aws

Release Notes for Aurora MySQL

Amazon Aurora

Amazon Aurora: Release Notes for Aurora MySQL

Table of Contents

Release notes for Amazon Aurora MySQL-Compatible Edition

Amazon Aurora MySQL-Compatible Edition releases updates regularly. Updates are applied to Aurora MySQL DB clusters during system maintenance windows. The timing when updates are applied depends on the AWS Region and maintenance window setting for the DB cluster, and also the type of update.

Amazon Aurora MySQL releases are made available to all AWS Regions over the course of multiple days. Some Regions might temporarily show an engine version that isn't available in a different Region yet.

Updates are applied to all instances in a DB cluster at the same time. An update requires a database restart on all instances in a DB cluster. So you experience 20 to 30 seconds of downtime, after which you can resume using your DB cluster or clusters. You can view or change your maintenance window settings from the AWS Management Console.

Topics

Release calendars for Amazon Aurora MySQL

The release calendars on this page can help you plan your major and minor version upgrades. For more information on Amazon Aurora upgrades, versioning, and lifecycle, see Amazon Aurora versions.

Topics

Release calendar for Aurora MySQL major versions

Aurora MySQL major versions are available under standard support at least until community end of life for the corresponding community version. You can continue running a major version past its Aurora end of standard support date for a fee. For more information, see Using Amazon RDS Extended Support and Amazon Aurora pricing.

Aurora MySQL currently supports the following major versions.

Communit major version	Aurora major version	Communit end of life date	Aurora end of standard support date	RDS start of Extended Support year 1 pricing date	RDS start of Extended Support year 3 pricing date	RDS end of Extended Support date	Minor versions eligible for Extended Support
MySQL 5.6 (deprecat ed)	Aurora MySQL version 1 (deprecat ed)	5 February 2021	28 February 2023	N/A	N/A	N/A	N/A
MySQL 5.7	Aurora MySQL version 2	October 2023	31 October 2024	1 December 2024	N/A	28 February 2027	Aurora MySQL

Communit major version	Aurora major version	Communit end of life date	Aurora end of standard support date	RDS start of Extended Support year 1 pricing date	RDS start of Extended Support year 3 pricing date	RDS end of Extended Support date	Minor versions eligible for Extended Support
							2.11 and 2.12
MySQL 8.0	Aurora MySQL version 3	April 2026	30 April 2027	1 May 2027	N/A	31 July 2029	To be determined

Note

Amazon RDS Extended Support for Aurora MySQL version 2 starts on November 1, 2024, but you won't be charged until December 1, 2024. Between November 1 and November 30, 2024, all Aurora MySQL version 2 DB clusters are covered under Amazon RDS Extended Support.

Release calendar for Aurora MySQL minor versions

Aurora MySQL currently supports the following minor versions.

Aurora MySQL version	Aurora MySQL release date	Aurora MySQL end of standard support date
3.07 (Compatible with Community MySQL 8.0.36)	June 4, 2024	August 4, 2025
3.06 (Compatible with Community MySQL 8.0.34)	March 7, 2024	May 31, 2025

Aurora MySQL version	Aurora MySQL release date	Aurora MySQL end of standard support date
3.05 (Compatible with Community MySQL 8.0.32)	October 25, 2023	January 31, 2025
3.04 (Compatible with Community MySQL 8.0.28) (LTS)	July 31, 2023	October 31, 2026
3.03 (Compatible with Community MySQL 8.0.26)	March 1, 2023	August 15, 2024
2.12[1] (Compatible with Community MySQL 5.7.40 or 5.7.44[2])	July 25, 2023	October 31, 2024
2.11[1] (Compatible with Community MySQL 5.7.12)	October 25, 2022	October 31, 2024
2.07 (Compatible with Community MySQL 5.7.12)	November 25, 2019	April 30, 2024

LTS – Aurora MySQL long-term support (LTS) versions. For more information, see Aurora MySQL long-term support (LTS) release.

[1] This minor version will continue to be available when the major version is in Amazon RDS Extended Support. For more information, see Amazon Aurora major versions.

[2] Aurora MySQL 2.12 versions through 2.12.1 are compatible with MySQL version 5.7.40, and versions 2.12.2 and higher are compatible with MySQL version 5.7.44.

Database engine updates for Amazon Aurora MySQL version 3

The following are database engine updates for Amazon Aurora MySQL version 3.

Topics

- Aurora MySQL database engine updates 2023-05-11 (version 3.03.1, compatible with MySQL 8.0.26)
- Aurora MySQL database engine updates 2023-03-01 (version 3.03.0, compatible with MySQL 8.0.26) Upgrades to this version aren't supported.
- Aurora MySQL database engine updates 2023-04-17 (version 3.02.3, compatible with MySQL 8.0.23) The end of standard support is January 15, 2024.
- Aurora MySQL database engine updates 2022-11-18 (version 3.02.2, compatible with MySQL 8.0.23) The end of standard support is January 15, 2024.
- Aurora MySQL database engine updates 2022-09-07 (version 3.02.1, compatible with MySQL 8.0.23) The end of standard support is January 15, 2024. Upgrades to this version aren't supported.
- Aurora MySQL database engine updates 2022-04-20 (version 3.02.0, compatible with MySQL 8.0.23) The end of standard support is January 15, 2024. Upgrades to this version aren't supported.
- Aurora MySQL database engine updates 2022-04-15 (version 3.01.1, compatible with MySQL 8.0.23) The end of standard support is January 15, 2024. Upgrades to this version aren't supported.
- Aurora MySQL database engine updates 2021-11-18 (version 3.01.0, compatible with MySQL 8.0.23) The end of standard support is January 15, 2024. Upgrades to this version aren't supported.

Aurora MySQL database engine updates 2024-06-04 (version 3.07.0, compatible with MySQL 8.0.36)

Version: 3.07.0

Aurora MySQL 3.07.0 is generally available. Aurora MySQL 3.07 versions are compatible with MySQL 8.0.36. For more information on the community changes that have occurred, see MySQL 8.0 Release Notes.

For details of the new features in Aurora MySQL version 3, see Aurora MySQL version 3 compatible with MySQL 8.0. For differences between Aurora MySQL version 3 and Aurora MySQL version 2, see Comparison of Aurora MySQL version 2 and Aurora MySQL version 3. For a comparison of Aurora MySQL version 3 and MySQL 8.0 Community Edition, see Comparison of Aurora MySQL version 3 and MySQL 8.0 Community Edition in the *Amazon Aurora User Guide*.

Currently supported Aurora MySQL releases are 2.07.9, 2.07.10, 2.11.*, 2.12.*, 3.03.*, 3.04.*, 3.05.*, 3.06.*, and 3.07.*.

You can perform an in-place upgrade, restore a snapshot, or initiate a managed blue/green upgrade using Amazon RDS Blue/Green Deployments from any currently supported Aurora MySQL version 2 cluster into an Aurora MySQL version 3.07.0 cluster.

For information on planning an upgrade to Aurora MySQL version 3, see Planning a major version upgrade for an Aurora MySQL cluster. For general information about Aurora MySQL upgrades, see Upgrading Amazon Aurora MySQL DB clusters in the *Amazon Aurora User Guide*.

For troubleshooting information, see Troubleshooting for Aurora MySQL in-place upgrade in the *Amazon Aurora User Guide*.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Improvements

Fixed security issues and CVEs:

- Enabled support for FIPS-validated cryptography, a fully owned AWS implementation. For more information, see AWS-LC is now FIPS 140-3 certified on AWS Security Blog.

This release includes all community CVE fixes up to and including MySQL 8.0.36. The following CVE fixes are included:

- CVE-2020-11104
- CVE-2020-11105
- CVE-2023-38545
- CVE-2023-38546
- CVE-2023-39975

Availability improvements:

- Fixed an issue that can cause a reader DB instance to restart when reading a table that is being altered or dropped on the writer DB instance.

- Fixed an issue that can cause an Aurora MySQL writer DB instance to restart when a write forwarding session is closed while running a forwarded query.
- Fixed an issue that causes a DB instance to restart when handling large GTID sets on a binary log–enabled instance.
- Fixed an issue when processing `INSERT` queries on InnoDB partitioned tables that can cause a gradual decline of free memory in the instance.
- Fixed an issue that, in rare conditions, can cause the reader DB instances to restart.
- Fixed an issue that can cause a database instance to restart when running SHOW STATUS and PURGE BINARY LOGS statements concurrently. `PURGE BINARY LOGS` is a managed statement that is run to honor the user-configured binlog retention period.
- Fixed an issue that can cause the server to unexpectedly close after running Data Manipulation Language (DML) statements on a table whose nonvirtual columns were reordered with a `MODIFY COLUMN` or `CHANGE COLUMN` statement.
- Fixed an issue that, during the restart of a database instance, can cause an additional restart.
- Fixed an issue that can cause a reader DB instance that uses write forwarding to restart when a forwarded implicit commit statement encounters an error.
- Fixed an issue that, in rare conditions, can cause a reader instance to restart when performing `SELECT` queries on tables with a foreign key constraint.
- Fixed an issue where DB instances using multi-TB Aurora cluster volumes can experience increased downtime during restart due to InnoDB buffer pool validation failures.
- Fixed an issue that can cause a database to restart when a cascading `UPDATE` or `DELETE` foreign key constraint is defined on a table where a virtual column is involved either as a column in the foreign key constraint, or as a member of the referenced table.
- Fixed an issue that can interrupt database recovery during startup if the restart occurred while running heavy insert operations involving `AUTO_INCREMENT` columns.
- Fixed an issue in Aurora Serverless v2 that can lead to a database restart while scaling up.

General improvements:

- Reduced I/O usage and improved performance for a subset of primary key range scan queries that employ parallel query.
- Aurora MySQL version 3.06.0 added support for Amazon Bedrock integration. As part of this, new reserved keywords (`accept`, `aws_bedrock_invoke_model`, `aws_sagemaker_invoke_endpoint`, `content_type`, and `timeout_ms`) were added. In

Aurora MySQL version 3.07.0, these keywords have been changed to nonreserved keywords, which are permitted as identifiers without quoting. For more information on how MySQL handles reserved and nonreserved keywords, see Keywords and reserved words in the MySQL documentation.

- Fixed an issue that didn't clearly return an error message to the client when invoking the Amazon Bedrock service from an Aurora MySQL DB cluster in an AWS Region where Amazon Bedrock isn't yet available.
- Fixed an issue that can cause excessive memory consumption when querying `BLOB` columns using Aurora parallel query.
- Added support for the `connection_memory_limit` and `connection_memory_chunk_size` parameters to be set at the session level to behave the same as in MySQL Community Edition. The `connection_memory_limit` is used to set the maximum amount of memory that can be used by a single user connection. The `connection_memory_chunk_size` parameter can be used to set the chunking size for updates to the global memory usage counter.
- Fixed an issue where the user is unable to interrupt any query or set session timeouts for `performance_schema` queries.
- Fixed an issue where binary log (binlog) replication configured to use custom SSL certificates (mysql.rds_import_binlog_ssl_material) could fail when the replication instance is undergoing a host replacement.
- Added the `Aurora_fts_cache_memory_used` global status variable to track memory usage for the full-text search system across all tables. For more information, see Aurora MySQL global status variables in the *Amazon Aurora User Guide*.
- Fixed an issue where an Amazon Redshift cluster configured as a zero-ETL destination might experience a temporary increase in IntegrationLag when an Amazon Aurora MySQL DB cluster is configured as a binary log replica, with Enhanced Binlog and zero-ETL integration enabled.
- Fixed an issue related to audit log file management that can cause log files to be inaccessible for download or rotation, and in some cases increase CPU usage.
- Optimized `AUTO_INCREMENT` key recovery to reduce the completion time for restoring snapshots, performing point-in-time recovery, and cloning DB clusters with large numbers of tables in the database.
- Fixed an issue where the wait/io/redo_log_flush event wasn't shown in the Performance Schema wait event summary tables.
- Fixed an issue that can cause duplicate key errors for `AUTO_INCREMENT` columns using descending indices after a snapshot restore, backtrack, or database cloning operation.

- Fixed an issue that can cause a writer DB instance to restart when a reader DB instance using write forwarding runs a Data Manipulation Language (DML) statement that contains a timestamp value and the `time_zone` database parameter is set to `UTC`.
- Fixed an issue where a `SELECT` query on an Aurora reader instance might fail with the error table doesn't exist when the table has at least one full-text search (FTS) index and a `TRUNCATE` statement is being run on the Aurora writer DB instance.
- Fixed an issue that, in rare cases, causes zero downtime patching (ZDP) to fail.
- Fixed an issue that can cause an incomplete result set when running queries involving `LEFT JOIN` or `RIGHT JOIN` operations using the hash join algorithm with parallel query.

Upgrades and migrations:

- Fixed an issue that can cause upgrade failures from Aurora MySQL version 2 to Aurora MySQL version 3 when there is a user-defined `FTS_DOC_ID` column present in the table schema.
- Fixed an issue that can cause upgrade failures from Aurora MySQL version 2 to Aurora MySQL version 3 due to a synchronization issue while processing InnoDB tablespaces.
- Fixed an issue that can cause major version upgrades to Aurora MySQL version 3 to fail due to the presence of orphaned entries for already deleted tablespaces in InnoDB system tables in Aurora MySQL version 2.
- Fixed an issue where the SERVER_ID value wasn't updated after an Amazon RDS Blue/Green Deployment switchover. This led to issues where smart drivers such as the Amazon Web Services (AWS) JDBC Driver were unable to discover the DB cluster topology after a blue/green switchover. With this fix, Aurora DB clusters renamed as part of an RDS Blue/Green Deployment, that are running on Aurora MySQL version 3.07 and higher, will have the `SERVER_ID` value updated as part of the switchover. For earlier versions, the DB instances in the blue and green clusters can be rebooted to update the `SERVER_ID` value.

Integration of MySQL Community Edition bug fixes

This release includes all community bug fixes up to and including 8.0.36, in addition to the following. For more information, see MySQL bugs fixed by Aurora MySQL 3.x database engine updates.

- Fixed an issue where cache line value can be calculated incorrectly, causing a failure during database restart on Graviton-based instances. (Community Bug Fix #35479763)

- Fixed an issue where some instances of subqueries within stored routines were not handled correctly. (Community Bug Fix #35377192)
- Fixed an issue that can cause higher CPU usage due to background TLS certificate rotation (Community Bug Fix #34284186).
- Fixed an issue where InnoDB allowed the addition of `INSTANT` columns to tables in the MySQL system schema in Aurora MySQL versions lower than 3.05, which could lead to the server unexpectedly closing (database instance restarting) after upgrading to Aurora MySQL version 3.05.0. (Community Bug Fix #35625510).

Aurora MySQL database engine updates 2024-06-26 (version 3.06.1, compatible with MySQL 8.0.34)

Version: 3.06.1

Aurora MySQL 3.06.1 is generally available. Aurora MySQL 3.06 versions are compatible with MySQL 8.0.34. For more information on the community changes that have occurred, see MySQL 8.0 Release Notes.

