value of the latest MySQL version supported by MySQL HeatWave on AWS.
- **User variables**: Set preferred values for (and only for) user variables whose values you want to be different from the default settings.
  - 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. If no value is selected or entered, the value in the base configuration you are copying (shown in the value box) is used. For any variables of the string data type, an empty value field means an empty string, except for nullable variables such as `innodb_ft_server_stopword_table` , for which an empty value field is taken as `MYSQL NULL` (i.e., a null value).
  - Click **Add new variable** to set more variables with the steps above, until you specified all the variables you want to.
  - Click **X** to remove a variable you have selected or added by the steps above, if you change your mind and no longer want to set a value for it.
  - Click the information icon to show the default value of the variable you have selected. A **Learn More** link for more information on the selected variable also appears.
  - Click the setting icon to reset a selected variable to its default value, if it has been set to something else. If the selected variable has MySQL version-specific default values, it is set to the default value of the latest MySQL version supported by MySQL HeatWave on AWS.

5. Click **Next**. On the **Review Configuration** page of the dialog, review all the variable values you have selected in Step 4 above.

   Select a filter to show the variables and their values associated with the configuration:

   - **All**: Show all configuration variables (default)
   - **Dynamic**: Show all configuration variables that are dynamic and can be set at runtime.

Copyright © 2022, 2025, Oracle and/or its affiliates.

- **Shape or feature specific**: Show all configuration variables that have MySQL shape or feature-specific values.
- **Default**: Show all configuration variables that have not been set by users.
- **User-defined**: Show all configuration variables that have been set by users. User-defined variables are those that have ever been set in the dialog when the configuration was created. The actual value of the variable does not matter: for example, you could have selected a variable in the configuration dialog and then left its displayed value (which was the value from your base configuration) unchanged, but the variable would still be taken as "user-defined" and displayed by this filter.

You can also refine the list by typing into the search bar part of the names of the variables you are interested in.

Click **Back** to go back and edit any variable values if needed.

Click **X** to remove a variable that you have added in the Step 4.

6. Click **Create** after you are happy with your configuration.

   MySQL HeatWave checks the values for each variable. If any values fail validation, the message `Could not create MySQL Configuration` appears together with the names of the failed variables. Correct any invalid values, and click **Create** again.

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

To review the default values for the MySQL global system variables and choose you preferred values, see the related links below. Notice that certain MySQL global system variables define whether a configuration can support MySQL HeatWave.

Related Topics

- System Initialization Variables
- User-Configurable System Variables
- User-Configurable Shape-Dependent System Variables
- View MySQL Configuration Details
- Update MySQL Configuration

19.2 Copying a MySQL Configuration

While it is not possible to change the values of any variables once a configuration has been created, you can create a custom configuration by copying and modifying an existing configuration. You can then update your DB system's configuration using your custom configuration.

Follow these steps to copy a MySQL configuration and create a custom configuration out of it:

1. In the MySQL HeatWave Console, select the **Resources** tab, and then click **Configurations** to open the **Configurations** tab.
2. On the **Configurations** page, select a base configuration you want to copy and then click **Copy MySQL Configuration** to open the **Copy MySQL Configuration** dialog.
3. On the **General Information** page of the **Copy MySQL Configuration** dialog, provide the following information:
   - **Basic information**:

Copyright © 2022, 2025, Oracle and/or its affiliates.

  - **Display Name**: Specify a display name for the configuration or use the generated default name.
  - **Description**: Specify a simple description for the configuration.
- **Source MySQL Configuration shape**: The MySQL shape of the configuration you are copying is displayed.

4. Click **Next**. On the **Configuration Variables** page of the dialog, provide the following information:
   - **Configuration validation**: Select one of these features to ensure that this MySQL Configuration supports it:
     - **None**: No feature validation will be performed
     - **MySQL HeatWave Clusters**: The configuration will be validated for MySQL HeatWave Cluster support.
     - **High Availability**: The configuration will be validated for high availability support.
   - **Initialization variables**: Set preferred values for (and only for) any initialization variables whose values you want to be different from the base configuration you are copying:
     - 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. If no value is selected or entered, the value in the base configuration you are copying (shown in the value box) is used.
     - Click **X** to remove a variable you have selected by the steps above, if you change your mind and no longer want to set a value for it.
     - Click the information icon to show the default value of the variable you have selected. A **Learn More** link for more information on the selected variable also appears.
     - Click the setting icon to reset a selected variable to its default value, if it has been set to something else. If the selected variable has MySQL version-specific default values, it is set to the default value of the latest MySQL version supported by MySQL HeatWave on AWS.
   - **User variables**: Set preferred values for (and only for) any user variables whose values you want to be different from the base configuration you are copying.
     - 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. If no value is selected or entered, the value in the base configuration you are copying (shown in the value box) is used. For any variables of the string data type, an empty value field means an empty string, except for nullable variables such as `innodb_ft_server_stopword_table` , for which an empty value field is taken as `MYSQL NULL` (i.e., a null value).
     - Click **Add new variable** to set more variables with the steps above, until you specified all the variables you want to.
     - Click **X** to remove a variable you have selected or added by the steps above, if you change your mind and no longer want to set a value for it.
     - Click the information icon to show the default value of the variable you have selected. A **Learn More** link for more information on the selected variable also appears.

Copyright © 2022, 2025, Oracle and/or its affiliates.

– Click the setting icon to reset a selected variable to its default value, if it has been set to something else. If the selected variable has MySQL version-specific default values, it is set to the default value of the latest MySQL version supported by MySQL HeatWave on AWS.

> **Note**
>
> User variables in the base configuration you are copying are pre-selected here and pre-populated with their values in the base configuration. They are not removable by the **X** button, and will remain as user variables in the new custom configuration you are creating.

5. Click **Next**. On the **Review Configuration** page of the dialog, review all the variable values you have selected in Step 4 above.

   Select a filter to show the variables and their values associated with the configuration:

   - **All**: Show all configuration variables (default)
   - **Dynamic**: Show all configuration variables that are dynamic and can be set at runtime.
   - **Shape or feature specific**: Show all configuration variables that have MySQL-shape-specific or feature-specific values.
   - **Default**: Show all configuration variables that have not been set by users.
   - **User-defined**: Show all configuration variables that have been set by users. User-defined variables are those that have ever been set in a configuration dialog when the configuration or its base configuration was created. The actual value of the variable does not matter: for example, you could have selected a variable in the configuration dialog and then left its displayed value (which was the value from your base configuration) unchanged, but the variable would still be taken as "user-defined" and displayed by this filter.

   You can also refine the list by typing into the search bar part of the names of the variables you are interested in.

   Click **Back** to go back and edit any variable values if needed.

   Click **X** to remove a variable that you have added in the Step 4.

6. Click **Copy** after you are happy with your configuration.

   MySQL HeatWave checks the values for each variable. If any values fail validation, the message `Could not create MySQL Configuration` appears together with the names of the failed variables. Correct any invalid values, and click **Copy** again.

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

To review the default values for the MySQL global system variables and choose you preferred values, see the related links below. Notice that certain MySQL global system variables define whether a configuration can support MySQL HeatWave.

Related Topics

- System Initialization Variables
- User-Configurable System Variables
- User-Configurable Shape-Dependent System Variables
- View MySQL Configuration Details

Copyright © 2022, 2025, Oracle and/or its affiliates.

19.3 MySQL Configuration Details

Follow these steps to view details of your MySQL configurations:

1. In the MySQL HeatWave Console, select the **Resources** tab, and then click the **Configurations** tab.
2. In the list of configurations, find the configuration for which you want to view its details, and do one of the following:
   - Click the row of the configuration to highlight it. The configuration details appear below the list of configurations. See MySQL Configuration Details Page for an explanation of the details given.
   - Click the name of the configuration to open the MySQL Configuration Details Page for the selected DB System.

The following actions can be taken for the configuration on either the MySQL HeatWave Console **Configurations** tab or on the **MySQL Configuration Details** page:

- Click **Edit Configuration** to open the **Edit MySQL Configuration** dialog box, in which you can edit the **Basic information** for the configuration including the **Display name** and **Description**. Note that no other properties of the configuration can be altered.
- Click **Copy Configuration** to create a new configuration based on the current configuration. After the **Copy MySQL Configuration** dialog opens, follow the instructions given in Copying a MySQL Configuration.
- Click **Delete** to delete the configuration if it is no longer needed by any DB Systems.