For details of the new features in Aurora MySQL version 3, see Aurora MySQL version 3 compatible with MySQL 8.0. For differences between Aurora MySQL version 3 and Aurora MySQL version 2, see Comparison of Aurora MySQL version 2 and Aurora MySQL version 3. For a comparison of Aurora MySQL version 3 and MySQL 8.0 Community Edition, see Comparison of Aurora MySQL version 3 and MySQL 8.0 Community Edition in the *Amazon Aurora User Guide*.

Currently supported Aurora MySQL releases are 2.07.9, 2.07.10, 2.11.*, 2.12.*, 3.03.*, 3.04.*, 3.05.*, 3.06.*, and 3.07.*.

You can perform an in-place upgrade, restore a snapshot, or initiate a managed blue/green upgrade using Amazon RDS Blue/Green Deployments from any currently supported Aurora MySQL version 2 cluster into an Aurora MySQL version 3.06.1 cluster.

For information on planning an upgrade to Aurora MySQL version 3, see Planning a major version upgrade for an Aurora MySQL DB cluster. For general information about Aurora MySQL upgrades, see Upgrading Amazon Aurora MySQL DB clusters in the *Amazon Aurora User Guide*.

For troubleshooting information, see Troubleshooting for Aurora MySQL in-place upgrade in the *Amazon Aurora User Guide*.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Improvements

Fixed security issues and CVEs:

This release includes all community CVE fixes up to and including MySQL 8.0.34. The following CVE fixes are included:

- CVE-2023-44487
- CVE-2024-0853

Availability improvements:

- Fixed an issue that causes an Aurora MySQL DB instance to restart when running a parallel query.
- Fixed an issue that can cause a reader DB instance to restart when reading a table that is being altered or dropped on the writer DB instance.
- Fixed an issue that caused a memory access violation leading to releasing a mutex object no longer owned by the thread.
- Fixed an issue that can cause an Aurora MySQL writer DB instance to restart when a write forwarding session is closed while running a forwarded query.
- Fixed an issue that causes a DB instance restart when handling large GTID sets on a binary log–enabled instance.
- Fixed an issue that, in rare conditions, can cause a reader instance to restart when performing SELECT queries on tables with a foreign key constraint.
- Fixed an issue that causes a DB instance to restart when attempting to recover the InnoDB data dictionary during database recovery.
- Fixed an issue in Aurora Serverless v2 that can lead to a database restart while scaling up.

General improvements:

- Fixed an issue in metrics publishing code where memory might be used after being freed.
- Fixed an issue that caused repeated DB engine restarts due to a nonexistent undo tablespace object.

- Fixed an issue with automatic truncation of undo tablespaces when they're larger than the threshold innodb_max_undo_log_size in upgrade scenarios.
- Fixed an issue that provided an incorrect value for the `threads_running` status variable when using Aurora Global Database.
- Fixed an issue where an Aurora MySQL binary log (binlog) read replica with parallel secondary index optimization enabled would experience a restart when applying replication changes on tables with foreign keys.
- Aurora MySQL version 3.06.0 added support for Amazon Bedrock integration. As part of this, new reserved keywords (`accept`, `aws_bedrock_invoke_model`, `aws_sagemaker_invoke_endpoint`, `content_type`, and `timeout_ms`) were added. In Aurora MySQL version 3.06.1, these keywords have been changed to nonreserved keywords, which are permitted as identifiers without quoting. For more information on how MySQL handles reserved and nonreserved keywords, see Keywords and reserved words in the MySQL documentation.
- Fixed an issue that didn't clearly return an error message to the client when invoking the Amazon Bedrock service from an Aurora MySQL DB cluster in an AWS Region where Amazon Bedrock isn't yet available.
- Fixed an issue that causes a DB instance to restart because of inaccurate lock holder information in `rw_lock` when using parallel reads.
- Fixed an issue that can cause a DB instance to restart when `SHOW VOLUME STATUS` is run.
- Fixed a memory management issue that led to a decrease in freeable memory over time when running `SELECT ... INTO OUTFILE ...` queries.
- Added support for the `connection_memory_limit` and `connection_memory_chunk_size` parameters to be set at the session level to behave similar to corresponding functionality in MySQL Community Edition. The `connection_memory_limit` parameter sets the maximum amount of memory that can be used by a single user connection. The `connection_memory_chunk_size` parameter sets the chunking size for updates to the global memory usage counter.
- Fixed an issue that can cause a DB instance to restart when the local storage on the DB instance reaches full capacity.
- Fixed an issue where the Performance Schema wasn't enabled when Performance Insights automated management was turned on for db.t4g.medium and db.t4g.large DB instances.

- Fixed an issue that can cause a writer DB instance to restart when a reader DB instance using write forwarding runs a Data Manipulation Language (DML) statement that contains a timestamp value and the `time_zone` database parameter is set to UTC.
- Fixed an issue during zero downtime patching (ZDP) that prevents a DB instance from closing client connections upon reaching the customer-configured minimum value of either `wait_timeout` or `interactive_timeout`.

Upgrades and migrations:

- Fixed an issue that causes upgrades or migrations to fail when the target Aurora MySQL DB engine version is 3.04.0 or higher. This occurs when the `lower_case_table_names` DB cluster parameter is set to 1, and MySQL database collation is incompatible with lowercase table names.

Integration of MySQL Community Edition bug fixes

This release includes all community bug fixes up to and including 8.0.34. For more information, see MySQL bugs fixed by Aurora MySQL 3.x database engine updates.

Aurora MySQL database engine updates 2024-03-07 (version 3.06.0, compatible with MySQL 8.0.34)

Version: 3.06.0

Aurora MySQL 3.06.0 is generally available. Aurora MySQL 3.06 versions are compatible with MySQL 8.0.34. For more information on the community changes that have occurred, see MySQL 8.0 Release Notes.

For details of the new features in Aurora MySQL version 3, see Aurora MySQL version 3 compatible with MySQL 8.0. For differences between Aurora MySQL version 3 and Aurora MySQL version 2, see Comparison of Aurora MySQL version 2 and Aurora MySQL version 3. For a comparison of Aurora MySQL version 3 and MySQL 8.0 Community Edition, see Comparison of Aurora MySQL version 3 and MySQL 8.0 Community Edition in the *Amazon Aurora User Guide*.

Currently supported Aurora MySQL releases are 2.07.9, 2.07.10, 2.11.*, 2.12.*, 3.03.*, 3.04.*, 3.05.*, and 3.06.*.

You can perform an in-place upgrade, restore a snapshot, or initiate a managed blue/green upgrade using Amazon RDS Blue/Green Deployments from any currently supported Aurora MySQL version 2 cluster into an Aurora MySQL version 3.06.0 cluster.

For information on planning an upgrade to Aurora MySQL version 3, see Planning a major version upgrade for an Aurora MySQL DB cluster. For general information about Aurora MySQL upgrades, see Upgrading Amazon Aurora MySQL DB clusters in the *Amazon Aurora User Guide*.

For troubleshooting information, see Troubleshooting for Aurora MySQL in-place upgrade in the *Amazon Aurora User Guide*.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

New features

- Aurora MySQL version 3.06.0 supports Amazon Bedrock integration and introduces the new reserved keywords `accept`, `aws_bedrock_invoke_model`, `aws_sagemaker_invoke_endpoint`, `content_type`, and `timeout_ms`. Check the object definitions for the usage of the new reserved keywords before upgrading to version 3.06.0. To mitigate the conflict with the new reserved keywords, quote the reserved keywords used in the object definitions. For more information on the Amazon Bedrock integration and handling the reserved keywords, see What is Amazon Bedrock? in the *Amazon Aurora User Guide*. For additional information, see Keywords and Reserved Words, The INFORMATION_SCHEMA KEYWORDS Table, and Schema Object Names in the MySQL documentation.
- Improved performance for binary log replicas when replicating transactions for large tables with more than one secondary index. This feature introduces a thread pool to apply secondary index changes in parallel on a binlog replica. The feature is controlled by the `aurora_binlog_replication_sec_index_parallel_workers` DB cluster parameter, which controls the total number of parallel threads available to apply the secondary index changes. For more information, see Optimizing binary log replication in the *Amazon Aurora User Guide*.
- Added a new stored procedure `mysql.rds_set_read_only` that allows changing the value of the global system variable `read_only` on database instances in your Aurora MySQL cluster. For more information, see Replicating in the *Amazon Aurora User Guide*.

- Added a new stored procedure `mysql.rds_set_binlog_source_ssl` that allows setting the encryption on a binary log replica by specifying a value for `SOURCE_SSL`. For more information, see Replicating in the *Amazon Aurora User Guide*.
- Amazon Aurora Machine Learning is an optimized integration between the Aurora MySQL database and AWS machine learning (ML) services. Amazon Bedrock is now supported, allowing you to invoke machine learning models in Amazon Bedrock directly from your Aurora MySQL DB cluster using SQL. For more information on using Amazon Bedrock with your Aurora MySQL DB cluster, see Using Amazon Aurora machine learning with Aurora MySQL in the *Amazon Aurora User Guide*.
- Aurora MySQL version 3.06 adds support for automated undo tablespace truncation. This optimization allows you to reclaim unused space in undo tablespaces after the undo logs have been purged.

Improvements

Fixed security issues and CVEs:

The following CVE fixes are included in this release:

- CVE-2020-11104
- CVE-2020-11105
- CVE-2023-38545
- CVE-2023-38546
- CVE-2023-39975

Availability improvements:

- Fixed an issue where a read replica DB instance can't be launched successfully when there's high workload in the writer DB instance.
- Fixed an issue where an Aurora MySQL writer DB instance can fail over due to a defect in communication with Aurora storage. The defect occurs as a result of a breakdown in the communication between the DB instance and underlying storage following a software update of the Aurora storage instance.
- Fixed an issue when processing `INSERT` queries on InnoDB partitioned tables that can cause a gradual decline of free memory in the instance.

- Fixed an issue that can cause an Aurora MySQL DB instance to restart or fail over due to a decrease in freeable memory when hash join is used while running queries.
- Fixed an issue that can cause a database instance restart when running SHOW STATUS and PURGE BINARY LOGS statements concurrently. `PURGE BINARY LOGS` is a managed statement that is run to honor the user-configured binlog retention period.
- Fixed an issue that can cause the server to unexpectedly close after running Data Manipulation Language (DML) statements on a table whose nonvirtual columns were reordered with a `MODIFY COLUMN` or `CHANGE COLUMN` statement.
- Fixed an issue that, during the restart of a database instance, can cause an additional restart.
- Fixed an issue that can cause a database restart when a cascading `UPDATE` or `DELETE` foreign key constraint is defined on a table where a virtual column is involved either as a column in the foreign key constraint, or as a member of the referenced table.
- In Aurora MySQL 2.10, we added support for rebooting an Aurora DB cluster with read availability. This feature allows reader DB instances to stay online while a writer DB instance is rebooted. This feature is now supported on secondary AWS Regions in Aurora MySQL global databases, ensuring that you can still serve read requests during a writer instance restart on the primary cluster. Previously, when a writer instance restarted, all reader instances in an Aurora MySQL secondary cluster also restarted. With this release, secondary cluster reader instances continue to serve read requests during a writer instance restart, improving read availability in the cluster. For more information, see Rebooting an Aurora cluster with read availability.
- Fixed an issue that can interrupt database recovery during startup if the restart occurred while running heavy insert operations involving `AUTO_INCREMENT` columns.

General improvements:

- Fixed an issue that can cause a parallel query to fail due to transient network issues while reading data from the Aurora cluster volume.
- Fixed an issue where the user is unable to interrupt any query or set session timeouts for `performance_schema` queries.
- Fixed an issue where binary log (binlog) replication configured to use custom SSL certificates (mysql.rds_import_binlog_ssl_material) could fail when the replication instance is undergoing a host replacement.
- Small DB instances with less than or equal to 4 GiB of memory now close the top memory-consuming connections when the DB instance is under memory pressure. You can also tune

the buffer pool to decrease its size. For more information, see Amazon Aurora MySQL out-of-memory issues in the *Amazon Aurora User Guide*.

- Changed the default response for `aurora_oom_response`, on all DB instance classes that have more than 4 GiB of memory, from empty to `print`. For more information, see Amazon Aurora MySQL out-of-memory issues in the *Amazon Aurora User Guide*.
- Fixed an issue related to audit log file management that can cause log files to be inaccessible for download or rotation, and in some cases increase CPU usage.
- Optimized `AUTO_INCREMENT` key recovery to reduce the completion time for restoring snapshots, performing point-in-time recovery, and cloning DB clusters with large numbers of tables in the database.
- Fixed an issue where the wait/io/redo_log_flush event wasn't shown in the Performance Schema wait event summary tables.
- Added the `Aurora_lockmgr_memory_used` and `Aurora_lockmgr_buffer_pool_memory_used` metrics to track the lock manager's memory usage. For more information, see Aurora MySQL global status variables in the *Amazon Aurora User Guide*.
- Fixed an issue where small read replica instances can experience increased replication lag after upgrading from Aurora MySQL versions lower than 2.11.*.
- Fixed an issue that can cause duplicate key errors for `AUTO_INCREMENT` columns using descending indices after a snapshot restore, backtrack, or database cloning operation.
- Fixed an issue where a `SELECT` query on an Aurora reader instance might fail with the error table doesn't exist when the table has at least one full-text search (FTS) index and a `TRUNCATE` statement is being run on the Aurora writer DB instance.
- Fixed an issue that can cause an incomplete result set when running queries involving `LEFT JOIN` or `RIGHT JOIN` operations using the hash join algorithm with parallel query.

Upgrades and migrations:

- Fixed an issue that can cause major version upgrades to fail if there is a user-defined `FTS_DOC_ID` column present in the table schema.
- Fixed an issue that can cause upgrade failures from Aurora MySQLversion 2 to Aurora MySQL version 3 due to a synchronization issue while processing InnoDB tablespaces.

- Fixed an issue that can cause major version upgrades to Aurora MySQL version 3 to fail due to the presence of orphaned entries for already deleted tablespaces in InnoDB system tables in Aurora MySQL version 2.

Integration of MySQL Community Edition bug fixes

This release includes all community bug fixes up to and including 8.0.34, in addition to the following. For more information, see MySQL bugs fixed by Aurora MySQL 3.x database engine updates.

- Fixed an issue where cache line value can be calculated incorrectly, causing a failure during database restart on Graviton-based instances. (Community Bug Fix #35479763)
- Fixed an issue where some instances of subqueries within stored routines were not always handled correctly. (Community Bug Fix #35377192)
- Fixed an issue that can cause higher CPU usage due to background TLS certificate rotation (Community Bug Fix #34284186).
- Fixed an issue where InnoDB allowed the addition of `INSTANT` columns to tables in the MySQL system schema in Aurora MySQL versions lower than 3.05, which could lead to the server unexpectedly closing (database instance restarting) after upgrading to Aurora MySQL version 3.05.0. (Community Bug Fix #35625510).