The following tables describe the information provided on the **MySQL Configuration Details** page.

Table 19-1 Tabs on MySQL Configuration Details Page

Name	Description
Summary	Configuration summary details. See .
Variables	Variable information. See Table 19-3.
DB Systems	Information on DB Systems that use the configuration. See Table 19-4 .
Events	Event details. See Table 19-5.

Table 19-2 MySQL Configuration Summary Details

Field	Description
State	The state of the configuration: • **ACTIVE**: The configuration was successfully created and is ready to be used. • **DELETED**: The configuration has been deleted and is no longer available.
Resource ID	The resource ID assigned by MySQL HeatWave to the configuration.
Type	The type of configuration: • **Default**: It is a default configuration for a MySQL shape. • **Custom** It is a custom configuration created based on a **Default** configuration or another custom configuration.
Source MySQL Configuration	The source configuration based on which the current configuration was created, if this is a custom configuration.

Copyright © 2022, 2025, Oracle and/or its affiliates.

Table 19-2 (Cont.) MySQL Configuration Summary Details

Field	Description
Feature Support	Feature supported by the configuration (MySQL HeatWave, High Availability, etc.) .
Shape	The MySQL shape for which the configuration was created. For information about shapes, see Supported Shapes.
Description	The description that was provided when the configuration was created.
Last updated	The date and time the configuration was last updated. If the configuration has never been updated, it shows the date of time of its creation.
Created	The date and time the configuration was created.

Table 19-3 MySQL Configuration Variables Details

Field	Description
General information	• **Name**: The name of the configuration. • The state of the configuration: – **ACTIVE**: The configuration was successfully created and is ready to be used. – **DELETED**: The configuration has been deleted and is no longer available. • **Resource ID**: The unique resource identifier assigned to the configuration when it is created. • **Description**: The user-defined description of the configuration. • **Last updated**: The date and time the configuration was last updated. If the configuration has never been updated, it shows the date of time of its creation. • **Created**: The date and time the configuration was created.
Configuration	• **Type**: The type of configuration: – **Default**: It is a default configuration for a MySQL shape. – **Custom** It is a custom configuration created based on a **Default** configuration or another custom configuration. • **Shape**: The MySQL shape for which the configuration was created. For information about shapes, see Supported Shapes. • **Source MySQL Configuration**: The source configuration based on which the current configuration was created, if this is a custom configuration. • **Supports MySQL HeatWave**: Whether the configuration supports MySQL HeatWave.
Initialization variables	All the System Initialization Variables in the configuration.

Copyright © 2022, 2025, Oracle and/or its affiliates.

Table 19-3 (Cont.) MySQL Configuration Variables Details

Field	Description
User variables	Select a filter to show the variables and their values associated with the configuration: • **All**: Show all configuration variables (default) • **Dynamic**: Show all configuration variables that are dynamic and can be set at runtime. • **Shape or feature specific**: Show all configuration variables that have MySQL shape or feature-specific values. • **Default**: Show all configuration variables that have not been set by users. • **User-defined**: Show all configuration variables that have been set by users. User-defined variables are those that have ever been set in either the **Create MySQL Configuration** or the **Copy MySQL Configuration** dialog when the configuration was created. The actual value of the variable does not matter: for example, you could have selected a variable in the configuration dialog and then left its displayed value (which was the value from your base configuration) unchanged, but the variable would still be taken as "user-defined" and displayed by this filter. You can also refine the list by typing into the search bar part of the names of the variables you are interested in.

Table 19-4 MySQL Configuration DB Systems Details

Field	Description
Name	Name of a DB System that uses the configuration. Click the name to see the MySQL DB System Details.
State	The state of the DB System that uses the configuration. See Table 3-8 for descriptions of the DB System states.
MySQL HeatWave Cluster	The MySQL HeatWave Cluster for the DB System that uses the configuration. Click the name to view the MySQL HeatWave Cluster Details.
MySQL HeatWave State	The state of the MySQL HeatWave Cluster of the DB System that uses the configuration. See Table 4-3 for descriptions of the MySQL HeatWave Cluster states.
Created	The date and time the DB System that uses the configuration was created.

Table 19-5 MySQL Configuration Event Details

Field	Description
Type	The event and its status.
Severity	The severity level of the event, which can be one of • `INFO`: General information is provided. No special concerns • `WARNING`: Warning on an event that might impact the DB System's operation if no actions are taken. • `CRITICAL`: A critical event. An urgent action is required to avoid adverse impact on your DB System.
Message	The message body of the event report.
Created	The date and time the event report was created.

Copyright © 2022, 2025, Oracle and/or its affiliates.

See Events for details on event reports.

19.4 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 and Copying a MySQL Configuration.

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

Table 19-6 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. SelectingCASE_INSENSITIVE_LOWERCASE sets the system variable, lower_case_table_names, to 1.

19.5 User-Configurable System Variables

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

> **Note**
>
> Click on the name of a system varuable in Table 19-7 to see its description in the MySQL Reference Manual. Notice that variable's **Default Value** for MySQL HeatWave on AWS, which is given in the table, might be different from the value given in the MySQL Reference Manual,

Table 19-7 User-Configurable System Variables

Name	Default Value
audit_log_disable	ON
autocommit	ON
auto_increment_increment	1
auto_increment_offset	1
big_tables	OFF

Copyright © 2022, 2025, Oracle and/or its affiliates.

Table 19-7 (Cont.) User-Configurable System Variables

Name	Default Value
binlog_expire_logs_seconds	3600
binlog_group_commit_sync_delay	0
binlog_group_commit_sync_no_delay_count	0
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
explicit_defaults_for_timestamp	ON
foreign_key_checks	ON
global_connection_memory_limit	9223372036854775807
global_connection_memory_tracking	OFF
group_replication_consistency	EVENTUAL
information_schema_stats_expiry	86400
innodb_autoinc_lock_mode	2
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_numa_interleave	ON
innodb_online_alter_log_max_size	134217728
innodb_rollback_on_timeout	OFF
innodb_sort_buffer_size	1048576
innodb_stats_persistent_sample_pages	20
innodb_stats_transient_sample_pages	8
innodb_write_io_threads	4

Copyright © 2022, 2025, Oracle and/or its affiliates.

Table 19-7 (Cont.) User-Configurable System Variables

Name	Default Value
interactive_timeout	28800
join_buffer_size	262144
long_query_time	10
--local-infile	OFF
mandatory_roles	Empty string
max_allowed_packet	67108864
max_binlog_cache_size	4294967296
max_connect_errors	18,446,744,073,709,551,615
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
max_seeks_for_key	18446744073709551615
max_user_connections	0
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

Copyright © 2022, 2025, Oracle and/or its affiliates.

Table 19-7 (Cont.) User-Configurable System Variables

Name	Default Value
optimizer_switch	**For MySQL 9.3.1 and later:** 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=on, derived_condition_pushdown=on **For MySQL 9.3.0 and earlier:** 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
parser_max_mem_size	18446744073709551615
range_optimizer_max_mem_size	8388608
regexp_time_limit	32
replica_pending_jobs_size_max	134217728
replica_type_conversions	Empty string
require_secure_transport	ON. Secure transport can be disabled only for private connections over PrivateLink. If your DB System allows public access, you need to update the network configuration to disable it before you can set this variable to OFF.

Copyright © 2022, 2025, Oracle and/or its affiliates.

Table 19-7 (Cont.) User-Configurable System Variables

Name	Default Value
skip_name_resolve	OFF
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
table_definition_cache	AUTOSIZED
table_open_cache	4000
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
transaction_isolation	REPEATABLE-READ
wait_timeout	28800

19.6 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 and Copying a MySQL Configuration.

> **Note**
>
> innodb_buffer_pool_instances , innodb_buffer_pool_size , innodb_read_io_threads , max_connections , and max_prepared_stmt_count have different default, minimum and maximum values for each shape and for MySQL HeatWave support.
>
> thread_pool_max_transactions_limit and thread_pool_query_threads_per_group have different default values for each shape.

Table 19-8 innodb_buffer_pool_instances

Shape	Supports MySQL 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

Copyright © 2022, 2025, Oracle and/or its affiliates.

Table 19-8 (Cont.) innodb_buffer_pool_instances

Shape	Supports MySQL HeatWave	Default	Minimum	Maximum
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 19-9 innodb_buffer_pool_size

Shape	Supports MySQL 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	51539607552
MySQL.8.64GB	Yes	46170898432	5242880	46170898432
MySQL.32.256GB	No	206158430208	5242880	206158430208
MySQL.32.256GB	Yes	21474836480	5242880	206158430208

Table 19-10 innodb_read_io_threads

Shape	Supports MySQL HeatWave	Default	Minimum	Maximum
MySQL.2.16GB	No	4	1	64
MySQL.2.16GB	Yes	32	1	64
MySQL.4.32GB	No	4	1	64
MySQL.4.32GB	Yes	32	1	64
MySQL.8.64GB	No	4	1	64
MySQL.8.64GB	Yes	32	1	64
MySQL.32.256GB	No	16	1	64
MySQL.32.256GB	Yes	32	1	64

Table 19-11 max_connections

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

Copyright © 2022, 2025, Oracle and/or its affiliates.

Table 19-11 (Cont.) max_connections

Shape	Supports MySQL HeatWave	Default	Minimum	Maximum
MySQL.32.256GB	No	8000	1	16000
MySQL.32.256GB	Yes	2000	1	16000

Table 19-12 max_prepared_stmt_count

Shape	Supports MySQL 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 19-13 replica_parallel_workers

Shape	Supports MySQL HeatWave	Default	Minimum	Maximum
MySQL.2.16GB	No	48	1	1024
MySQL.2.16GB	Yes	48	1	1024
MySQL.4.32GB	No	48	1	1024
MySQL.4.32GB	Yes	48	1	1024
MySQL.8.64GB	No	64	1	1024
MySQL.8.64GB	Yes	64	1	1024
MySQL.32.256GB	No	256	1	1024
MySQL.32.256GB	Yes	16	1	1024

Table 19-14 thread_pool_max_transactions_limit

Shape	Default	Minimum	Maximum
MySQL.2.16GB	32	0	100000
MySQL.4.32GB	64	0	100000
MySQL.8.64GB	128	0	100000
MySQL.32.256GB	512	0	100000

Table 19-15 thread_pool_query_threads_per_group

Shape	Default	Minimum	Maximum
MySQL.2.16GB	2	1	4096
MySQL.4.32GB	2	1	4096

Copyright © 2022, 2025, Oracle and/or its affiliates.

Table 19-15 (Cont.) thread_pool_query_threads_per_group

Shape	Default	Minimum	Maximum
MySQL.8.64GB	2	1	4096
MySQL.32.256GB	2	1	4096

19.7 Service-Specific System Variables

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

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

Table 19-16 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%';
```


Copyright © 2022, 2025, Oracle and/or its affiliates.

19.8 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.

> **Note**
>
> innodb_parallel_read_threads is also a session variable. See: Session Variables.

Table 19-17 MySQL.2.16GB Global System Variables

Name	Default Value
back_log	1000
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 19-18 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 19-19 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

Copyright © 2022, 2025, Oracle and/or its affiliates.

Table 19-20 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
temptable_max_ram	1073741824
thread_pool_size	32

19.9 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 19-21 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

Copyright © 2022, 2025, Oracle and/or its affiliates.

Table 19-21 (Cont.) User Settable Session Variables

Name	Default Value
--default_tmp_storage_engine	InnoDB
default_week_format	0
div_precision_increment	4
end_markers_in_json	OFF
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

Copyright © 2022, 2025, Oracle and/or its affiliates.

Table 19-21 (Cont.) User Settable Session Variables

Name	Default Value
optimizer_switch	**For MySQL 9.3.1 and later:** 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=on, derived_condition_pushdown=on **For MySQL 9.3.0 and earlier:** 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

Copyright © 2022, 2025, Oracle and/or its affiliates.

Table 19-21 (Cont.) User Settable Session Variables

Name	Default Value
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
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

Copyright © 2022, 2025, Oracle and/or its affiliates.

20

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 `MySQL 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
- Groups and Policies
- User Management

20.1 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

> **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. Any other users must be added to at least one of the groups for access to MySQL HeatWave on AWS.

Copyright © 2022, 2025, Oracle and/or its affiliates.

Table 20-1 OracleMySQLHeatwaveDBUsers Group

Group Description	Resources and Permissions
OracleMySQLHeatwaveDBUsers: Members of this group can use DB Systems, DB System Backup, MySQL HeatWave Clusters, MySQL Configurations, Inbound Replication Channel, Service Events, and PrivateLinks resources	DB Systems • View supported shapes • View supported MySQL versions • View DB Systems • View DB System Details • Update DB Systems • Run queries • View query statuses • Stop queries • Import data • View data imports • View data import details • Cancel data imports DB System Backups • View DB System backups • View DB System backup details • Update DB System backups MySQL HeatWave Clusters • View MySQL HeatWave Clusters • View MySQL HeatWave Clusters details • Estimate MySQL HeatWave Cluster size • View supported shapes MySQL Configurations • View configurations • View configuration details • Update configurations • View configuration variable metadata Inbound Replication Channels • View channels • View channel details • Resume channels Service Events • View events • View event details PrivateLinks • View PrivateLinks • View PrivateLink Details • Update PrivateLinks

Copyright © 2022, 2025, Oracle and/or its affiliates.

Table 20-2 OracleMySQLHeatwaveDBAdmin Group

Group Description	Resources and Permissions
OracleMySQLHeatwaveDBAdmin: Members of this group can manage all aspects of DB Systems, DB System Backups,MySQL HeatWave Clusters, MySQL Configurations, Inbound Replication Channel, and PrivateLinks resources.	**In addition to *OracleMySQLHeatwaveDBUsers* group permissions**, this group has these permissions: DB Systems • Create DB Systems • Delete DB Systems • Start DB Systems • Stop DB Systems • Restart DB Systems DB System Backups • Create DB System backups • Delete DB System backups MySQL HeatWave Clusters • Create MySQL HeatWave Clusters • Delete MySQL HeatWave Clusters • Start MySQL HeatWave Clusters • Stop MySQL HeatWave Clusters • Restart MySQL HeatWave Clusters MySQL Configurations • Create configurations • Delete configurations Inbound Replication Channels • Create channels • Delete channels • Update channels • Reset channels PrivateLinks • Create PrivateLinks • Delete PrivateLinks • Update existing PrivateLinks

Table 20-3 OracleMySQLHeatwaveServiceAccountAdmin Group

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

20.2 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*.

Copyright © 2022, 2025, Oracle and/or its affiliates.

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.

20.3 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.

Copyright © 2022, 2025, Oracle and/or its affiliates.

> **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*.

Copyright © 2022, 2025, Oracle and/or its affiliates.

21
Account Management

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

- Manage Regions
- Service Limits
- Billing
- Viewing OCID of the Tenancy
  Use the Oracle Cloud Infrastructure (OCI) Console to view and copy the OCID of the tenancy.
- Manage AWS Access

21.1 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**.

21.2 Service Limits

A service limit is the quota or allowance set on a resource. The 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.

Copyright © 2022, 2025, Oracle and/or its affiliates.

Table 21-1 Service Limits

Resource	Default Limit	Free Trial Limit
DB Systems	1000*	2
MySQL HeatWave Clusters	1000*	2
DB System Block Storage	128 TiB	300 GiB
MySQL Instance Manual Backups	40 (for all DB System instances)	2
Sum of DB System Backup Retention Days**	300	4
MySQL.32.256GB instances	10	3
MySQL.8.64GB instances	10	3
MySQL.4.32GB instances	10	3
MySQL.2.16GB instances	10	3
HeatWave.16GB nodes	30	4
HeatWave.256GB nodes	30	1
DB Systems Outbound Data Transfer	None	30 GiB per day***
Egress private endpoint	30	2
Ingress private endpoint	30	2

* The maximum number of DB Systems and MySQL HeatWave Clusters you can create are also constrained by the MySQL instances and MySQL HeatWave Cluster nodes limits.

** This refers to the sum of the retention periods for all the automatic backups of all the DB Systems in your tenancy. For example, if there are two DB Systems in your tenancy and both have automatic backup set up, with one having a retention period of 5 days and another having a retention period of 7 days, the Sum of DB System Backup Retention Days will be 12.

*** During a free trial, when the total outbound data transfer from the DB Systems exceeds the daily limit, connections to DB Systems will be blocked for the rest of the day. At midnight in UTC, the daily outbound data transfer is reset and connections are unblocked.

> **Tip**
>
> Even when connections to DB Systems are blocked, you can continue to use the MySQL HeatWave Console.

21.3 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**.

Copyright © 2022, 2025, Oracle and/or its affiliates.

3. Select **Billing**.

21.4 Viewing OCID of the Tenancy

Use the Oracle Cloud Infrastructure (OCI) Console to view and copy the OCID of the tenancy.

This task requires the following:

- Access to OCI Console.

Do the following to view and copy the OCID of the tenancy:

1. Open the Oracle Cloud Infrastructure (OCI) Console.
2. Sign in with your **Cloud Account Name** (sometimes referred to as your tenancy name), user name, and password.
3. In the Oracle Cloud Infrastructure Console home page, click the **Navigation menu**, and then click **Governance & Administration**. Under **Account Management**, click **Tenancy Details**.
4. In the **Tenancy details** page, in the **Tenancy information** tab, click **Show** or **Copy** present besides the **OCID** field to view or copy the OCID of the tenancy respectively.

21.5 Manage AWS Access

Use the AWS Management Console to manage AWS access.

`IDDetails`: You need to specify the `IDDetails` in the trust policy when creating some of the IAM roles described in this chapter. The `IDDetails` to use depend on the situation:

- To grant access to a specific DB System in the tenancy, use <*TenancyOCID*>/<*DBSystemResourceId*> (see Viewing OCID of the Tenancy on how to find the OCID of the tenancy, and see Viewing DB System Details on how to find the resource ID of the DB System). For example:

  ```
  ocid1.tenancy.oc1..aaaaaaaaba4pv6wkcr4jqae5f44n2b2m2yt2j6rx32uzr4h25vqstifsfdsk/5281bb96-99a1-23fe-a65f-370cd85b979g
  ```

- To grant access to all DB Systems in the tenancy, use <*TenancyOCID*>/*. For example:

  ```
  ocid1.tenancy.oc1..aaaaaaaaba4pv6wkcr4jqae5f44n2b2m2yt2j6rx32uzr4h25vqstifsfdsk/*
  ```

> **Note**
>
> If you are creating the IAM role to grant access to a specificDB System that is yet to be created, grant access to all DB Systems in the tenancy by using <*TenancyOCID*>/* first for the `IDDetails`, and once the DB System is created, update the trust policy to limit access to the new DB System by using <*TenancyOCID*>/<*DBSystemResourceId*> instead.

- Creating an IAM Policy to Access an Amazon S3 Bucket
- Creating an IAM Role to Access an Amazon S3 Bucket
- Creating an IAM Policy to Access Amazon Bedrock LLMs

Copyright © 2022, 2025, Oracle and/or its affiliates.

- Creating an IAM Role to Access Amazon Bedrock LLMs
- Creating an IAM Role and Policy for CloudWatch Monitoring

21.5.1 Creating an IAM Policy to Access an Amazon S3 Bucket

Use the AWS Management Console to create an IAM policy to access an Amazon S3 bucket.

This task requires the following:

- Access to AWS Management Console.
- The name of the Amazon S3 bucket you want to grant access to.
- Sufficient IAM permissions for creating the IAM policies.

Do the following to create an IAM policy:

1. Open the AWS Management Console and sign in with your credentials.
2. In the AWS Management Console home page, click **Services**, and click **Security, Identity, & Compliance**, and then click **IAM**.
3. In the navigation pane of the Console, under **Access management**, click **Policies**, and then click **Create policy**.

   It opens the **Specify permissions** page.
4. In the **Specify permissions** page, in the **Policy editor** section, click **JSON**, and enter the following Amazon S3 permissions as per the feature you use. For more information on policies, see Generate policies.
   - Sample permissions for the **Data Import role** for using the Bulk Ingest Feature:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [

"arn:aws:s3:::<BulkIngestAmazonS3BucketName>/*"
            ]
        }
    ]
}
```

   - Sample permissions for the **Data Import role** for using the Data Import Feature:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
```


Copyright © 2022, 2025, Oracle and/or its affiliates.

```
"arn:aws:s3:::<DataImportAmazonS3BucketName>/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": [

"arn:aws:s3:::<DataImportAmazonS3BucketName>"
            ]
        }
    ]
}
```

- Sample permissions for the **Lakehouse role** for using Lakehouse:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [

"arn:aws:s3:::<LakehouseAmazonS3BucketName>/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::<LakehouseAmazonS3BucketName>"
            ]
        }
    ]
}
```

- Sample permissions for the **Lakehouse role** for exporting MySQL HeatWave query results to S3:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": [
```


Copyright © 2022, 2025, Oracle and/or its affiliates.

```
                "arn:aws:s3:::<S3ExportAmazonS3BucketName>/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                 "s3:DeleteObject"
             ],
            "Resource": [
                "arn:aws:s3:::<S3ExportAmazonS3BucketName>/*"
             ]
        }
    ]
}
```

- If the objects in the Amazon S3 bucket are encrypted with a customer-managed KMS key, add the following permission for the keys used for encrypting the objects:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt"
            ],
            "Resource": [
                "<KmsKeyArn>"
            ]
        }
    ]
}
```

  See Using IAM policies with AWS KMS.

5. Resolve any warnings or errors generated during permissions validation, and then click **Next**.
6. In the **Review and create** page, in the **Policy details** section, enter the following:
   - **Policy name**: Specify a name to identify this policy.
   - **Description**: (Optional) Specify a description of the policy.
7. Click **Create policy**.

Related Topics

- Creating an IAM Role to Access an Amazon S3 Bucket

21.5.2 Creating an IAM Role to Access an Amazon S3 Bucket

Use the AWS Management Console to create an IAM role for accessing an Amazon S3 bucket (see Using an external ID for third-party access for more information on the topic).

This task requires the following:

- Access to AWS Management Console.

Copyright © 2022, 2025, Oracle and/or its affiliates.

- IAM policy that specifies the Amazon S3 and/or KMS permissions required for the feature you want to use. See Creating an IAM Policy to Access an Amazon S3 Bucket.
- The tenancy Oracle Cloud Identifier (OCID). To view the OCID of the tenancy, see Viewing OCID of the Tenancy.
- If you want to grant access to a specific DB System, the resource ID of the DB System.
- The name of the Amazon S3 bucket you want to grant access to.
- Sufficient IAM permissions for creating the IAM roles.

Do the following to create an IAM role:

1. Open the AWS Management Console and sign in with your credentials.
2. In the AWS Management Console home page, click **Services**, and click **Security, Identity, & Compliance**, and then click **IAM**.
3. In the navigation pane of the Console, under **Access management**, click **Roles**, and then click **Create role**.
4. In the **Select trusted entity** panel, do the following:
   a. Select **Custom trust policy**.
   b. Specify the following trust policy:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Principal": {
                "AWS": "612981981079"
            },
            "Condition": {
                "StringLike": {
                    "sts:ExternalId": "<IDDetails>"
                }
            }
        }
    ]
}
```

   See Manage AWS Access for what to use for `<IDDetails>`.
5. Resolve any warnings or errors generated during policy validation, and then click **Next**.
6. In the **Add permissions** page, search for the policy you created, and select the check box to attach the policy to your new role. See Creating an IAM Policy to Access an Amazon S3 Bucket.
7. Click **Next**.
8. In the **Name, review, and create** page, in the **Role details** section, enter the following:
   - **Role name**: Enter a name to the identify the role.
   - **Description**: (Optional) Specify a description of the policy.
9. Click **Create role**.
10. Click the role you just created.

Copyright © 2022, 2025, Oracle and/or its affiliates.

11. In the **Summary** section, copy the ARN.

After you create the role, enter the role ARN in an existing DB System, or create a new DB System and enter the ARN details. See Editing a DB System and Creating a DB System.

21.5.3 Creating an IAM Policy to Access Amazon Bedrock LLMs

Use the AWS Management Console to create an IAM policy for MySQL HeatWave GenAI to access Amazon Bedrock LLMs.

This task requires the following:

- Access to AWS Management Console.

Do the following to create an IAM policy:

1. Open the AWS Management Console and sign in with your credentials.
2. In the AWS Management Console home page, click **Services**, and click **Security, Identity, & Compliance**, and then click **IAM**.
3. In the navigation pane of the Console, under **Access management**, click **Policies**, and then click **Create policy**.

   It opens the **Specify permissions** page.
4. In the **Specify permissions** page, in the **Policy editor** section, click **JSON**, and enter the following permissions to access the Amazon Bedrock LLMs supported by MySQL HeatWave GenAI. For more information on policies, see Generate policies.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AuthenticatedInvokeModelBedrockPolicy",
      "Action": [
        "bedrock:InvokeModel"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:bedrock:us-east-1::foundation-model/meta.*",
        "arn:aws:bedrock:us-east-1::foundation-model/cohere.*"
      ]
    },
    {
      "Sid": "AuthenticatedListFoundationModelsBedrockPolicy",
      "Action": [
        "bedrock:ListFoundationModels"
      ],
      "Effect": "Allow",
      "Resource" : "*"
    }
  ]
}
```

5. Resolve any warnings or errors generated during permissions validation, and then click **Next**.
6. In the **Review and create** page, in the **Policy details** section, enter the following:

Copyright © 2022, 2025, Oracle and/or its affiliates.