Aurora MySQL database engine updates 2024-01-31 (version 3.05.2, compatible with MySQL 8.0.32) Default

Version: 3.05.2

Aurora MySQL 3.05.2 is generally available. Aurora MySQL 3.05 versions are compatible with MySQL 8.0.32. For more information on the community changes that have occurred, see MySQL 8.0 Release Notes.

For details of the new features in Aurora MySQL version 3, see Aurora MySQL version 3 compatible with MySQL 8.0. For differences between Aurora MySQL version 3 and Aurora MySQL version 2, see Comparison of Aurora MySQL version 2 and Aurora MySQL version 3. For a comparison of Aurora MySQL version 3 and MySQL 8.0 Community Edition, see Comparison of Aurora MySQL version 3 and MySQL 8.0 Community Edition in the *Amazon Aurora User Guide*.

Currently supported Aurora MySQL releases are 2.07.9, 2.07.10, 2.11.*, 2.12.*, 3.03.*, 3.04.*, and 3.05.*.

You can perform an in-place upgrade, restore a snapshot, or initiate a managed blue/green upgrade using Amazon RDS Blue/Green Deployments from any currently supported Aurora MySQL version 2 cluster into an Aurora MySQL version 3.05.2 cluster.

For information on planning an upgrade to Aurora MySQL version 3, see Upgrade planning for Aurora MySQL version 3. For general information about Aurora MySQL upgrades, see Upgrading Amazon Aurora MySQL DB clusters in the *Amazon Aurora User Guide*.

For troubleshooting information, see Troubleshooting upgrade issues with Aurora MySQL version 3 in the *Amazon Aurora User Guide*.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Improvements

Fixed security issues and CVEs:

The following CVE fixes are included in this release:

- CVE-2020-11104
- CVE-2020-11105
- CVE-2023-38545
- CVE-2023-39975

Availability improvements:

- Fixed an issue where processing `INSERT` queries on InnoDB partitioned tables can cause a gradual decline of free memory in the instance.
- Fixed an issue that can cause a database instance restart when running SHOW STATUS and PURGE BINARY LOGS statements concurrently. `PURGE BINARY LOGS` is a managed statement that is run to honor the user-configured binlog retention period.
- Fixed an issue that can cause the server to unexpectedly close after running Data Manipulation Language (DML) statements on a table whose nonvirtual columns were reordered with a `MODIFY COLUMN` or `CHANGE COLUMN` statement.

- Fixed an issue that, during the restart of a database instance, can cause an additional restart.

General improvements:

- Fixed an issue where the user is unable to interrupt any query or set session timeouts for `performance_schema` queries.
- Fixed an issue where binary log (binlog) replication setup using custom SSL certificates (mysql.rds_import_binlog_ssl_material) could fail when the replication instance is undergoing a host replacement.
- Fixed an issue related to audit log file management that can cause log files to be inaccessible for download or rotation, and in some cases increase CPU usage.

Upgrades and migrations:

- Fixed an issue that can cause upgrade failures from Aurora MySQL version 2 to Aurora MySQL version 3 when there is a user-defined `FTS_DOC_ID` column present in the table schema.
- Fixed an issue that can cause upgrade failures from Aurora MySQL version 2 to Aurora MySQL version 3 due to a synchronization issue while processing InnoDB tablespaces.
- Fixed an issue that can cause major version upgrades to Aurora MySQL version 3 to fail due to the presence of orphaned entries for already deleted tablespaces in InnoDB system tables in Aurora MySQL version 2.

Integration of MySQL Community Edition bug fixes

This release includes all community bug fixes up to and including 8.0.32. For more information, see MySQL bugs fixed by Aurora MySQL 3.x database engine updates.

Aurora MySQL database engine updates 2023-11-21 (version 3.05.1, compatible with MySQL 8.0.32)

Version: 3.05.1

Aurora MySQL 3.05.1 is generally available. Aurora MySQL 3.05 versions are compatible with MySQL 8.0.32. For more information, see MySQL 8.0 Release Notes.

Currently supported Aurora MySQL releases are 2.07.*, 2.11.*, 2.12.*, 3.01.*, 3.02.*, 3.03.*, 3.04.*, and 3.05.*.

You can upgrade an existing Aurora MySQL 3.* database cluster to Aurora MySQL 3.05.1. You can also restore a snapshot from any currently supported Aurora MySQL release into Aurora MySQL 3.05.1.

If you upgrade an Aurora MySQL global database to version 3.05.*, you must upgrade your primary and secondary DB clusters to the exact same version, including the patch level. For more information on upgrading the minor version of an Aurora global database, see Minor version upgrades.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Improvements

Fixed security issues and CVEs listed below:

This release includes all community CVEs fixes up to and including MySQL 8.0.32.

- CVE-2023-38545

Integration of MySQL Community Edition bug fixes

This release includes all community bug fixes up to and including 8.0.32, in addition to the below. For more information, see MySQL bugs fixed by Aurora MySQL 3.x database engine updates.

- Fixed an issue in InnoDB when, if a MySQL table in a system schema had an `INSTANT ADD` column added between Aurora MySQL versions 3.01 through Aurora MySQL versions 3.04, and after Aurora MySQL was upgraded to version 3.05.0, DMLs on these tables would result in the server unexpectedly closing. (Community Bug Fix #35625510)

Aurora MySQL database engine updates 2023-10-30 (version 3.05.0.1, compatible with MySQL 8.0.32) Beta

Version: 3.05.0.1

Aurora MySQL 3.05.0.1 is generally available in the following regions: US East (N. Virginia), US East (Ohio), US West (N. California), US West (Oregon), AWS GovCloud (US-East), and AWS GovCloud (US-West). This is an early, security fix–only release. These fixes will be deployed more broadly across all Regions with the next patch release, 3.05.1. Aurora MySQL 3.05 versions are compatible with MySQL 8.0.32. For more information on the community changes that have occurred, see MySQL 8.0 Release Notes.

Currently supported Aurora MySQL releases are 2.07.*, 2.11.*, 2.12.*, 3.01.*, 3.02.*, 3.03.*, 3.04.*, and 3.05.*.

You can upgrade an existing Aurora MySQL 3.* database cluster to Aurora MySQL 3.05.0.1. You can also restore a snapshot from any currently supported Aurora MySQL release into Aurora MySQL 3.05.0.1.

If you upgrade an Aurora MySQL global database to version 3.05.*, you must upgrade your primary and secondary DB clusters to the exact same version, including the patch level. For more information on upgrading the minor version of an Aurora global database, see Minor version upgrades.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Improvements

Fixed security issues and CVEs listed below:

This release includes all community CVEs fixes up to and including MySQL 8.0.32.

- CVE-2023-38545

Aurora MySQL database engine updates 2023-10-25 (version 3.05.0, compatible with MySQL 8.0.32)

Version: 3.05.0

Aurora MySQL 3.05.0 is generally available. Aurora MySQL 3.05 versions are compatible with MySQL 8.0.32. For more information on the community changes that have occurred, see MySQL 8.0 Release Notes.

For details of the new features in Aurora MySQL version 3, see Aurora MySQL version 3 compatible with MySQL 8.0. For differences between Aurora MySQL version 3 and Aurora MySQL version 2, see Comparison of Aurora MySQL version 2 and Aurora MySQL version 3. For a comparison of Aurora MySQL version 3 and MySQL 8.0 Community Edition, see Comparison of Aurora MySQL version 3 and MySQL 8.0 Community Edition.

Currently supported Aurora MySQL releases are 2.07.9, 2.07.10, 2.11.*, 2.12.*, 3.03.*, 3.04.*, and 3.05.*.

You can perform an in-place upgrade, restore a snapshot, or initiate a managed blue/green upgrade using Amazon RDS Blue/Green Deployments from any currently supported Aurora MySQL version 2 cluster into an Aurora MySQL version 3.05.0 cluster.

For information on planning an upgrade to Aurora MySQL version 3, see Upgrade planning for Aurora MySQL version 3 in the *Amazon Aurora User Guide*. For general information about Aurora MySQL upgrades, see Upgrading Amazon Aurora MySQL DB clusters in the *Amazon Aurora User Guide*.

For troubleshooting information, see Troubleshooting upgrade issues with Aurora MySQL version 3.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Improvements

New features:

- Added support for saving data from an Aurora MySQL database cluster into text files in an Amazon S3 bucket encrypted with a KMS key (SSE-KMS). For more information, see Saving data from an Amazon Aurora MySQL DB cluster into text files in an Amazon S3 bucket.
- Introduced a new global status variable `aurora_tmz_version` to denote the current version of the time zone (TZ) information used by the engine. The values follow the IANA time zone database version and are formatted as "YYYYsuffix", for example, 2022a and 2023c. For more information, see Aurora MySQL global status variables.

Fixed security issues and CVEs listed below:

Fixes and other enhancements to fine-tune handling in a managed environment. Additional CVE fixes are below:

- CVE-2022-37434

Availability improvements:

- Fixed an issue where Aurora MySQL database instances using parallel query may experience a database restart when running a high number of concurrent parallel queries.
- Fixed an issue with lock contention caused by an audit logging thread eventually leading to high CPU utilization and client application timeouts.
- Fixed an issue which can cause the executed GTID set to be recovered incorrectly on a binary log (binlog) replica cluster with enhanced binlog enabled when any binlog source has `gtid_mode` set to `ON` or `ON_PERMISSIVE`. This issue may cause the replica cluster's writer instance to restart an additional time during recovery, or lead to incorrect results when querying the executed GTID set.
- Fixed a memory management issue that can cause an Aurora MySQL database instance restart or a failover due to a decrease in freeable memory when enhanced binary log is enabled.
- Fixed an issue which can cause a database instance restart when attempting to read a database page that belongs to a dropped table.
- Fixed an issue which can cause the reader instance to restart when the writer instance grows the database volume to a multiple of 160GB.
- Fixed an issue where an Aurora MySQL database instance with the enhanced binary log feature enabled might get stuck during the database instance startup as the binary log recovery process is being executed.
- Fixed an issue where an Aurora MySQL database instance may experience multiple restarts during instance startup while large rollback segments are initialized.
- Fixed an issue during zero downtime patching which causes an instance restart that leads to database connections being unexpectedly closed.
- Fixed an issue which can cause a database instance restart due to a deadlatch when running SHOW STATUS and PURGE BINARY LOGS statements concurrently. The purge binary logs is a managed statement that is executed to honor the user configured binlog retention period.
- Fixed an issue which can cause database cluster unavailability if the writer instance restarts while the database is creating or dropping triggers on internal system tables.

- Fixed an issue which can cause a database instance restart due to a long semaphore wait when using the enhanced binlog feature on a cluster with an Aurora replica.
- Fixed an issue which can cause a database instance to restart while executing a query which references an aggregate function.
- Fixed an issue which, in rare conditions, can cause the database instance to restart when Aurora Serverless v2 incorrectly attempts to update the table cache while scaling.
- Fixed an issue where unsupported index scan access methods were considered for common table expressions (CTE) while materializing intermediate temporary tables, which can lead to undesired behavior including database restarts or incorrect query results. We fixed this issue by avoiding the use of such unsupported index scan access methods on tables using the TempTable storage engine.

General improvements:

- Fixed an issue which can cause database unavailability when enhanced binlog is enabled on an Aurora Serverless v2 database cluster running on Aurora MySQL 3.04.0.
- Removed unused storage metadata before writing to Aurora storage when the enhanced binlog feature is enabled. This avoids certain scenarios when a database restart or failover may occur because of increased write latency due to increased bytes transmitted over the network.
- With the addition of the `malloc_stats` and `malloc_stats_totals` tables in the `performance_schema`, three advanced system variables were added to control the behavior of Jemalloc, an internal memory allocator:
  - `aurora_jemalloc_background_thread`.
  - `aurora_jemalloc_dirty_decay_ms`.
  - `aurora_jemalloc_tcache_enabled`.
- Fixed an issue where Aurora specific performance schema tables were not created upon an upgrade or migration.
- Added a new system variable, `aurora_use_vector_instructions`. When this parameter is enabled, Aurora MySQL uses optimized vector processing instructions to improve performance on I/O heavy workloads. This setting is turned `ON` by default in Aurora MySQL 3.05 and higher. For more details, see Aurora MySQL configuration parameters.
- Fixed an issue which can cause the `NumBinaryLogFiles` metrics on CloudWatch to display incorrect results when enhanced binlog is enabled.

- The request timeout for Aurora MySQL Machine Learning operations to Amazon Sagemaker has been increased from 3 to 30 seconds. This helps resolve an issue where customers may see an increased number of retries or failures for requests to Amazon Sagemaker from Aurora MySQL Machine Learning when using larger batch sizes.
- Added support for `malloc_stats` and `malloc_stats_totals` tables in the performance_schema database.
- Updated the `FROM` keyword in the `LOAD DATA FROM S3` command to be optional. For more information, see Loading data into an Amazon Aurora MySQL DB cluster from text files in an Amazon S3 bucket.
- Added support for the parameter `innodb_aurora_instant_alter_column_allowed`, which controls whether the `INSTANT` algorithm can be used for `ALTER COLUMN` operations. For more information see Cluster-level parameters.
- Fixed an issue which can prevent new client connections from being established to the database when write forwarding is enabled.
- Fixed an issue which can cause the modification of the `table_open_cache` database parameter to not take effect until the database instance is restarted.
- Fixed an issue which can cause duplicate key errors for `AUTO_INCREMENT` columns using descending indices after a snapshot restore, backtrack, or database clone operation.
- Fixed an issue involving index scans where an inaccurate result might be returned when executing a `SELECT` query with the `GROUP BY` clause and the `aurora_parallel_query`parameter turned `ON`.
- Fixed an issue which may cause the depletion of available memory when executing queries against the `INFORMATION_SCHEMA INNODB_TABLESPACES` table.
- Fixed an issue where the reader instance is unable to open a table, with ERROR 1146. This issue occurs when executing certain types of online Data Definition Language (DDL) while the `INPLACE` algorithm is being used on the writer instance.
- Fixed an issue to avoid an instance restart during Aurora Serverless v2 scaling when the internal monitoring process erroneously submits duplicate scaling requests.
- Fixed an issue which can cause a database restart when connected binary log (binlog) consumers are using duplicate binlog replication server IDs.
- Introduced an in-memory relay log cache for Aurora MySQL managed binary log replicas. This improvement can help achieve up to a 40% increase in binary log replication throughput. This enhancement is enabled automatically when using single-threaded binary log replication or when using multi-threaded replication with GTID auto-positioning enabled.