- **Policy name**: Specify a name to identify this policy.
- **Description**: (Optional) Specify a description of the policy.

7. Click **Create policy**.

21.5.4 Creating an IAM Role to Access Amazon Bedrock LLMs

Use the AWS Management Console to create an IAM role for MySQL HeatWave GenAI to access Amazon Bedrock LLMs.

This task requires the following:

- Access to AWS Management Console.
- IAM policy that specifies the permissions required. See Creating an IAM Policy to Access Amazon Bedrock LLMs.
- The tenancy Oracle Cloud Identifier (OCID).
- If you want to grant access to a specific DB System, the resource ID of the DB System.

Do the following to create an IAM role:

1. Open the AWS Management Console and sign in with your credentials.
2. In the AWS Management Console home page, click **Services**, and click **Security, Identity, & Compliance**, and then click **IAM**.
3. In the navigation pane of the Console, under **Access management**, click **Roles**, and then click **Create role**.
4. In the **Select trusted entity** panel, do the following:
   a. Select **Custom trust policy**.
   b. Specify the following trust policy:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Principal": {
                "AWS": "612981981079"
            },
            "Condition": {
                "StringLike": {
                    "sts:ExternalId": "<IDDetails>"
                }
            }
        }
    ]
}
```

   See Manage AWS Access for what to use for `<IDDetails>`. .

5. Resolve any warnings or errors generated during policy validation, and then click **Next**.

Copyright © 2022, 2025, Oracle and/or its affiliates.

6. In the **Add permissions** page, search for the policy you created (see Creating an IAM Policy to Access Amazon Bedrock LLMs), and select the check box to attach the policy to your new role.
7. Click **Next**.
8. In the **Name, review, and create** page, in the **Role details** section, enter the following:
   - **Role name**: Enter a name to identify the role.
   - **Description**: (Optional) Specify a description of the policy.
9. Click **Create role**.
10. Click the role you just created.
11. In the **Summary** section, copy the ARN.

After you created the role, supply the role ARN to an existing DB System, or create a new DB System and enter the ARN details. See Editing a DB System and Creating a DB System.

21.5.5 Creating an IAM Role and Policy for CloudWatch Monitoring

Use the AWS Management Console to create an IAM role for using CloudWatch to monitor your DB Systems and MySQL HeatWave Clusters.

This task requires the following:

- Access to AWS Management Console.
- An AWS account with rights to create and manage IAM roles. This is the account for receiving the metrics emitted by the monitoring service.

Do the following to create an IAM role:

1. Open the AWS Management Console and sign in with your credentials.
2. In the AWS Management Console home page, click **Services**, and click **Security, Identity, & Compliance**, and then click **IAM**.
3. In the navigation pane of the Console, under **Access management**, click **Roles**, and then click **Create role** button.
4. In the **Trusted entity type** panel, select **Custom Trust Policy**, and modify the trust policy to allow the monitoring service to assume this role (see Manage AWS Access on the `<IDDetails>` you should use.):

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "612981981079"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringLike": {
                    "sts:ExternalId":"<IDDetails>"
                }
            }
        }
```


Copyright © 2022, 2025, Oracle and/or its affiliates.

```
    ]
}
```

Click **Next**.