Upgrades and migrations:

- Upgrading from MySQL 5.7 to MySQL 8.0 with a very large number of tables in a single database caused the server to consume excessive memory. It was found that, during the process of checking whether tables could be upgraded, we fetched all the data dictionary `Table` objects upfront, processing each and fetching its name, then performed Checking Version Compatibility on the list. Fetching all objects beforehand was not necessary in this case, and contributed greatly to memory consumption. To correct this problem, we now fetch one `Table` object at a time in such cases, performing any required checks, fetching its name, and releasing the object, before proceeding with the next one. (Bug #34526001)
- Improved the performance of major version upgrades from Aurora MySQL version 2 to version 3 by executing tablespace checks in parallel using all available vCPUs on the database instance.

Integration of MySQL Community Edition bug fixes

This release includes all community bug fixes up to and including 8.0.32, in addition to the below. For more information, see MySQL bugs fixed by Aurora MySQL 3.x database engine updates.

- Fixed an issue which can cause higher CPU utilization due to background TLS certificate rotation. (Community Bug Fix #34284186)

Aurora MySQL database engine updates 2024-06-26 (version 3.04.3, compatible with MySQL 8.0.28)

Version: 3.04.3

Aurora MySQL 3.04.3 is generally available. Aurora MySQL 3.04 versions are compatible with MySQL 8.0.28. For more information on community changes that have occurred, see MySQL 8.0 Release Notes.

For details of the new features in Aurora MySQL version 3, see Aurora MySQL version 3 compatible with MySQL 8.0. For differences between Aurora MySQL version 3 and Aurora MySQL version 2, see Comparison of Aurora MySQL version 2 and Aurora MySQL version 3. For a comparison of Aurora MySQL version 3 and MySQL 8.0 Community Edition, see Comparison of Aurora MySQL version 3 and MySQL 8.0 Community Edition.

Note

This version is designated as a long-term support (LTS) release. For more information, see Aurora MySQL long-term support (LTS) releases in the *Amazon Aurora User Guide*.
We recommend that you don't set the `AutoMinorVersionUpgrade` parameter to `true` (or enable **Auto minor version upgrade** in the AWS Management Console) for LTS versions. Doing so could lead to your DB cluster being upgraded to a non-LTS version such as 3.05.2.

Currently supported Aurora MySQL releases are 2.07.9, 2.7.10, 2.11.*, 2.12.*, 3.03.*, 3.04.*, 3.05.*, 3.06.*, and 3.07.*.

You can perform an in-place upgrade, restore a snapshot, or initiate a managed blue/green upgrade using Amazon RDS Blue/Green Deployments from any currently available Aurora MySQL version 2 cluster into an Aurora MySQL version 3.04.3 cluster.

For information on planning an upgrade to Aurora MySQL version 3, see Planning a major version upgrade for an Aurora MySQL DB cluster. For general information about Aurora MySQL upgrades, see Upgrading Amazon Aurora MySQL DB clusters in the *Amazon Aurora User Guide*.

For troubleshooting information, see Troubleshooting for Aurora MySQL in-place upgrade in the *Amazon Aurora User Guide*.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Improvements

Fixed security issues and CVEs:

This release includes all community CVE fixes up to and including MySQL 8.0.28. The following CVE fixes are included:

- CVE-2024-0853

Availability improvements:

- Fixed an issue that causes an Aurora MySQL DB instance to restart when running a parallel query.

- Fixed an issue that can cause a reader DB instance to restart when reading a table that is being altered or dropped on the writer DB instance.
- Fixed an issue that caused a memory access violation leading to releasing a mutex object no longer owned by the thread.
- Fixed an issue that can cause an Aurora MySQL writer DB instance to restart when a write forwarding session is closed while running a forwarded query.
- Fixed an issue that causes a DB instance to restart when handling large GTID sets on a binary log–enabled instance.
- Fixed an issue when processing `INSERT` queries on InnoDB partitioned tables that can cause a gradual decline of free memory in the DB instance.
- Fixed an issue that, in rare conditions, can cause a reader instance to restart when performing `SELECT` queries on tables with a foreign key constraint.
- Fixed an issue that can cause a database to restart when InnoDB data dictionary recovery takes a long time during database recovery.
- Fixed an issue that can cause a database to restart when a cascading `UPDATE` or `DELETE` foreign key constraint is defined on a table where a virtual column is involved either as a column in the foreign key constraint, or as a member of the referenced table.
- Fixed an issue in Aurora Serverless v2 that can lead to a database restart while scaling up.

General improvements:

- Fixed an issue that provided an incorrect value for the `threads_running` status variable when using Aurora Global Database.
- Fixed an issue that causes a DB instance to restart because of inaccurate lock holder information in `rw_lock` when using parallel reads.
- Fixed a memory management issue that led to a decrease in freeable memory over time when running `SELECT ... INTO OUTFILE ...` queries.
- Fixed an issue that can cause a DB instance to restart when the local storage on the DB instance reaches full capacity.
- Fixed an issue where the Performance Schema wasn't enabled when Performance Insights automated management was turned on for db.t4g.medium and db.t4g.large DB instances.
- Fixed an issue during zero downtime patching (ZDP) that prevents a DB instance from closing client connections upon reaching the customer-configured of either `wait_timeout` or `interactive_timeout`.

- Fixed an issue where a `SELECT` query on an Aurora reader instance might fail with the error table doesn't exist when the table has at least one full-text search (FTS) index and a `TRUNCATE` statement is being run on the Aurora writer DB instance.

Upgrades and migrations:

- Fixed an issue that causes upgrades or migrations to fail when the target Aurora MySQL DB engine version is 3.04.0 or higher. This occurs when the `lower_case_table_names` DB cluster parameter is set to 1, and MySQL database collation is incompatible with lowercase table names.

Integration of MySQL Community Edition bug fixes

This release includes all community bug fixes up to and including 8.0.28. For more information, see MySQL bugs fixed by Aurora MySQL 3.x database engine updates.

Aurora MySQL database engine updates 2024-03-15 (version 3.04.2, compatible with MySQL 8.0.28)

Version: 3.04.2

Aurora MySQL 3.04.2 is generally available. Aurora MySQL 3.04 versions are compatible with MySQL 8.0.28. For more information on community changes that have occurred, see MySQL 8.0 Release Notes.

For details of the new features in Aurora MySQL version 3, see Aurora MySQL version 3 compatible with MySQL 8.0. For differences between Aurora MySQL version 3 and Aurora MySQL version 2, see Comparison of Aurora MySQL version 2 and Aurora MySQL version 3. For a comparison of Aurora MySQL version 3 and MySQL 8.0 Community Edition, see Comparison of Aurora MySQL version 3 and MySQL 8.0 Community Edition.

> **Note**
>
> This version is designated as a long-term support (LTS) release. For more information, see Aurora MySQL long-term support (LTS) releases in the *Amazon Aurora User Guide*.
> We recommend that you don't set the `AutoMinorVersionUpgrade` parameter to `true` (or enable **Auto minor version upgrade** in the AWS Management Console) for LTS versions. Doing so could lead to your DB cluster being upgraded to a non-LTS version such as 3.05.2.

Currently supported Aurora MySQL releases are 2.07.9, 2.7.10, 2.11.*, 2.12.*, 3.03.*, 3.04.*, 3.05.*, and 3.06.*.

You can perform an in-place upgrade, restore a snapshot, or initiate a managed blue/green upgrade using Amazon RDS Blue/Green Deployments from any currently available Aurora MySQL version 2 cluster into an Aurora MySQL version 3.04.2 cluster.

For information on planning an upgrade to Aurora MySQL version 3, see Upgrade planning for Aurora MySQL version 3 in the *Amazon Aurora User Guide*. For general information about Aurora MySQL upgrades, see Upgrading Amazon Aurora MySQL DB clusters in the *Amazon Aurora User Guide*.

For troubleshooting information, see Troubleshooting upgrade issues with Aurora MySQL version 3.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Improvements

Fixed security issues and CVEs:

The following CVE fixes are included in this release:

- CVE-2020-11104
- CVE-2020-11105
- CVE-2023-38545
- CVE-2023-38546
- CVE-2023-39975

Availability improvements:

- Fixed an issue where a read replica DB instance can't be launched successfully when there's high workload in the writer DB instance.
- Fixed an issue where an Aurora MySQL writer DB instance can fail over due to a defect in the component that communicates with Aurora storage. The defect occurs as a result of a breakdown in the communication between the DB instance and underlying storage following a software update.

- Fixed an issue that can cause a database instance restart when running SHOW STATUS and PURGE BINARY LOGS statements concurrently. `PURGE BINARY LOGS` is a managed statement that is run to honor the user-configured binlog retention period.
- Fixed an issue that, during the restart of a database instance, can cause an additional restart.
- Fixed an issue with lock contention caused by an audit logging thread that can lead to high CPU utilization and client application timeouts.
- Fixed an issue where an Aurora MySQL database instance can experience multiple restarts during instance startup while large rollback segments are initialized.
- Fixed an issue that can cause a DB instance to restart while running a query that references an aggregate function.

General improvements:

- Fixed an issue that can cause a parallel query to fail due to transient network issues while reading data from the Aurora DB cluster volume
- Fixed an issue where the user is unable to interrupt any query or set session timeouts for `performance_schema` queries.
- Fixed an issue where binary log (binlog) replication configured to use custom SSL certificates (mysql.rds_import_binlog_ssl_material) could fail when the replication instance is undergoing a host replacement.
- Fixed an issue related to audit log file management that can cause log files to be inaccessible for download or rotation, and in some cases increase CPU usage.
- Optimized `AUTO_INCREMENT` key recovery to reduce the completion time for restoring snapshots, performing point-in-time recovery, and cloning DB clusters with large numbers of tables in the database.
- Fixed an issue where SQL statements referring to some `performance_schema` tables can return an error due to these tables being missing after migrating from Community MySQL to Aurora MySQL versions 3.04.0 and 3.04.1.
- Fixed an issue where small read replica instances can experience increased replication lag after upgrading from Aurora MySQL versions lower than 2.11.*.
- Fixed an issue that can cause duplicate key errors for `AUTO_INCREMENT` columns using descending indices after a snapshot restore, backtrack, or database cloning operation.
- Fixed an issue that can cause modifications of the `table_open_cache` database parameter not to take effect until the DB instance is restarted.

- Fixed an issue where the reader DB instance is unable to open a table, with ERROR 1146. This issue occurs when running certain types of online Data Definition Language (DDL) statements while the `INPLACE` algorithm is being used on the writer DB instance.
- Fixed an issue to avoid an instance restart during Aurora Serverless v2 scaling when the internal monitoring process erroneously submits duplicate scaling requests.
- Fixed an issue that can cause a database restart when connected binary log (binlog) consumers are using duplicate binlog replication server IDs.

Upgrades and migrations:

- Fixed an issue that can cause major version upgrades to Aurora MySQL version 3 to fail due to the presence of orphaned entries for already deleted tablespaces in InnoDB system tables in Aurora MySQL version 2.

Integration of MySQL Community Edition bug fixes

This release includes all community bug fixes up to and including 8.0.28, in addition to the following. For more information, see MySQL bugs fixed by Aurora MySQL 3.x database engine updates.

- Fixed an issue where cache line value can be calculated incorrectly, causing a failure during database restart on Graviton-based instances. (Community Bug Fix #35479763)
- Repeated running of a stored routine, having as a subquery a SELECT statement containing multiple `AND`, `OR`, or `XOR` conditions, led to excessive consumption and possibly eventual exhaustion of virtual memory. (Community Bug Fix #33852530)

Aurora MySQL database engine updates 2023-11-13 (version 3.04.1, compatible with MySQL 8.0.28)

Version: 3.04.1

Aurora MySQL 3.04.1 is generally available. Aurora MySQL 3.04 versions are compatible with MySQL 8.0.28. For more information on community changes that have occurred, see MySQL 8.0 Release Notes.

> **Note**
>
> This version is designated as a long-term support (LTS) release. For more information, see Aurora MySQL long-term support (LTS) releases in the *Amazon Aurora User Guide*.
> We recommend that you don't set the `AutoMinorVersionUpgrade` parameter to `true` (or enable **Auto minor version upgrade** in the AWS Management Console) for LTS versions. Doing so could lead to your DB cluster being upgraded to a non-LTS version such as 3.05.2.

For details of the new features in Aurora MySQL version 3, see Aurora MySQL version 3 compatible with MySQL 8.0. For differences between Aurora MySQL version 3 and Aurora MySQL version 2, see Comparison of Aurora MySQL version 2 and Aurora MySQL version 3. For a comparison of Aurora MySQL version 3 and MySQL 8.0 Community Edition, see Comparison of Aurora MySQL version 3 and MySQL 8.0 Community Edition.

Currently supported Aurora MySQL releases are 2.07.9, 2.7.10, 2.11.*, 2.12.*, 3.01.*, 3.02.*, 3.03.*, 3.04.*, and 3.05.*.

You can perform an in-place upgrade, restore a snapshot, or initiate a managed blue/green upgrade using Amazon RDS Blue/Green Deployments from any currently available Aurora MySQL version 2 cluster into an Aurora MySQL version 3.04.1 cluster.

For information on planning an upgrade to Aurora MySQL version 3, see Upgrade planning for Aurora MySQL version 3 in the *Amazon Aurora User Guide*. For general information about Aurora MySQL upgrades, see Upgrading Amazon Aurora MySQL DB clusters in the *Amazon Aurora User Guide*.

For troubleshooting information, see Troubleshooting upgrade issues with Aurora MySQL version 3.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Improvements

Availability improvements:

- Fixed an issue where Aurora MySQL database instances using parallel query may experience a database restart when running a high number of concurrent parallel queries.

- Fixed an issue which can cause the executed GTID set to be recovered incorrectly on a binary log (binlog) replica cluster with enhanced binlog enabled when any binlog source has set `gtid_mode` to `ON` or `ON_PERMISSIVE`. This issue may cause the replica cluster's writer instance to restart an additional time during recovery, or lead to incorrect results when querying the executed GTID set.
- Fixed a memory management issue that can cause an Aurora MySQL database instance restart or a failover due to a decrease in freeable memory when enhanced binary log is enabled.
- Fixed an issue which can cause the reader instance to restart when the writer instance grows the database volume to a multiple of 160GB.
- Fixed an issue where an Aurora MySQL database instance with the enhanced binary log feature enabled might get stuck during the database instance startup as the binary log recovery process is being executed.
- Fixed an issue which can cause a database instance restart due to a deadlatch when running `SHOW STATUS` and `PURGE BINARY LOGS` statements concurrently. The purge binary logs is a managed statement that is executed to honor the user configured binlog retention period.
- Fixed an issue which can cause database cluster unavailability if the writer instance restarts while the database is creating or dropping triggers on internal system tables.
- Fixed an issue which can cause a database instance restart due to a long semaphore wait when using the enhanced binlog feature on a cluster with an Aurora replica.

General improvements:

- Fixed an issue which can cause database unavailability when enhanced binlog is enabled on an Aurora Serverless v2 database cluster running on Aurora MySQL 3.04.0.
- Removed unused storage metadata before writing to Aurora Storage when the enhanced binlog feature is enabled. This avoids certain scenarios when a database restart or failover may occur because of increased write latency due to increased bytes transmitted over the network.
- Fixed an issue where Aurora specific performance schema tables were not created upon an upgrade or migration.
- Fixed an issue which can cause the `NumBinaryLogFiles` metrics on CloudWatch to display incorrect results when enhanced binlog is enabled.

Upgrades and migrations:

- Upgrading from MySQL 5.7 to MySQL 8.0 with a very large number of tables in a single database caused the server to consume excessive memory. It was found that, during the process of checking whether tables could be upgraded, we fetched all the data dictionary `Table` objects upfront, processing each and fetching its name, then performed `CHECK TABLE ... FOR UPGRADE` on the list. Fetching all objects beforehand was not necessary in this case, and contributed greatly to memory consumption. To correct this problem, we now fetch one `Table` object at a time in such cases, performing any required checks, fetching its name, and releasing the object, before proceeding with the next one. (Bug #34526001)

Integration of MySQL Community Edition bug fixes

This release includes all community bug fixes up to and including 8.0.28, in addition to the below. For more information, see MySQL bugs fixed by Aurora MySQL 3.x database engine updates.

- Fixed an issue which can cause higher CPU utilization due to background TLS certificate rotation (Community Bug Fix #34284186)

Aurora MySQL database engine updates 2023-07-31 (version 3.04.0, compatible with MySQL 8.0.28)

Version: 3.04.0

Aurora MySQL 3.04.0 is generally available. Aurora MySQL 3.04 versions are compatible with MySQL 8.0.28, Aurora MySQL 3.03 versions are compatible with MySQL 8.0.26, and Aurora MySQL 3.02 versions are compatible with MySQL 8.0.23. For more information on community changes that have occurred from 8.0.23 to 8.0.28, see MySQL 8.0 Release Notes.

> **Note**
>
> This version is designated as a long-term support (LTS) release. For more information, see Aurora MySQL long-term support (LTS) releases in the *Amazon Aurora User Guide*.
> We recommend that you don't set the `AutoMinorVersionUpgrade` parameter to `true` (or enable **Auto minor version upgrade** in the AWS Management Console) for LTS versions. Doing so could lead to your DB cluster being upgraded to a non-LTS version such as 3.05.2.

For details of the new features in Aurora MySQL version 3, see Aurora MySQL version 3 compatible with MySQL 8.0. For differences between Aurora MySQL version 3 and Aurora MySQL version 2, see Comparison of Aurora MySQL version 2 and Aurora MySQL version 3. For a comparison of Aurora MySQL version 3 and MySQL 8.0 Community Edition, see Comparison of Aurora MySQL version 3 and MySQL 8.0 Community Edition.

Currently supported Aurora MySQL releases are 2.07.9, 2.11.1, 2.11.2, 3.01.*, 3.02.*, 3.03.*, and 3.04.0.

You can perform an in-place upgrade, restore a snapshot, or initiate a managed blue/green upgrade using Amazon RDS Blue/Green Deployments from any currently supported Aurora MySQL version 2 cluster into an Aurora MySQL version 3.04.0 cluster.

For information on planning an upgrade to Aurora MySQL version 3, see Upgrade planning for Aurora MySQL version 3 in the *Amazon Aurora User Guide*. For general information about Aurora MySQL upgrades, see Upgrading Amazon Aurora MySQL DB clusters in the *Amazon Aurora User Guide*.

For troubleshooting information, see Troubleshooting upgrade issues with Aurora MySQL version 3.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

> **Note**
>
> Aurora MySQL enhanced binary log (binlog) is currently not supported for the Aurora Serverless v2 database instance on Aurora MySQL version 3.04.0. Enabling this feature may lead to database unavailability. If you require the use of enhanced binary log on Aurora MySQL version 3.04.0, we recommend using a non-serverless database instance class or setting the minimum and maximum ACU of the Serverless v2 database instance to the same value.
> More information on enhanced binary logging in Aurora MySQL is available in the Aurora User Guide.

Improvements

New features:

- Improved the performance of queries using InnoDB full-text indices to search phrases in natural language mode. For more information on full text searches in MySQL, refer to Full-Text Search Functions.
- Amazon Aurora MySQL supports local (in-cluster) write forwarding. You can now forward write operations from a reader DB instance to a writer DB instance within an Aurora MySQL DB cluster. For more information, see Using local write forwarding in an Aurora MySQL DB cluster.
- Added the capability to change the value of the `aurora_replica_read_consistency` parameter for the Using write forwarding in an Amazon Aurora global database feature in sessions which have `autocommit` disabled. For more information, see Configuration parameters for write forwarding.
- Starting with Aurora MySQL 3.04, for the global database write forwarding feature, you can now set the value of the `aurora_replica_read_consistency` parameter via the database cluster and database instance parameter groups. Prior to Aurora MySQL version 3.04, this parameter's value could only be configured at the session level.

Fixed security issues and CVEs:

- Changed the SSL/TLS provider from OpenSSL to AWS-LC. This brings a number of changes including, but not limited to the following:
  - Database connections using SSL can now be restored by Zero Downtime Restart and Zero Downtime Patching when upgrading from Aurora MySQL version 3.04.0 to a higher version.
  - Support for TLSv1.3 which includes support for TLS_AES_128_GCM_SHA256, TLS_AES_256_GCM_SHA384 and TLS_CHACHA20_POLY1305_SHA256 SSL ciphers.
  - Removal of support for less secure DHE-RSA-* ciphers.

  For more information, see Using TLS with Aurora MySQL DB clusters
- Added the dynamic privilege `SHOW_ROUTINE` to the `rds_superuser_role` which enables access to definitions and properties of all stored routines, such as stored procedures and functions. For more details, see SHOW_ROUTINE.
- Fixed an issue which may cause the audit log to miss events during audit log file rotation.
- Enabled support for secure and performant Transport Layer Security (TLS) 1.3 protocol while maintaining compatibility with TLS 1.2 version.
- TLS versions TLSv1 and TLSv1.1 were deprecated in community MySQL 8.0.26 and correspondingly in Aurora MySQL 3.03. These protocols have now been removed in community MySQL 8.0.28 and correspondingly in Aurora MySQL 3.04. By default, any secure client that

cannot communicate over TLS 1.2 or higher will be rejected. For more information on connecting to your database instances using TLS, please see Security with Amazon Aurora MySQL.

The following CVE fixes are included in this release:

- CVE-2023-21963
- CVE-2023-21912
- CVE-2023-0215
- CVE-2022-43551
- CVE-2022-37434
- CVE-2022-21635
- CVE-2022-21556
- CVE-2022-21352
- CVE-2021-35630
- CVE-2021-35624

Availability improvements:

- Fixed an issue that can cause database restarts during long transaction recovery.
- Fixed an issue within database activity streams event encryption that can cause database restarts.
- Fixed a memory management issue due to out of memory errors when the InnoDB buffer pool is being initialized during startup or while scaling in Aurora Serverless v2. This issue might have caused database instance restarts or performance degradation including throughput reduction or increased latency.
- Fixed an issue that can cause an Aurora MySQL reader instance to restart while executing a query which utilizes an Aurora MySQL parallel query execution plan.
- Fixed an issue that, in certain situations, can cause Aurora reader instances to restart during a range estimation.
- Fixed an issue that can interrupt database recovery during startup if the restart occurred while executing heavy insert operations involving auto-increment columns.

- Fixed an issue with Aurora advanced auditing that causes excess logging of informational messages to the Aurora MySQL error log when the server variable `server_audit_events` is set to `ALL` or `QUERY`. This issue might cause a database instance restart.
- Fixed an issue that can cause a database restart during the rollback of an `INSERT` statement when parallel query is enabled.
- Fixed an issue that can cause the database instance to restart when running the `EXPLAIN ANALYZE` profiling tool on a query which returned the output `all select tables were optimized away` within the `EXTRA` information column. For more information, see the MySQL documentation on EXPLAIN Output Format.
- Fixed an issue that can cause an Aurora global database secondary Region reader instance using global write forwarding to restart when a forwarded implicit commit statement encounters an error.
- Fixed an issue that can cause the writer instance in an Aurora global database primary Region to restart when a `SELECT FOR UPDATE` query is executed using global write forwarding from an Aurora global database secondary Region.

General improvements:

- Added a new stored procedure, `mysql.rds_gtid_purged`, to allow customers to set the `GTID_PURGED` system variable. For more information, see mysql.rds_gtid_purged.
- Added two new stored procedures, `mysql.rds_start_replication_until` and `mysql.rds_start_replication_until_gtid`, which allow customers to configure a location to stop binary log replication. For more information on configuring a stop location for binary log replication in Aurora MySQL, see mysql.rds_start_replication_until.
- Fixed an issue that would prevent Aurora MySQL replication control stored procedures from modifying the `sql_log_bin` variable, when called from a session with autocommit mode disabled.
- Added logical replication support for the following Data Control Language (DCL) statements: `GRANT/REVOKE` and `CREATE/DROP/ALTER/RENAME USER`.
- Fixed an issue to prevent InnoDB statistics from getting stale, which can sometimes generate a sub-optimal query execution plan that may lead to an increase in the query execution time.
- Added two new system views, `information_schema.aurora_global_db_instance_status` and `information_schema.aurora_global_db_status`. These views can be used to display

the status and topology of primary and secondary resources in an Aurora MySQL global database cluster. The details of the two system views can be found here, Aurora MySQL–specific information_schema tables.

- Fixed an issue where a user is unable to access the database with a wildcard character in the database name after executing the `SET ROLE` statement with an escaped wildcard character.
- Fixed an issue where events that were reported while processing audit log rotations might not be written to the audit log.
- Fixed an issue where creating an internal temporary table, via a `TRIGGER` execution, can cause a writer database instance to restart.
- Added a new system variable, `innodb_aurora_max_partitions_for_range`. In some cases where persisted stats are not available, this parameter can be used to improve the execution time of row count estimations on partitioned tables. More information can be found in the documentation, Aurora MySQL configuration parameters.
- Fixed an issue which incorrectly allows customers to set `ROW_FORMAT` as `COMPRESSED` when creating partitioned tables. Tables will be implicitly converted to `COMPACT` format with a warning to inform that Aurora MySQL doesn't support compressed tables.
- Fixed an issue which can cause multi-threaded binary log replication to stop when the `replica_parallel_type` variable is set to `LOGICAL_CLOCK` and the `replica_preserve_commit_order` variable is turned `ON`. This issue can occur when a transaction larger than 500MB is executed on the source.
- Fixed an issue when the global database write forwarding feature is enabled that can cause changes to the `performance_schema` configuration on the reader instances in the secondary Regions to be unintentionally forwarded to the writer instance in the primary Region.
- Fixed an issue where the server status variable `innodb_buffer_pool_reads` may not be updated after a data page is read from the Aurora storage file system.
- Aurora MySQL parallel query is not supported when choosing the Aurora I/O-Optimized cluster configuration. For more information, see Amazon Aurora MySQL parallel query limitations.
- Fixed an issue, when parallel query is enabled, that causes the query plan optimizer to pick an inefficient execution plan for certain `SELECT` queries that benefit from a primary or secondary index.
- Upgraded the time zone definitions to the IANA 2023c version.
- Introduced file management performance optimizations on binlog replicas to help reduce contention when writing to relay log files.

- Fixed an issue where the `RPO_LAG_IN_MILLISECONDS` column in the `information_schema.aurora_global_db_status` table and `AuroraGlobalDBRPOLag` CloudWatch metric always displayed zero regardless of the user workload.
- Introduced a new parameter `aurora_tmptable_enable_per_table_limit`. When this parameter is enabled, the `tmp_table_size` variable defines the maximum size of the individual in-memory internal temporary table created by the TempTable storage engine. For additional details, see Storage engine for internal (implicit) temporary tables.
- Fixed an issue where an additional connection is created when the global database write forwarding feature is enabled. The issue occurs when read-only transactions on a reader instance incorrectly forward an implicit commit to the writer.
- Fixed an issue where the `PROCESSLIST_USER` and `PROCESSLIST_HOST` fields in the `performance_schema.threads` table were not populated on the writer in the primary Region for connections using the global database write forwarding feature. More information about this table and Performance Schema can be found in the MySQL Reference Manual, The threads Table, and the Amazon Aurora User Guide Overview of the Performance Schema.
- Fixed an issue where the `CommitLatency` Cloudwatch metric displayed incorrect values for reader instances in secondary Regions when the global database write forwarding feature is used. To monitor the forwarded DML statement latency on secondary database clusters, it is recommended to use the `ForwardingReplicaDMLLatency` and `ForwardingWriterDMLLatency` metrics. Commit latency can also be observed using the `CommitLatency` metric on the primary Region's writer instance. More information is available in the Aurora User Guide, Amazon CloudWatch metrics for write forwarding.
- Fixed an issue where the Aurora MySQL replication control stored procedures used to manage and configure binary log replication incorrectly report errors when multi-threaded binary log replication is configured by setting the `replica_parallel_workers` variable's value greater than 0.
- Fixed an issue that can cause high CPU consumption when multiple sessions are trying to access a page that doesn't exist in memory.

Upgrades and migrations:

- To perform a minor version upgrade for an Aurora global database from Aurora MySQL version 3.01, 3.02 or 3.03 to Aurora MySQL version 3.04 or higher, refer to Upgrading Aurora MySQL by modifying the engine version.

- Fixed an issue that can cause upgrade precheck failures due to schema inconsistency errors reported for the `mysql.general_log_backup`, `mysql.general_log`, `mysql.slow_log_backup` and `mysql.slow_log` tables when upgrading from Aurora MySQL 2 to Aurora MySQL 3. For more information about upgrade troubleshooting, see Troubleshooting upgrade issues with Aurora MySQL version 3.
- Fixed an issue which can cause major version upgrade failures when upgrading to Aurora MySQL 3 when a trigger definition contains a reserved keyword that is not within quotes.

Integration of MySQL Community Edition bug fixes

This release includes all community bug fixes up to and including 8.0.28, in addition to the below. For more information, see MySQL bugs fixed by Aurora MySQL 3.x database engine updates.

- Fixed an issue where a buffer block containing intrinsic temporary table page was relocated during page traversal, causing an assertion failure (Bug# 33715694)
- InnoDB: Prevent online DDL operations from accessing out-of-bounds memory (Bug# 34750489, Bug# 108925)
- Fixed an issue which can sometimes produce incorrect query results while processing complex SQL statements consisting of multiple nested Common Table Expressions (CTEs) (Bug# 34572040, Bug# 34634469, Bug# 33856374)

Aurora MySQL database engine updates 2023-12-08 (version 3.03.3, compatible with MySQL 8.0.26)

Version: 3.03.3

Aurora MySQL 3.03.3 is generally available. Aurora MySQL 3.03 versions are compatible with MySQL 8.0.26. For more information on community changes that have occurred from 8.0.23 to 8.0.28, see MySQL 8.0 Release Notes.