5. On the **Add permissions** page, skip to the next step by clicking **Next**.
6. On the **Name, review, and create** page, do the following:
   a. Enter a Role name, for example `OracleHeatWaveMonitoringRole`, and an optional **Description** if you like.
   b. Review the **Trust policy**, and click **Create role**.
7. On the **IAM** > **Roles** page, search for the role you just created using its name, and then click on it.
8. Modify the **Permissions** for your role:
   a. On the information page for the role, on the **Permissions** tab, click on the **Add permissions** button and select **Create inline policy**,
   b. On the **Specify permissions** page, in the **Policy editor** pane, select the **JSON** tab, and replace the shown policy with the following:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:PutRetentionPolicy"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:/OracleHeatWave/
metrics:*"
        }
    ]
}
```

   This permission settings allows the DB System to publish custom metrics to the **OracleHeatWave** namespace in CloudWatch, create log groups `/OracleHeatWave/metrics:*` and log streams, and perform basic logging operations. Click **Next** after you have finished the editing.
9. On the **Review and create** page, review the policy, provide a **Policy name** for it (for example, `OracleHeatWaveMonitoringPermissions`), and click **Create policy**.
10. In the **Summary** section, copy the role's ARN. You will need it later.

After you create the role, enter the role ARN in an existing DB System, or create a new DB System and enter the ARN to the **Monitoring configuration**. See Editing a DB System and Creating a DB System.

Copyright © 2022, 2025, Oracle and/or its affiliates.

22

Maintenance

This section describes the maintenance of DB Systems and MySQL 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 Editing a 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.

Copyright © 2022, 2025, Oracle and/or its affiliates.

23
Release Notes

Release notes for MySQL HeatWave on AWS

2025-08-26: Data Storage Size Can Now Be Increased For Highly Available DB Systems

- You can now increase the data storage size of a highly available 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.

2025-08-21: New CloudWatch Monitoring Metrics

- The following metrics were added:
  - system.dbvolume.bytes_read
  - system.dbvolume.bytes_write
  - system.dbvolume.read_operations
  - system.dbvolume.write_operations
  - system.network.bytes_received
  - system.network.bytes_sent

  For more information, see CloudWatch Monitoring Metrics

2025-08-07: MySQL HeatWave on AWS Now Available in the Mexico (Central) Region

- MySQL HeatWave on AWS is now available in the AWS Mexico (Central) region. See Region Availability.

2025-07-24: MySQL 8.4.6 Now Supported

- MySQL HeatWave on AWS now supports MySQL 8.4.6. See MySQL 8.4.6 Release Notes for more information on the server release.

  MySQL 8.4.6 is a new release of the 8.4 LTS (Long Term Support) series, which focuses on security and bug fixes and are recommended for customers who prefer established behavior. The innovation releases will include new features.

  We recommend customers who are using any lower versions of MySQL Server to upgrade to MySQL 8.4.6. See Upgrade MySQL Version for instructions.

2025-07-24: MySQL 8.0.43 Now Supported

- MySQL HeatWave on AWS now supports MySQL 8.0.43. See MySQL 8.0.43 Release Notes for more information on the server release.

  MySQL 8.0.43 is a new release of the 8.0 BugFix series, which focuses on security and bug fixes and are recommended for customers who prefer established behavior. The innovation releases will include new features.

  We recommend customers who are using any lower versions of MySQL 8.0 Server to upgrade to MySQL 8.0.43. See Upgrade MySQL Version for instructions.

Copyright © 2022, 2025, Oracle and/or its affiliates.

2025-07-22: Running SQL statements as Asynchronous Tasks Now Supported

- For MySQL 9.3.1 and later, MySQL HeatWave on AWS now supports creating asynchronous tasks for long-running queries and commands, allowing them to run in the background as separate tasks. This enhancement enables you to run complex queries without blocking other operations, thus enhancing the overall performance and usability of . For more information, see Run Tasks Asynchronously.

2025-07-09: Disabling High Availability Now Supported

- You can now disable high availability for a DB system. This converts the primary instance to a standalone DB system and deletes the two secondary instances. See Disabling high availability on a DB System for details.

2025-06-27: CloudWatch Monitoring Now Supported

- MySQL HeatWave on AWS now integrates with AWS CloudWatch, enabling real-time monitoring of your DB Systems and MySQL HeatWave Clusters. With this integration, you can easily track performance metrics, monitor resource utilization, and gain insights into operational health for your systems. You can also set up CloudWatch alarms and build customizable dashboards for an at-a-glance view of your MySQL HeatWave resources. See CloudWatch Monitoring for details.

2025-06-25: New Way to List Available GenAI Models

- MySQL HeatWave GenAI now supports automated discovery and listing of all available in-database and AWS Bedrock LLMs and embedding models starting with MySQL 9.3.2. This enhancement lets you stay up-to-date on model availability and enables you to make informed usage decisions. See View Available Models for details.

2025-06-25: MySQL 9.3.2 Now Supported

- MySQL HeatWave on AWS now supports MySQL 9.3.2, the latest Innovation Release of the MySQL Server. If you want to access the latest features and improvements and stay updated with the latest technologies, the MySQL Innovation Releases might be the best choice for you. You can choose release 9.3.2 when creating a new DB System, and for existing DB Systems, you can manually upgrade to MySQL 9.3.2. Refer to Changes in MySQL HeatWave 9.3.2 and Upgrade MySQL Version.

2025-06-02: Egress PrivateLink for High Availability DB Systems

- With Egress PrivateLink, you can now securely replicate data from an external system running in your AWS account to a High Availability DB System on MySQL HeatWave on AWS, and ensure that all traffic to or from your DB System stays on a private network connection.

2025-05-29: Hypergraph Optimizer Default Setting

- The **Hypergraph Optimizer** is now enabled by default starting with MySQL 9.3.1, whose default value of optimizer_switch has `hypergraph_optimizer=on`.

2025-05-29: Expanded Support for In-Database LLMs

- MySQL HeatWave GenAI now supports an expanded list of in-database LLMs. See MySQL HeatWave GenAI In-Database LLMs for details.

Copyright © 2022, 2025, Oracle and/or its affiliates.

2025-05-29: MySQL 9.3.1 Now Supported

- MySQL HeatWave on AWS now supports MySQL 9.3.1, the latest Innovation Release of the MySQL Server. If you want to access the latest features and improvements and stay updated with the latest technologies, the MySQL Innovation Releases might be the best choice for you. You can choose release 9.3.1 when creating a new DB System, and for existing DB Systems, you can manually upgrade to MySQL 9.3.1. Refer to Changes in MySQL HeatWave 9.3.1 and Upgrade MySQL Version.

2025-05-21: MySQL 9.3.0 Now Supported

- MySQL HeatWave on AWS now supports MySQL 9.3.0, the latest Innovation Release of the MySQL Server. If you want to access the latest features and improvements and stay updated with the latest technologies, the MySQL Innovation Releases might be the best choice for you. You can choose release 9.3.0 when creating a new DB System, and for existing DB Systems, you can manually upgrade to MySQL 9.3.0. Refer to Changes in MySQL HeatWave 9.3.0 and Upgrade MySQL Version.

2025-05-13: Increased Storage Capacity per DB System

- You can now create DB Systems or expand their total storage capacity to 131,072 GiB. The storage expansion is online, so you get elasticity without compromising uptime or performance. For more information, see Creating a DB System and Increasing DB System Storage.

2025-05-12: Enabling High Availability on Standalone DB Systems

- You can now enable High Availability on a standalone DB System. See Enabling high availability on a DB System for details. A high availability DB System consists of three MySQL instances, one primary and two secondary. It assures that when the primary instance fails, one of the secondary instances is promoted automatically to become the primary instance, with zero data loss and minimal downtime.

2025-05-07: PrivateLink for High Availability DB Systems

- Your application running in AWS can now connect to a high availability DB System through a Query PrivateLink to ensure that all traffic to or from your DB System stays on a private network connection. See Deploying a Query PrivateLink.

2025-05-01: MySQL 8.4.5 Now Supported

- MySQL HeatWave on AWS now supports MySQL 8.4.5. See MySQL 8.4.5 Release Notes for more information on the server release.

  MySQL 8.4.5 is a new release of the 8.4 LTS (Long Term Support) series, which focuses on security and bug fixes and are recommended for customers who prefer established behavior. The innovation releases will include new features.

  We recommend customers who are using any lower versions of MySQL Server to upgrade to MySQL 8.4.5. See Upgrade MySQL Version for instructions.

2025-05-01: MySQL 8.4.4 Now Supported

- MySQL HeatWave on AWS now supports MySQL 8.4.4. See MySQL 8.4.4 Release Notes for more information on the server release.

  MySQL 8.4.4 is a new release of the 8.4 LTS (Long Term Support) series, which focuses on security and bug fixes and are recommended for customers who prefer established behavior. The innovation releases will include new features.

Copyright © 2022, 2025, Oracle and/or its affiliates.

We recommend customers who are using any lower versions of MySQL Server to upgrade to MySQL 8.4.4. See Upgrade MySQL Version for instructions.

2025-04-30: Disable Encryption for Private Client Connections

- MySQL HeatWave on AWS requires that all MySQL client and application connections over the public internet are encrypted. However, when a PrivateLink is used for client connections to a DB System, MySQL HeatWave on AWS now provides the flexibility to disable encryption. See Connecting from a Client for details.

2025-04-25: Server Release, Deprecation, and Upgrade Schedule