For details of the new features in Aurora MySQL version 3, see Aurora MySQL version 3 compatible with MySQL 8.0. For differences between Aurora MySQL version 3 and Aurora MySQL version 2, see Comparison of Aurora MySQL version 2 and Aurora MySQL version 3. For a comparison of Aurora MySQL version 3 and MySQL 8.0 Community Edition, see Comparison of Aurora MySQL version 3 and MySQL 8.0 Community Edition.

Currently available Aurora MySQL releases are 2.07.9, 2.07.10, 2.11.*, 2.12.*, 3.01.*, 3.02.*, 3.03.*, 3.04.*, and 3.05.*.

You can perform an in-place upgrade, restore a snapshot, or initiate a managed blue/green upgrade using Amazon RDS Blue/Green Deployments from any currently available Aurora MySQL version 2 cluster into an Aurora MySQL version 3.03.3 cluster.

For information on planning an upgrade to Aurora MySQL version 3, see Upgrade planning for Aurora MySQL version 3 in the *Amazon Aurora User Guide*. For general information about Aurora MySQL upgrades, see Upgrading Amazon Aurora MySQL DB clusters in the *Amazon Aurora User Guide*.

For troubleshooting information, see Troubleshooting upgrade issues with Aurora MySQL version 3.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Improvements

Fixed security issues and CVEs listed below:

Fixes and other enhancements to fine-tune handling in a managed environment. Additional CVE fixes are below:

- CVE-2023-38545

Availability improvements:

- Fixed an issue where Aurora MySQL database instances using parallel query may experience a database restart when running a high number of concurrent parallel queries.
- Fixed an issue which can cause the executed GTID set to be recovered incorrectly on a binary log (binlog) replica cluster with enhanced binlog enabled when any binlog source has `gtid_mode` set to `ON` or `ON_PERMISSIVE`. This issue may cause the replica cluster's writer instance to restart an additional time during recovery, or lead to incorrect results when querying the executed GTID set.
- Fixed a memory management issue that can cause an Aurora MySQL database instance restart or a failover due to a decrease in the freeable memory when enhanced binary log is enabled.

- Fixed an issue which can cause the reader instance to restart when the writer instance grows the database volume to a multiple of 160GB.
- Fixed an issue where an Aurora MySQL database instance with the enhanced binary log feature enabled might get stuck during the database instance startup as the binary log recovery process is being executed.
- Fixed an issue during zero downtime patching which causes an instance restart that leads to the database connections being unexpectedly closed.
- Fixed an issue which can cause a database instance restart due to a deadlatch when running `SHOW STATUS` and `PURGE BINARY LOGS` statements concurrently. The purge binary logs is a managed statement that is executed to honor the user configured binlog retention period.
- Fixed an issue which can cause a database instance restart due to a long semaphore wait when using the enhanced binlog feature on a cluster with an Aurora replica.

General improvements:

- Removed unused storage metadata before writing to Aurora storage when the enhanced binlog feature is enabled. This avoids certain scenarios when a database restart or failover may occur because of increased write latency due to increased bytes transmitted over the network.
- Fixed an issue which can cause the `NumBinaryLogFiles` metrics on CloudWatch to display incorrect results when enhanced binlog is enabled.
- Fixed an issue which can cause the modification of the `table_open_cache` database parameter to not take effect until the database instance is restarted.
- Fixed an issue which can cause a database restart when connected binary log (binlog) consumers are using duplicate binlog replication server IDs.

Integration of MySQL Community Edition bug fixes

This release includes all community bug fixes up to and including 8.0.26, in addition to the below. For more information, see MySQL bugs fixed by Aurora MySQL 3.x database engine updates.

- Fixed an issue which can cause higher CPU utilization due to background TLS certificate rotation (Community Bug Fix #34284186)

Aurora MySQL database engine updates 2023-08-29 (version 3.03.2, compatible with MySQL 8.0.26)

Version: 3.03.2

Aurora MySQL 3.03.2 is generally available. Aurora MySQL 3.04 versions are compatible with MySQL 8.0.28, Aurora MySQL 3.03 versions are compatible with MySQL 8.0.26, and Aurora MySQL 3.02 versions are compatible with MySQL 8.0.23. For more information on community changes that have occurred from 8.0.23 to 8.0.28, see MySQL 8.0 Release Notes.

For details of the new features in Aurora MySQL version 3, see Aurora MySQL version 3 compatible with MySQL 8.0. For differences between Aurora MySQL version 3 and Aurora MySQL version 2, see Comparison of Aurora MySQL version 2 and Aurora MySQL version 3. For a comparison of Aurora MySQL version 3 and MySQL 8.0 Community Edition, see Comparison of Aurora MySQL version 3 and MySQL 8.0 Community Edition.

Currently available Aurora MySQL releases are 2.07.9, 2.07.10, 2.11.*, 3.01.*, 3.02.*, 3.03.*, and 3.04.*.

You can perform an in-place upgrade, restore a snapshot, or initiate a managed blue/green upgrade using Amazon RDS Blue/Green Deployments from any currently available Aurora MySQL version 2 cluster into an Aurora MySQL version 3.03.2 cluster.

For information on planning an upgrade to Aurora MySQL version 3, see Upgrade planning for Aurora MySQL version 3 in the *Amazon Aurora User Guide*. For general information about Aurora MySQL upgrades, see Upgrading Amazon Aurora MySQL DB clusters in the *Amazon Aurora User Guide*.

For troubleshooting information, see Troubleshooting upgrade issues with Aurora MySQL version 3.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Improvements

Fixed security issues and CVEs:

- Fixed an issue that might cause the audit log to miss events during audit log file rotation.

The following CVE fixes are included in this release:

- CVE-2023-21963
- CVE-2023-21912
- CVE-2023-0215
- CVE-2022-43551
- CVE-2022-37434

Availability improvements:

- Fixed an issue that can cause database restarts during long transaction recovery.
- Fixed an issue that can cause the database cluster to become unavailable when the writer instance restarts while the database was creating or dropping triggers on internal system tables.
- Fixed an issue that can cause a database instance to restart while executing a query which references an aggregate function.
- Fixed an issue that can cause a database restart during the rollback of an `INSERT` statement when parallel query is enabled.
- Fast insert is enabled only for regular InnoDB tables in Aurora MySQL version 3.03.2 and higher. This optimization doesn't work for InnoDB temporary tables. For more information on the fast insert optimization, see Amazon Aurora MySQL performance enhancements.

General improvements:

- Fixed an issue where the reader instance is unable to open a table, with `ERROR 1146`. This issue occurs when executing certain types of online Data Definition Language (DDL) while the `INPLACE` algorithm is being used on the writer instance.
- Introduced file management performance optimizations on binlog replicas to help reduce contention when writing to relay log files.
- Fixed an issue when parallel query is enabled that causes the query plan optimizer to pick an inefficient execution plan for certain `SELECT` queries that benefit from a primary or secondary index.
- Added logical replication support for the following Data Control Language (DCL) statements: `GRANT/REVOKE` and `CREATE/DROP/ALTER/RENAME USER`.

- Parallel query for Amazon Aurora MySQL is not supported when choosing the Aurora I/O-Optimized cluster configuration. See Limitations of parallel query for Aurora MySQL for more information.

Upgrades and migrations:

- To perform a minor version upgrade for an Aurora global database from Aurora MySQL version 3.01 or 3.02 to Aurora MySQL version 3.03 or higher, refer to Upgrading Aurora MySQL by modifying the engine version.
- Fixed an issue which can cause major version upgrade failures when upgrading to Aurora MySQL version 3 when a trigger definition contains a reserved keyword that is not enclosed in quotation marks.

Integration of MySQL Community Edition bug fixes

This release includes all community bug fixes up to and including 8.0.26, in addition to the below. For more information, see MySQL bugs fixed by Aurora MySQL 3.x database engine updates.

- Fixed an issue which can sometimes produce incorrect query results while processing complex SQL statements consisting of multiple nested Common Table Expressions (CTEs). (Bug #34572040, Bug #34634469, Bug #33856374)
- InnoDB: A race condition between threads attempting to deinitialize and initialize statistics for the same table raised an assertion failure. (Bug #33135425)
- InnoDB: Prevent online DDL operations from accessing out-of-bounds memory. (Bug #34750489, Bug #108925)

Aurora MySQL database engine updates 2023-05-11 (version 3.03.1, compatible with MySQL 8.0.26)

Version: 3.03.1

Aurora MySQL 3.03.1 is generally available. Aurora MySQL 3.03 versions are compatible with MySQL 8.0.26, and Aurora MySQL 3.02 versions are compatible with MySQL 8.0.23. For more information on community changes that have occurred from 8.0.23 to 8.0.26, see MySQL 8.0 Release Notes.

For details of the new features in Aurora MySQL version 3, see Aurora MySQL version 3 compatible with MySQL 8.0. For differences between Aurora MySQL version 3 and Aurora MySQL version 2, see Comparison of Aurora MySQL version 2 and Aurora MySQL version 3. For a comparison of Aurora MySQL version 3 and MySQL 8.0 Community Edition, see Comparison of Aurora MySQL version 3 and MySQL 8.0 Community Edition.

Currently supported Aurora MySQL releases are 2.07.9, 2.11.1, 2.11.2, 3.01.*, 3.02.* and 3.03.*.

You can perform an in-place upgrade or restore a snapshot from any currently supported Aurora MySQL version 2 cluster into Aurora MySQL 3.03.1.

For information on planning an upgrade to Aurora MySQL version 3, see Upgrade planning for Aurora MySQL version 3 in the *Amazon Aurora User Guide*. For general information about Aurora MySQL upgrades, see Upgrading Amazon Aurora MySQL DB clusters in the *Amazon Aurora User Guide*.

For troubleshooting information, see Troubleshooting upgrade issues with Aurora MySQL version 3.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Improvements

New features:

- Aurora Optimized I/O storage configuration is available starting from version 3.03.1. For more information, see Storage configurations for Amazon Aurora DB clusters.
- Added a new system variable, `innodb_aurora_max_partitions_for_range`. In some cases where persisted stats are not available, this parameter can be used to improve the execution time of row count estimations on partitioned tables. More information can be found in the documentation, Aurora MySQL configuration parameters.

Availability improvements:

- Fixed an issue which can cause the database instance to restart due to incorrectly accessing invalid memory when a connection is closed immediately after committing a transaction.

- Fixed an issue with Aurora Advanced Auditing that causes excess logging of informational messages to the Aurora MySQL error log when the server variable `server_audit_events` is set to ALL or QUERY. This issue may cause a database instance restart.
- Fixed an issue which, in certain situations, can cause Aurora reader instances to restart when attempting to read a page which is no longer accessible during a range estimation.
- Fixed an issue which can cause an Aurora MySQL reader instance to restart while executing a query which utilizes an Aurora parallel query execution plan.
- Fixed an issue where database instances using binary log replication may experience an increase in CPU utilization and connection failures when multiple binary log replication consumers are attached.
- Fixed an issue where unsupported index scan access methods were considered for common table expressions (CTE) while materializing intermediate temporary tables, which can lead to undesired behavior including database restarts or incorrect query results. We fix this issue by avoiding the use of such unsupported index scan access methods on tables using the TempTable storage engine.
- Fast insert isn't enabled in this Aurora MySQL version, due to an issue that can cause inconsistencies when running queries such as `INSERT INTO`, `SELECT`, and `FROM`. For more information on the fast insert optimization, see Amazon Aurora MySQL performance enhancements.

General improvements:

- Fixed an issue which can cause higher than expected execution times for the SHOW BINARY LOGS statement. This could lead to a drop in commit throughput of the database.
- Fixed an issue which can cause parallel export for user tables that have columns added using the Instant ADD COLUMN functionality to fail.
- Fixed an issue where events that were reported while processing audit log rotations might not be written to the audit log.
- Fixed an issue which may cause depletion of available memory when executing queries against the INFORMATION_SCHEMA INNODB_TABLESPACES table.
- Fixed an issue which incorrectly allows customers to set ROW_FORMAT as COMPRESSED when creating partitioned tables. Tables will be implicitly converted to COMPACT format with a warning to inform that Aurora MySQL doesn’t support compressed tables.

Upgrades and migrations:

- To perform a minor version upgrade for an Aurora global database from Aurora MySQL version 3.01 or 3.02 to Aurora MySQL version 3.03 or higher, refer to Upgrading Aurora MySQL by modifying the engine version.
- Fixed an issue that can cause upgrade precheck failures due to schema inconsistency errors reported for the `mysql.general_log_backup`, `mysql.general_log`, `mysql.slow_log_backup` and `mysql.slow_log` tables when upgrading from Aurora MySQL 2 to Aurora MySQL 3. For more information about upgrade troubleshooting, see Troubleshooting upgrade issues with Aurora MySQL version 3.

Integration of MySQL Community Edition bug fixes

This release includes all community bug fixes up to and including 8.0.26, in addition to the below. For more information, see MySQL bugs fixed by Aurora MySQL 3.x database engine updates.

- Fixed an issue where a buffer block containing intrinsic temporary table page was relocated during page traversal, causing an assertion failure. (Bug #33715694)

Aurora MySQL database engine updates 2023-03-01 (version 3.03.0, compatible with MySQL 8.0.26) Upgrades to this version aren't supported.

Version: 3.03.0

Aurora MySQL 3.03.0 is generally available. Aurora MySQL 3.03 versions are compatible with MySQL 8.0.26, and Aurora MySQL 3.02 versions are compatible with MySQL 8.0.23. For more information on community changes that have occurred from 8.0.23 to 8.0.26, see MySQL 8.0 Release Notes.

For details of the new features in Aurora MySQL version 3, see Aurora MySQL version 3 compatible with MySQL 8.0. For differences between Aurora MySQL version 3 and Aurora MySQL version 2, see the Comparison of Aurora MySQL version 2 and Aurora MySQL version 3. For a comparison of Aurora MySQL version 3 and MySQL 8.0 Community Edition, see Comparison of Aurora MySQL version 3 and MySQL 8.0 Community Edition.

Currently supported Aurora MySQL releases are 2.07.*, 2.11.*, 3.01.*, 3.02.* and 3.03.*.

You can perform an in-place upgrade or restore a snapshot from any currently supported Aurora MySQL version 2 cluster into Aurora MySQL 3.03.0.

For information on planning an upgrade to Aurora MySQL version 3, see Upgrade planning for Aurora MySQL version 3 in the *Amazon Aurora User Guide*. For general information about Aurora MySQL upgrades, see Upgrading Amazon Aurora MySQL DB clusters in the *Amazon Aurora User Guide*.

For troubleshooting information, see Troubleshooting upgrade issues with Aurora MySQL version 3.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Improvements

Fixed security issues and CVEs listed below:

Fixes and other enhancements to fine-tune handling in a managed environment. Additional CVE fixes below:

- CVE-2022-32221
- CVE-2022-21451
- CVE-2022-21444

Availability improvements:

- Fixed an issue where larger DB instance classes may experience issues during restart due to the buffer pool initialization taking longer than expected.
- Fixed an issue where the DB instance may restart during the database recovery process when binary logging is enabled.
- Fixed an issue that can cause connection failures on reader instances while executing Data Control Language (DCL) statements, for example `GRANT` and `REVOKE`, or while establishing new connections on the writer instance.
- Fixed an issue where parallel query was incorrectly used for Data Manipulation Language (DML) operations, such as the `DELETE` and `UPDATE` statements, which aren't currently supported, that

led to a DB instance restart. For more information on operations supported in parallel query, see Aurora MySQL parallel query limitations.

- Fixed an issue which, in rare cases, can cause Aurora replicas to restart during the simultaneous execution of large update operations or Data Definition Language (DDL) workloads on the writer instance and read operations on the same set of tables on the Aurora replica.
- Fixed an issue with the Aurora Serverless v2 reader instance scale down operation that can cause that reader instance to restart, and in some rare cases, cause data inconsistency.
- Fixed an issue which can cause the DB instance to restart due to incorrectly accessing an invalid memory location when a connection to the DB instance is closed.
- Fixed an issue which, in rare conditions, can cause the DB instance to restart while processing a query with a `GROUP BY` clause that truncates a decimal column to zero decimal places.
- Fixed an issue that can cause a DB instance restart due to incorrectly accessing a record when executing a range query using spatial index.
- Fixed an issue that can cause a DB instance restart on Aurora MySQL replica instances when internal temporary tables exceed the default or customer-configured memory or mmap values.
- Fixed an issue where the Advanced Audit log rotation may cause memory management issues.
- Fast insert isn't enabled in this Aurora MySQL version, due to an issue that can cause inconsistencies when running queries such as `INSERT INTO`, `SELECT`, and `FROM`. For more information on the fast insert optimization, see Amazon Aurora MySQL performance enhancements.

General improvements:

- Improved the read query latency of global database write forwarding sessions using the GLOBAL read consistency setting.
- Fixed an issue where the `wait_timeout` parameter value wasn't being honored after a client session executed the `reset_connection` or `change_user` commands.
- Fixed an issue where applications may experience increased latency while connecting to a DB instance when the instance is experiencing a sudden increase in incoming connections. Two new CloudWatch metrics, `AuroraSlowHandshakeCount` and `AuroraSlowConnectionHandleCount`, were introduced to help troubleshoot connection establishment delays for Aurora MySQL DB instances. More information on these metrics can be found in the Aurora CloudWatch metrics definitions documentation, Amazon CloudWatch metrics for Amazon Aurora.

- The `temptable_use_mmap` parameter has been deprecated, and support for it is expected to be removed in a future MySQL release. For more information, see Storage engine for internal (implicit) temporary tables.
- Fixed an issue which can cause higher than expected execution times for the `SHOW BINARY LOGS` statement. This could lead to a drop in the commit throughput of the database.

Upgrades and migrations:

- To perform a minor version upgrade for an Aurora global database from Aurora MySQL version 3.01 or 3.02 to Aurora MySQL version 3.03 or higher, refer to Upgrading Aurora MySQL by modifying the engine version.
- Fixed an issue that can cause major version upgrades from Aurora MySQL version 2 to Aurora MySQL version 3 to fail when there are a large number of tables (over 750K) in the cluster.
- Fixed an issue that can cause major version upgrades from Aurora MySQL version 2 to Aurora MySQL version 3 to fail because migrating the `mysql.innodb_table_stats` and `mysql.innodb_index_stats` tables took longer than expected. This issue mainly affected DB clusters with millions of tables.
- Fixed an issue that can cause failures while upgrading from Aurora MySQL version 2 to Aurora MySQL version 3 due to schema inconsistency errors. These errors are reported by the upgrade pre-checker for the `mysql.general_log_template` and `mysql.slow_log_template` tables. For more information about upgrade troubleshooting, see Troubleshooting upgrade issues with Aurora MySQL version 3.
- Fixed an issue that can cause upgrade failures from Aurora MySQL version 2 to Aurora MySQL version 3 due to the `schemaInconsistencyCheck` error. This error is caused by schema inconsistencies within the `mysql.table_migration_index_info` table, as reported by the `upgrade-prechecks.log`. For more information about troubleshooting upgrades to Aurora MySQL version 3, see Troubleshooting upgrade issues with Aurora MySQL version 3.

Integration of MySQL Community Edition bug fixes

This release includes all community bug fixes up to and including 8.0.26, in addition to the below. For more information, see MySQL bugs fixed by Aurora MySQL 3.x database engine updates.

- Fixed an issue where sorts of some column types, including `JSON` and `TEXT`, sometimes exhausted the sort buffer if its size wasn't at least 15 times that of the largest row in the sort.

Now the sort buffer need only be 15 times as large as the largest sort key. (Bug #103325, Bug #105532, Bug #32738705, Bug #33501541)

- Fixed an issue where InnoDB did't always handle some legal names for table partitions correctly. (Bug #32208630)
- Fixed an issue which, in certain conditions, may return incorrect results due to an inaccurate calculation of the nullability property when executing a query with an `OR` condition. (Bug #34060289)
- Fixed an issue which, in certain conditions, may return incorrect results when the following two conditions are met:
  - a derived table is merged into the outer query block
  - the query includes a left join and an `IN` subquery

  (Bug #34060289)
- Fixed an issue where incorrect `AUTO_INCREMENT` values were generated when the maximum integer column value was exceeded. The error was due to the maximum column value not being considered. The previous valid `AUTO_INCREMENT` value should have been returned in this case, causing a duplicate key error. (Bug #87926, Bug #26906787)
- Fixed an issue where it wasn't possible to revoke the `DROP` privilege on the Performance Schema. (Bug #33578113)
- Fixed an issue where a stored procedure containing an `IF` statement using `EXISTS`, which acted on one or more tables that were deleted and recreated between executions, didn't execute correctly for the subsequent invocations following the first one. (Bug #32855634).
- Fixed an issue where a query that references a view in a subquery and an outer query block can cause an unexpected restart. (Bug#32324234)

Aurora MySQL database engine updates 2023-04-17 (version 3.02.3, compatible with MySQL 8.0.23) The end of standard support is January 15, 2024.

Version: 3.02.3

Aurora MySQL 3.02.3 is generally available. Aurora MySQL 3.02 versions are compatible with MySQL 8.0.23, and Aurora MySQL 2.x versions are compatible with MySQL 5.7.

For details of the new features in Aurora MySQL version 3, and differences between Aurora MySQL version 3 and Aurora MySQL version 2 or community MySQL 8.0, see Comparison of Aurora MySQL version 2 and Aurora MySQL version 3 in the *Amazon Aurora User Guide*.

Currently supported Aurora MySQL releases are 2.07.*, 2.11.1, 2.11.2, 3.01.*, 3.02.*, and 3.03.*.

You can perform an in-place upgrade or restore a snapshot from any currently supported Aurora MySQL version 2 cluster into Aurora MySQL 3.02.3.

For information on planning an upgrade to Aurora MySQL version 3, see Upgrade planning for Aurora MySQL version 3 in the *Amazon Aurora User Guide*. For the upgrade procedure itself, see Upgrading to Aurora MySQL version 3 in the *Amazon Aurora User Guide*. For general information about Aurora MySQL upgrades, see Upgrading Amazon Aurora MySQL DB clusters in the *Amazon Aurora User Guide*.

For troubleshooting information, see Troubleshooting upgrade issues with Aurora MySQL version 3.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Improvements

Availability improvements:

- Fixed an issue which can cause the database instance to restart due to incorrectly accessing invalid memory when a connection is closed immediately after committing a transaction.
- Fast insert isn't enabled in this Aurora MySQL version, due to an issue that can cause inconsistencies when running queries such as `INSERT INTO`, `SELECT`, and `FROM`. For more information on the fast insert optimization, see Amazon Aurora MySQL performance enhancements.

General improvements:

- Fixed an issue where unsupported index scan access methods were considered for common table expressions (CTE) while materializing intermediate temporary tables, which can lead to undesired behavior including database restarts or incorrect query results. This issue was fixed by avoiding the use of such unsupported index scan access methods on tables using the TempTable storage engine.

- Fixed an issue which, in rare cases, can cause an Aurora MySQL reader instance to restart when accessing a table which has large update or Data Definition Language (DDL) operations running concurrently on the Aurora MySQL writer instance.
- Fixed an issue which, in certain situations, can cause Aurora MySQL reader instances to restart when attempting to read a page which is no longer accessible during a range estimation.
- Fixed an issue where database instances using binary log replication may experience an increase in CPU utilization and connection failures when multiple binary log replication consumers are attached.
- Fixed an issue which can cause an Aurora MySQL reader instance to restart while executing a query which utilizes an Aurora parallel query execution plan.

Aurora MySQL database engine updates 2022-11-18 (version 3.02.2, compatible with MySQL 8.0.23) The end of standard support is January 15, 2024.

Version: 3.02.2

Aurora MySQL 3.02.2 is generally available. Aurora MySQL 3.02 versions are compatible with MySQL 8.0.23, Aurora MySQL 2.x versions are compatible with MySQL 5.7, and Aurora MySQL 1.x versions are compatible with MySQL 5.6.

For details of new features in Aurora MySQL version 3 and differences between Aurora MySQL version 3 and Aurora MySQL version 2 or community MySQL 8.0, see Comparison of Aurora MySQL version 2 and Aurora MySQL version 3 in the *Amazon Aurora User Guide*.

Currently supported Aurora MySQL releases are 1.19.5, 1.19.6, 1.22.*, 1.23.*, 2.04.*, 2.07.*, 2.08.*, 2.09.*, 2.10.*, 3.01.* and 3.02.*.

You can restore a snapshot from any currently supported Aurora MySQL version 2 cluster into Aurora MySQL 3.02.2.

For information on planning an upgrade to Aurora MySQL version 3, see Upgrade planning for Aurora MySQL version 3 in the *Amazon Aurora User Guide*. For the upgrade procedure itself, see Upgrading to Aurora MySQL version 3 in the *Amazon Aurora User Guide*. For general information about Aurora MySQL upgrades, see Upgrading Amazon Aurora MySQL DB clusters in the *Amazon Aurora User Guide*.

For troubleshooting information, see Troubleshooting upgrade issues with Aurora MySQL version 3.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Improvements

Aurora MySQL version 3.02.2 is generally available and generally compatible with community MySQL 8.0.23.

Fixed security issues and CVEs listed below:

Fixes and other enhancements to fine-tune handling in a managed environment. Additional CVE fixes below:

- CVE-2022-21451
- CVE-2021-36222
- CVE-2021-22926

- CVE-2022-21444

Availability improvements:

- Fixed an issue which can cause the database instance to restart due to incorrectly accessing the invalid memory when a connection to the database instance is closed explicitly or implicitly.
- Fixed an issue which can cause the database startup to be interrupted repeatedly on larger instance classes due to buffer pool initialization taking longer than expected.
- Fixed an issue which, in rare conditions, can cause the database instance to restart when Aurora Serverless v2 incorrectly attempts to update the table cache while scaling.
- Fixed an issue which, in rare conditions, could cause the database to restart when processing a query with a GROUP BY clause that truncates a decimal column to zero decimal places.
- Fast insert isn't enabled in this Aurora MySQL version, due to an issue that can cause inconsistencies when running queries such as `INSERT INTO`, `SELECT`, and `FROM`. For more information on the fast insert optimization, see Amazon Aurora MySQL performance enhancements.

General improvements:

- Fixed an issue that can cause upgrade failures from Aurora MySQL version 2 (compatible with MySQL 5.7) to Aurora MySQL version 3 (compatible with MySQL 8.0) due to a metadata inconsistency in the mysql.host table.
- Added performance improvements to reduce the upgrade time from Aurora MySQL version 2 (compatible with MySQL 5.7) to Aurora MySQL version 3 (compatible with MySQL 8.0). By parallelizing certain upgrade steps, the time is further shortened when using larger instance classes, such as db.r6g.16xlarge or db.r5.24xlarge.
- Added support for displaying all errors when upgrading from Aurora MySQL version 2 (compatible with MySQL 5.7) to Aurora MySQL version 3 (compatible with MySQL 8.0) when previous versions were limited to displaying only 50 errors.
- Fixed an issue, which in rare conditions, can cause the auto-increment counters to be incorrect after a major version upgrade from Aurora MySQL version 2 (compatible with MySQL 5.7) to Aurora MySQL version 3 (compatible with MySQL 8.0).
- Fixed an issue that can cause major version upgrades from Aurora MySQL version 2 to Aurora MySQL version 3 to fail because migrating the `mysql.innodb_table_stats` and `mysql.innodb_index_stats` tables took longer than expected. This issue mainly affected database clusters with a large numbers of tables (>1.5 million).
- Fixed an issue that can cause major version upgrades from Aurora MySQL version 2 to Aurora MySQL version 3 to fail due to a defect in AMS 8.0 engine upgrade workflow, which causes the log records to be accumulated on the Aurora storage cluster volume and stops normal write operations. This issue mainly affected database clusters with large numbers of tables, approximately >750k.
- Fixed an issue that prevents Aurora MySQL Serverless v2 idle instances from scaling down to 0.5 ACUs because the MySQL purge threads were incorrectly kept active.
- Fixed an issue where applications may experience increased latency while connecting to a database instance when the instance is experiencing a sudden increase in incoming connections.
- Introduced two new Amazon CloudWatch metrics to help troubleshoot connection establishment delays for Aurora MySQL database instances. More information on AuroraSlowHandshakeCount and AuroraSlowConnectionHandleCount metrics can be found in the Aurora CloudWatch metrics definitions.

Integration of MySQL Community Edition bug fixes

This release includes all community bug fixes up to and including 8.0.23, in addition to the below. For more information, see MySQL bugs fixed by Aurora MySQL 3.x database engine updates.