- Learn here about the schedule of support for MySQL Server versions by MySQL HeatWave on AWS, the version deprecation and unavailable dates, and other important information on the subject. You can choose between two release tracks, Innovation or Long-Term-Support (LTS)/Bugfix, depending on your application and environment needs.

2025-04-25: High Availability DB Systems with Oracle Cloud Free Tier

- You can now create high availability DB Systems with an Oracle Cloud Free Tier Account. A high availability DB System consists of three MySQL instances, one primary and two secondary. It assures that when the primary instance fails, its function is taken over by one of the secondary instances, with zero data loss and minimal downtime. See High Availability for details.

2025-04-23: MySQL 8.0.42 Now Supported

- MySQL HeatWave on AWS now supports MySQL 8.0.42. See MySQL 8.0.42 Release Notes for more information on the server release.

  MySQL 8.0.42 is a new release of the 8.0 BugFix series, which focuses on security and bug fixes and are recommended for customers who prefer established behavior. The innovation releases will include new features.

  We recommend customers who are using any lower versions of MySQL 8.0 Server to upgrade to MySQL 8.0.42. See Upgrade MySQL Version for instructions.

2025-04-02: MySQL 9.2.2 Now Supported

- MySQL HeatWave on AWS now supports MySQL 9.2.2, the latest Innovation Release of the MySQL Server. If you want to access the latest features and improvements and stay updated with the latest technologies, the MySQL Innovation Releases might be the best choice for you. You can choose release 9.2.2 when creating a new DB System, and for existing DB Systems, you can manually upgrade to MySQL 9.2.2. Refer to Changes in MySQL HeatWave 9.2.2 and Upgrade MySQL Version.
- With MySQL 9.2.2, a MySQL HeatWave Cluster now reloads data from the MySQL HeatWave Storage Layer after a planned or unplanned restart of a Standalone DB System, reducing the recovery time and improving overall system reliability. See MySQL HeatWave Cluster Data Recovery for details.

2025-04-02: MySQL 9.2.1 Now Supported

- MySQL HeatWave on AWS now supports MySQL 9.2.1, the latest Innovation Release of the MySQL Server. If you want to access the latest features and improvements and stay updated with the latest technologies, the MySQL Innovation Releases might be the best choice for you. You can choose release 9.2.1 when creating a new DB System, and for existing DB Systems, you can manually upgrade to MySQL 9.2.1. Refer to Changes in MySQL HeatWave 9.2.1 and Upgrade MySQL Version.

Copyright © 2022, 2025, Oracle and/or its affiliates.

- With MySQL 9.2.1, MySQL HeatWave GenAI supports an expanded list of external LLMs provided by Amazon Bedrock. See MySQL HeatWave GenAI External LLMs for details.

2025-04-01: MySQL HeatWave on AWS Now Available in More Regions

- MySQL HeatWave on AWS is now available in the AWS US West (Oregon) and Asia Pacific (Seoul) regions. See Region Availability.

2025-03-24: New Unload Advisor for the MySQL HeatWave Cluster:

- To help identify unused tables in the MySQL HeatWave Cluster, the MySQL HeatWave Console now provides an **Unload Advisor**, which recommends tables to be unloaded based on the query history. This can help free up memory for other tables and allow for a better use of available resources. See Unload Advisor for details.

2025-03-17: New Summary Panes for Data Loading and Refreshing Issues for MySQL HeatWave Clusters

- If there is a failure in loading data into a MySQL HeatWave Cluster or in refreshing Lakehouse data, the MySQL HeatWave Console now provides details about the failure in a new summary pane. See Loading or Unloading Data into MySQL HeatWave Cluster and Refreshing Lakehouse Data for details.

2025-03-07: New Setting Added for Importing MySQL Dump Files

- A new import setting has been added for importing MySQL dump files: **Maximum bytes per transaction** sets the maximum number of bytes that can be loaded from a MySQL dump file with each `LOAD DATA` statement. Use this setting if you receive an error when importing a file because the file's size exceeds the target server's limits. See Importing Data Using the Data Import Feature for details.

2025-02-18: Server Upgrade Now Supported When Restoring a Backup of a High Availability DB System

- You can now restore the backup of a high availability DB System to a higher server version than the one on which the backup was created. See Restoring a Backup to a New DB System.

2025-02-18: Switchover Now Supported for High Availability DB Systems

- You can now manually promote one of the secondary MySQL instances to be the primary instance in a high availability DB System. When you perform a switchover, the IP address of the DB System (that is, the read write endpoint) does not change, and hence you do not need to make any changes in your application. You can use Switchover to test if your application works correctly during a failover. See Switchover.

2025-02-03: High Availability DB Systems Now Supported

- You can now create a DB System with High Availability, which contains three MySQL instances, a primary and two secondary, deployed across different availability zones. Your data is replicated using MySQL Group Replication. Your application connects to a single endpoint to read and write data to the DB System. In the event of a failure of the primary instance, its function is taken up by one of the secondary instances, so that you do not have to reconfigure your application. You can also restore a backup of a high availability DB System to either a standalone or a high availability DB System.
- You can now create an asynchronous replication channel for a DB System with High Availability enabled, allowing more advanced topologies and enabling easier live

Copyright © 2022, 2025, Oracle and/or its affiliates.

migrations or disaster recovery. For more information, see Inbound Replication to a High Availability DB System.

2025-01-22: MySQL 9.2.0 Now Supported

- MySQL HeatWave on AWS now supports MySQL 9.2.0, the latest Innovation Release of the MySQL Server. If you want to access the latest features and improvements and stay updated with the latest technologies, the MySQL Innovation Releases might be the best choice for you. You can choose release 9.2.0 when creating a new DB System, and for existing DB Systems, you can manually upgrade to MySQL 9.2.0. Refer to Changes in MySQL HeatWave 9.2.0 and Upgrade MySQL Version.

2024-12-18: MySQL 9.1.2 Now Supported

- MySQL HeatWave on AWS now supports MySQL 9.1.2, the latest Innovation Release of the MySQL Server. If you want to access the latest features and improvements and stay updated with the latest technologies, the MySQL Innovation Releases might be the best choice for you. You can choose release 9.1.2 when creating a new DB System, and for existing DB Systems, you can manually upgrade to MySQL 9.1.2. Refer to Changes in MySQL HeatWave 9.1.2 and Upgrade MySQL Version.

2024-11-25: Refreshing Lakehouse Data

- When using MySQL 9.1.0 or later, if the data you loaded into MySQL HeatWave Lakehouse has been altered, or if you want to add similar data that maps to your Lakehouse tables, you can perform a **Lakehouse Incremental Load** using the MySQL HeatWave Console. See Refreshing Lakehouse Data for details.

2024-11-20: MySQL 9.1.1 Now Supported

- MySQL HeatWave on AWS now supports MySQL 9.1.1, the latest Innovation Release of the MySQL Server. If you want to access the latest features and improvements and stay updated with the latest technologies, the MySQL Innovation Releases might be the best choice for you. You can choose release 9.1.1 when creating a new DB System, and for existing DB Systems, you can manually upgrade to MySQL 9.1.1. Refer to Changes in MySQL HeatWave 9.1.1 and Upgrade MySQL Version.

2024-11-06: MySQL 8.4.3 Now Supported

- MySQL HeatWave on AWS now supports MySQL 8.4.3. See MySQL 8.4.3 Release Notes for more information on the server release.

  MySQL 8.4.3 is a new release of the 8.4 LTS (Long Term Support) series, which focuses on security and bug fixes and are recommended for customers who prefer established behavior. The innovation releases will include new features.

  We recommend customers who are using any lower versions of MySQL Server to upgrade to MySQL 8.4.3. See Upgrade MySQL Version for instructions.

2024-10-23: MySQL 9.1.0 Now Supported

- MySQL HeatWave on AWS now supports MySQL 9.1.0, the latest Innovation Release of the MySQL Server. Innovation Release 9.1.0 includes new features and bug fixes. If you want to access the latest features and improvements and stay updated with the latest technologies, the MySQL Innovation Releases might be the best choice for you. You can choose release 9.1.0 when creating a new DB System, and for existing DB Systems, you can manually upgrade to MySQL 9.1.0. Refer to MySQL 9.1.0 Release Notes and Upgrade MySQL Version.

Copyright © 2022, 2025, Oracle and/or its affiliates.

2024-10-16: Enhanced Console Navigation

- The MySQL HeatWave on AWS Console has been enhanced with improved navigation. Page-level navigation is now relocated to a new collapsible sidebar and the **Performance** and **MySQL HeatWave AutoML** tabs are merged into the **Workspace** tab. See Console Overview for details.

2024-09-18: MySQL HeatWave Console Home Screen Enhanced

- The console home screen now has a new **Monitoring** tab. The tab contains a new **Service Events** pane. See Console Overview for details.

2024-09-04: MySQL HeatWave GenAI Now Available

- MySQL HeatWave on AWS now supports MySQL HeatWave GenAI. It provides integrated and automated generative AI with in-database large language models (LLMs), an automated, in-database vector store, and the ability to have contextual conversations in natural language. MySQL HeatWave GenAI enables enterprises to use generative AI without AI expertise, data movement, or additional cost. See MySQL HeatWave GenAI for details.

2024-08-28: MySQL 9.0.1 Now Supported

- MySQL HeatWave on AWS now supports MySQL 9.0.1, the latest Innovation Release of the MySQL Server. Innovation Release 9.0.1 includes new features and bug fixes. If you want to access the latest features and improvements and stay updated with the latest technologies, the MySQL Innovation Releases might be the best choice for you. You can choose release 9.0.1 when creating a new DB System, and for existing DB Systems, you can manually upgrade to MySQL 9.0.1. Refer to MySQL 9.0.1 Release Notes and Upgrade MySQL Version.

2024-08-22: MySQL HeatWave Console Home Screen Refreshed

- The MySQL HeatWave on AWS console now has a new home screen for an enhanced user experience. It has the following improvements:
  - Revamped with key information presented in a dashboard-like layout. There is now a **What's New** pane, listing latest changes to the MySQL HeatWave Console.
  - The **MySQL HeatWave** and **MySQL HeatWave Clusters** tabs are now consolidated into a new **Resources** tab. There is now a **MySQL HeatWave Clusters** tab on the new **Resources** page, for information and actions on the MySQL HeatWave Clusters.
  - The **Workspaces** tab is renamed to **Workspace**.

2024-08-22: PrivateLink Now Supported for Inbound replication

- MySQL HeatWave on AWS now supports inbound replication on a PrivateLink. It enables secure data replication from an external system running in your AWS account to MySQL HeatWave on AWS. See Configuring Inbound Replication over an Egress PrivateLink for details.

2024-08-08: DB System Shapes Now Changeable

- You can now reduce the shape of a DB System to save cost or expand the shape to enhance the DB System's function or performance. Decisions on shape changes are aided by the Autopilot Shape Advisor, which uses the system workload to access the suitability of a shape and make recommendations on shape changes. See Update MySQL Shape for details.

Copyright © 2022, 2025, Oracle and/or its affiliates.

2024-07-10: MySQL 9.0.0 Now Supported

- MySQL HeatWave on AWS now supports MySQL 9.0.0, the latest Innovation Release of the MySQL Server. Innovation Release 9.0.0 includes new features and bug fixes. If you want to access the latest features and improvements and stay updated with the latest technologies, the MySQL Innovation Releases might be the best choice for you. You can choose release 9.0.0 when creating a new DB System, and for existing DB Systems, you can manually upgrade to MySQL 9.0.0. Refer to MySQL 9.0.0 Release Notes and Upgrade MySQL Version.

2024-07-10: Autopilot Index Advisor Now Available

- You can now enhance the performance of OLTP workloads on DB Systems running MySQL Server 9.0.0 or later by utilizing the Autopilot Index Advisor, which provides recommendations for secondary indexes. See Autopilot Index Advisor for details.

2024-06-25: AWS PrivateLink Now Supported

- You can now utilize AWS PrivateLink to establish a secure connection to MySQL HeatWave on AWS from your VPC. This ensures that sensitive data such as customer details remain protected from the internet, enabling secure and private transfer of critical data in a scalable manner. For more information, please refer to PrivateLink.

2024-06-25: MySQL HeatWave Lakehouse Now Available

- MySQL HeatWave on AWS now includes MySQL HeatWave Lakehouse, enabling organizations to process and query hundreds of terabytes of data stored in object storage and optionally combine it with transactional data in MySQL databases without copying the data from the object storage into the MySQL database. Customers can also use this data loaded from the object storage to train machine learning models, run inferences, and explain results, without moving the data to a separate ML service. With Lakehouse, MySQL HeatWave on AWS offers a single service for analytics across data warehouses and data lakes, machine learning, and transaction processing, eliminating the need for ETL across cloud services. For more information, please refer to Creating Lakehouse Mapping.

2024-06-25: Free Trial for MySQL HeatWave on AWS Now Available

- You can now try out MySQL HeatWave on AWS for free without having to upgrade to a paid Oracle Cloud account. See Try MySQL MySQL HeatWave for free and Sign-up Overview for details.

2024-05-22: MySQL 8.4.0 Now Supported

- MySQL HeatWave on AWS now supports MySQL 8.4.0. See MySQL 8.4.0 Release Notes for more information on the server release.

  MySQL 8.4.0 marks the start of a new series of LTS (Long Term Support) releases. The 8.4 LTS releases will focus on security and bug fixes and are recommended for customers who prefer established behavior, while the next innovation series will include new features.

  We recommend customers who are using any lower versions of MySQL Server to upgrade to MySQL 8.4.0. See Upgrade MySQL Version for instructions.

2024-05-02: Pre-configured DB System Now Available

- You can now explore the capabilities of MySQL HeatWave on AWS by launching a pre-configured DB System with sample data. See Launching a Starter DB System for details.

Copyright © 2022, 2025, Oracle and/or its affiliates.

2024-04-08: Sample Database Now Available

- You can now explore MySQL HeatWave on AWS features using sample databases. See Importing Sample Database for details.

2024-03-04: Events Tab Added to the Console

- Events allow you to monitor the state changes of resources such as DB Systems, MySQL HeatWave Clusters, and Backups. An event reports a create, update, or delete operation for a resource, or its lifecycle state change. Events are listed under the **Events** tab of the corresponding resource. For more details, see Events.

2024-02-20: Custom MySQL Configurations Can Now be Created

- You can now create a custom configuration by copying an existing MySQL configuration and modifying its user variables. You can then apply the custom configuration to a DB System. For more information, see Copying a MySQL Configuration and Update MySQL Configuration.

2024-02-15: MySQL Enterprise Audit Now Supported

- MySQL HeatWave on AWS now supports the MySQL Enterprise Audit plugin. The audit plugin enables the MySQL Server to produce a log file containing an audit record of server activities such as when clients connected and disconnected, what actions they performed while being connected, and which databases and tables they accessed. You can also add statistics for the time and size of the queries performed. See MySQL Enterprise Audit for details.

2024-01-24: MySQL 8.3.0 Now Supported

- MySQL HeatWave on AWS now supports MySQL version 8.3.0.

2024-01-17: Inbound Replication Support Now Further Enhanced

- MySQL HeatWave on AWS supports inbound replication that does not use GTID auto-positioning, and also for tables that do not have primary keys. See Creating a Channel for details.

2023-12-21: Auto Shape Prediction Now Enhanced

- The Auto Shape Prediction feature in MySQL Autopilot has been enhanced. In addition to the overall buffer pool usage, workload activity, and access patterns, Auto Shape Prediction now utilizes the CPU statistics to estimate the required buffer pool size and CPU cores. See Autopilot Shape Advisor for details.

2023-11-28: Inbound Replication Support Now Enhanced

- MySQL HeatWave on AWS supports adding filters in inbound replication. Filters allows you to selectively replicate databases and tables from the MySQL source. For more information on inbound replication filters, please see Channel Filter Rules for Inbound Replication.
- MySQL HeatWave on AWS introduces delayed inbound replication. You can choose to specify the time (in seconds) to wait before the target DB System in MySQL HeatWave on AWS replicates the source transactions. For more information on delayed replication, please see Replication delay.

Copyright © 2022, 2025, Oracle and/or its affiliates.

2023-11-16: Inbound Replication Now Supported

- MySQL HeatWave on AWS supports inbound replication from a MySQL source to a MySQL DB System.
  For more information on the inbound replication feature, see Inbound Replication.
- You can now update a MySQL HeatWave Cluster's size and shape. For more information, see Editing a MySQL HeatWave Cluster.

2023-10-26: MySQL 8.2.0 Now Supported

- MySQL HeatWave on AWS supports MySQL version 8.2.0.

2023-09-13: Improved Data Import Methods Now Supported

- MySQL HeatWave on AWS introduces a faster and easier feature to import data from an Amazon S3 bucket into a MySQL DB System. The data import feature enables you to import data in a variety of formats such as MySQL dump and text files (such as CSV and TSV).
  For more information on the data import feature, see Data Import Feature.
- MySQL HeatWave on AWS supports the bulk ingest feature to import text files (such as CSV and TSV) directly from an Amazon S3 bucket into a MySQL DB System. This method is faster and more efficient in terms of computing and storage consumption. To use the bulk ingest feature, first connect to the DB system, and then import data in an existing or new table using the BULK algorithm.
  For more information on the bulk ingest feature, see Bulk Ingest Feature.

2023-08-08: Data Storage Size Can Now Be Increased

- 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.

2023-07-31: Auto Error Recovery Now Supported

- MySQL HeatWave on AWS now supports auto error recovery. Whenever a MySQL HeatWave node fails because of a hardware or a software issue, the cluster becomes unhealthy and error recovery is triggered. During the recovery process, MySQL 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 MySQL HeatWave Cluster Data Recovery.

2023-07-13: MySQL HeatWave on AWS Now Available in More Regions

- MySQL HeatWave on AWS is now available in the AWS Europe (Frankfurt) and Europe (London) region. See Region Availability.

2023-07-13: Fast Data Reload Now Available

- Fast data reload is now available when you pause and resume MySQL HeatWave on AWS. Besides storing the MySQL HeatWave formatted data in-memory, MySQL HeatWave also stores the formatted data in MySQL 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 MySQL HeatWave Cluster. You can now pause the MySQL HeatWave Cluster when you don't need it to save cost, and resume the cluster very fast when you want to use MySQL HeatWave for query acceleration.

Copyright © 2022, 2025, Oracle and/or its affiliates.

2023-06-08: MySQL HeatWave on AWS Now Available in the AWS Asia Pacific (Mumbai) Region

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

2023-05-04: MySQL HeatWave on AWS Now Available in the AWS Asia Pacific (Tokyo) Region

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

2023-04-27: New Features Now Availability

- MySQL 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 MySQL HeatWave in-memory hybrid columnar representation of the data. This allows data to be reloaded to each MySQL 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 MySQL HeatWave Architectural Features.
- MySQL Autopilot: Auto Error Recovery from MySQL failure
  With Auto Error Recovery, now when MySQL fails and restarts, the MySQL 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 MySQL HeatWave Cluster after MySQL upgrade
  MySQL 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 MySQL 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.

Copyright © 2022, 2025, Oracle and/or its affiliates.