- Fixed an issue which, in certain conditions, may return incorrect results due to an inaccurate calculation of the nullability property when executing a query with an OR condition. (Bug #34060289)
- Fixed an issue which, in certain conditions, may return incorrect results when the following two conditions are met:
  - A derived table is merged into the outer query block.
  - The query includes a left join and an IN subquery. (Bug #34060289)
- Fixed an issue where it was not possible to revoke the DROP privilege on the Performance Schema. (Bug #33578113)
- Fixed an issue where a stored procedure containing an IF statement using EXISTS, which acted on one or more tables that were deleted and recreated between executions, did not execute correctly for subsequent invocations following the first one. (MySQL Bug #32855634).
- Incorrect AUTO_INCREMENT values were generated when the maximum integer column value was exceeded. The error was due to the maximum column value not being considered. The previous valid AUTO_INCREMENT value should have been returned in this case, causing a duplicate key error. (Bug #87926, Bug #26906787)
- Fixed an issue which can lead to a failure while upgrading an Aurora MySQL version 1 (Compatible with MySQL 5.6) database cluster containing user-created table with certain table IDs. Assignment of these table IDs may result in conflicting data dictionary table IDs while upgrading from Aurora MySQL version 2 (Compatible with MySQL 5.7) to Aurora MySQL version 3 (Compatible with MySQL 8.0) (Bug #33919635)

Aurora MySQL database engine updates 2022-09-07 (version 3.02.1, compatible with MySQL 8.0.23) The end of standard support is January 15, 2024. Upgrades to this version aren't supported.

Version: 3.02.1

Aurora MySQL 3.02.1 is generally available. Aurora MySQL 3.02 versions are compatible with MySQL 8.0.23, Aurora MySQL 2.x versions are compatible with MySQL 5.7, and Aurora MySQL 1.x versions are compatible with MySQL 5.6.

For details of new features in Aurora MySQL version 3 and differences between Aurora MySQL version 3 and Aurora MySQL version 2 or community MySQL 8.0, see Comparison of Aurora MySQL version 2 and Aurora MySQL version 3 in the *Amazon Aurora User Guide*.

Currently supported Aurora MySQL releases are 1.19.5, 1.19.6, 1.22.*, 1.23.*, 2.04.*, 2.07.*, 2.08.*, 2.09.*, 2.10.*, 3.01.* and 3.02.*.

For information on planning an upgrade to Aurora MySQL version 3, see Upgrade planning for Aurora MySQL version 3 in the *Amazon Aurora User Guide*. For the upgrade procedure itself, see Upgrading to Aurora MySQL version 3 in the *Amazon Aurora User Guide*. For general information about Aurora MySQL upgrades, see Upgrading Amazon Aurora MySQL DB clusters in the *Amazon Aurora User Guide*.

For troubleshooting information, see Troubleshooting upgrade issues with Aurora MySQL version 3.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Improvements

Aurora MySQL version 3.02.1 is generally available and generally compatible with community MySQL 8.0.23.

Fixed security issues and CVEs listed below:

Fixes and other enhancements to fine-tune handling in a managed environment. Additional CVE fixes below:

- CVE-2022-0778

Availability improvements:

- Fixed an issue which can cause connection failure and high latency when multiple MySQL binary log (binlog) replicas are attached to an Aurora writer node or when there are a large number of concurrent long running queries in conjunction with a surge in new connection requests.

- Fixed an issue that causes a database restart when advanced auditing for `CONNECT` events are turned on.
- Fixed an issue that can cause a database restart on Aurora MySQL read replica instances when internal temporary tables exhaust the allocated size in memory and mmap files set as a customer-configured or default value.
- Fixed an issue that can cause a read replica to repeatedly restart during concurrent DDL operations on stored procedures.
- Fast insert isn't enabled in this Aurora MySQL version, due to an issue that can cause inconsistencies when running queries such as `INSERT INTO`, `SELECT`, and `FROM`. For more information on the fast insert optimization, see Amazon Aurora MySQL performance enhancements.

General improvements:

- Added support for R6i instances.

Additional Information:

- Aurora MySQL version 3.02.1 does not contain support for major version upgrades directly from Aurora MySQL version 2 (compatible with MySQL 5.7). To perform a major version upgrade to this version, first perform a major version upgrade to Aurora MySQL version 3.02.0, then perform an in-place minor version upgrade to Aurora MySQL version 3.02.1.

Aurora MySQL database engine updates 2022-04-20 (version 3.02.0, compatible with MySQL 8.0.23) The end of standard support is January 15, 2024. Upgrades to this version aren't supported.

Version: 3.02.0

Aurora MySQL 3.02.0 is generally available. Aurora MySQL 3.02 versions are compatible with MySQL 8.0.23, Aurora MySQL 2.x versions are compatible with MySQL 5.7, and Aurora MySQL 1.x versions are compatible with MySQL 5.6.

For details of new features in Aurora MySQL version 3 and differences between Aurora MySQL version 3 and Aurora MySQL version 2 or community MySQL 8.0, see Comparison of Aurora MySQL version 2 and Aurora MySQL version 3 in the *Amazon Aurora User Guide*.

Currently supported Aurora MySQL releases are 1.19.5, 1.19.6, 1.22.*, 1.23.*, 2.04.*, 2.07.*, 2.08.*, 2.09.*, 2.10.*, 3.01.* and 3.02.*.

You can restore a snapshot from any currently supported Aurora MySQL version 2 cluster into Aurora MySQL 3.02.0.

For information on planning an upgrade to Aurora MySQL version 3, see Upgrade planning for Aurora MySQL version 3 in the *Amazon Aurora User Guide*. For the upgrade procedure itself, see Upgrading to Aurora MySQL version 3 in the *Amazon Aurora User Guide*. For general information about Aurora MySQL upgrades, see Upgrading Amazon Aurora MySQL DB clusters in the *Amazon Aurora User Guide*.

For troubleshooting information, see Troubleshooting upgrade issues with Aurora MySQL version 3.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Improvements

Aurora MySQL version 3.02.0 is generally available and generally compatible with community MySQL 8.0.23.

Fixed security issues and CVEs listed below:

Fixes and other enhancements to fine-tune handling in a managed environment. Additional CVE fixes below:

- CVE-2021-22946

New features:

- Amazon Aurora Serverless v2 is generally available. For more information, see the Amazon Aurora Serverless overview, blog, and Using Aurora Serverless v2 documentation. Get started

today by creating an Aurora Serverless v2 database using only a few steps in the AWS Management Console.

Availability improvements:

- Fixed an issue that can cause the server to potentially go into a restart loop and cause unavailability while deleting a record or dropping a table containing two or more variable-length columns (VARCHAR, VARBINARY, BLOB, and TEXT types). For more details about column types, see innodb-row-format.
- Fixed an issue where existing connections timeout and new connections could not be established on a cluster with Binary Log turned on and has at least one Binary Log consumer attached that results in resource contention between the application and the consumer(s).
- Freeable memory is indicated by the `FreeableMemory` CloudWatch metric. For more information, see Amazon CloudWatch metrics for Amazon Aurora.
  - Fixed an issue that can cause a DB instance restart or a failover due to a decrease in freeable memory when binary log replication is enabled.
  - Fixed an issue that can cause a DB instance restart or a failover due to a decrease in freeable memory when setting session variables.
  - Fixed an issue that can cause a DB instance restart or a failover due to a decrease in freeable memory when the database process opens an existing file.
- Fixed an issue which, in rare conditions, can cause a duplicate entry error when inserting new rows into a table containing an `AUTO_INCREMENT` column on a cluster restored from snapshot.
- Fast insert isn't enabled in this Aurora MySQL version, due to an issue that can cause inconsistencies when running queries such as `INSERT INTO`, `SELECT`, and `FROM`. For more information on the fast insert optimization, see Amazon Aurora MySQL performance enhancements.

General improvements:

- Fixed an issue where the volume status was not shown when using the `SHOW VOLUME STATUS` command. For more information, see AuroraMySQL.Managing.VolumeStatus.
- Fixed an issue that caused calls to mysql_rds_import_binlog_ssl_material to fail with MySQL server ERROR 3512.
- Fixed an issue where Aurora replica lag is incorrectly reported for deleted Aurora reader instances.

Upgrades/Migration:

- Fixed an issue that can cause migration failures of MySQL 8.0.x databases to Aurora MySQL version 3 due to an issue in copying ibdata files and tablespaces to Aurora storage.
- Fixed an issue which can cause upgrades of clusters from Aurora MySQL version 2 to Aurora MySQL version 3 to fail when database tables contained a large amount of data.
- Fixed an issue that can cause failures when restoring clusters from Aurora MySQL version 2 to Aurora MySQL version 3 due to a failure in creating serialized data dictionary information (SDI) for a table.
- Fixed an issue that can cause upgrade failures from Aurora MySQL version 2 to Aurora MySQL version 3 due to schema inconsistency errors reported by upgrade prechecks for RDS system tables.
- Fixed an issue that can cause failures when migrating or restoring from RDS for MySQL 8.0 or Aurora MySQL version 2 to Aurora MySQL version 3 databases due to invalid syntax in an RDS managed stored procedure.
- Fixed an issue that can cause upgrade failures from Aurora MySQL 2 to Aurora MySQL 3 due to schema inconsistency errors reported by upgrade prechecks for the general log and slow log tables.

Integration of MySQL community edition bug fixes

This release includes all community bug fixes up to and including 8.0.23, in addition to the below. For more information, see MySQL bugs fixed by Aurora MySQL 3.x database engine updates.

- Fixed the improper handling of temporary tables used for cursors within stored procedures that could result in unexpected server behavior, mysqld-8-0-24-bug. (Bug #32416811)

Aurora MySQL database engine updates 2022-04-15 (version 3.01.1, compatible with MySQL 8.0.23) The end of standard support is January 15, 2024. Upgrades to this version aren't supported.

Version: 3.01.1

Aurora MySQL 3.01.1 is generally available. Aurora MySQL 3.01 versions are compatible with MySQL 8.0.23, Aurora MySQL 2.x versions are compatible with MySQL 5.7, and Aurora MySQL 1.x versions are compatible with MySQL 5.6.

For details of new features in Aurora MySQL version 3 and differences between Aurora MySQL version 3 and Aurora MySQL version 2 or community MySQL 8.0, see Comparison of Aurora MySQL version 2 and Aurora MySQL version 3 in the *Amazon Aurora User Guide*.

Currently supported Aurora MySQL releases are 1.19.5, 1.19.6, 1.22.*, 1.23.*, 2.04.*, 2.07.*, 2.08.*, 2.09.*, 2.10.*, 3.01.* and 3.02.*.

You can restore a snapshot from any currently supported Aurora MySQL version 2 cluster into Aurora MySQL 3.01.1.

For information on planning an upgrade to Aurora MySQL version 3, see Upgrade planning for Aurora MySQL version 3 in the *Amazon Aurora User Guide*. For the upgrade procedure itself, see Upgrading to Aurora MySQL version 3 in the *Amazon Aurora User Guide*. For general information about Aurora MySQL upgrades, see Upgrading Amazon Aurora MySQL DB clusters in the *Amazon Aurora User Guide*.

For troubleshooting information, see Troubleshooting upgrade issues with Aurora MySQL version 3.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Improvements

Aurora MySQL version 3.01.1 is generally available and generally compatible with community MySQL 8.0.23.

Aurora MySQL version 3.01.1 is recommended for upgrades and migrations to a MySQL 8.0 compatible Aurora database.

Fixed security issues and CVEs listed below:

Fixes and other enhancements to fine-tune handling in a managed environment. Additional CVE fixes below:

- CVE-2021-36222

- CVE-2021-22946
- CVE-2021-22926

Availability improvements:

- Freeable memory is indicated by the `FreeableMemory` CloudWatch metric. For more information, see Amazon CloudWatch metrics for Amazon Aurora.
  - Fixed an issue that can cause a DB instance restart or a failover due to a decrease in freeable memory when binary log replication is enabled.
  - Fixed an issue that can cause a DB instance restart or a failover due to a decrease in freeable memory when setting session variables.
  - Fixed an issue that can cause a DB instance restart or a failover due to a decrease in freeable memory when the database process opens an existing file.
- Fixed an issue which, in rare conditions, can cause a duplicate entry error when inserting new rows into a table containing an `AUTO_INCREMENT` column on a cluster restored from snapshot.
- Fast insert isn't enabled in this Aurora MySQL version, due to an issue that can cause inconsistencies when running queries such as `INSERT INTO`, `SELECT`, and `FROM`. For more information on the fast insert optimization, see Amazon Aurora MySQL performance enhancements.

General improvements:

- Fixed an issue where the volume status was not shown when using the `SHOW VOLUME STATUS` command. For more information, see AuroraMySQL.Managing.VolumeStatus.
- Fixed an issue that caused calls to mysql_rds_import_binlog_ssl_material to fail with MySQL server ERROR 3512.
- Fixed an issue where Aurora replica lag is incorrectly reported for deleted Aurora reader instances.

Upgrades/Migration:

- Fixed an issue that can cause migration failures of MySQL 8.0.x databases to Aurora MySQL version 3 due to an issue in copying ibdata files and tablespaces to Aurora storage.
- Fixed an issue which can cause upgrades of clusters from Aurora MySQL version 2 to Aurora MySQL version 3 to fail when database tables contained a large amount of data.

- Fixed an issue that can cause failures when restoring clusters from Aurora MySQL version 2 to Aurora MySQL version 3 due to a failure in creating serialized data dictionary information (SDI) for a table.
- Fixed an issue that can cause upgrade failures from Aurora MySQL version 2 to Aurora MySQL version 3 due to schema inconsistency errors reported by upgrade prechecks for RDS system tables.
- Fixed an issue that can cause failures when migrating or restoring from RDS for MySQL 8.0 or Aurora MySQL version 2 to Aurora MySQL version 3 databases due to invalid syntax in an RDS managed stored procedure.
- Fixed an issue that can cause upgrade failures from Aurora MySQL 2 to Aurora MySQL 3 due to schema inconsistency errors reported by upgrade prechecks for the general log and slow log tables.

Integration of MySQL community edition bug fixes

This release includes all community bug fixes up to and including 8.0.23, in addition to the below. For more information, see MySQL bugs fixed by Aurora MySQL 3.x database engine updates.

- Fixed the improper handling of temporary tables used for cursors within stored procedures that could result in unexpected server behavior, mysqld-8-0-24-bug. (Bug #32416811)

Aurora MySQL database engine updates 2021-11-18 (version 3.01.0, compatible with MySQL 8.0.23) The end of standard support is January 15, 2024. Upgrades to this version aren't supported.

Version: 3.01.0

Aurora MySQL 3.01.0 is generally available. Aurora MySQL 3.01 versions are compatible with MySQL 8.0.23, Aurora MySQL 2.x versions are compatible with MySQL 5.7, and Aurora MySQL 1.x versions are compatible with MySQL 5.6.

For details of new features in Aurora MySQL version 3 and differences between Aurora MySQL version 3 and Aurora MySQL version 2 or community MySQL 8.0, see Comparison of Aurora MySQL version 2 and Aurora MySQL version 3 in the *Amazon Aurora User Guide*.

Currently supported Aurora MySQL releases are 1.19.5, 1.19.6, 1.22.*, 1.23.*, 2.04.*, 2.07.*, 2.08.*, 2.09.*, 2.10.*, 3.01.* and 3.02.*.

You can restore a snapshot from any currently supported Aurora MySQL version 2 cluster into Aurora MySQL 3.01.0.

For information on planning an upgrade to Aurora MySQL version 3, see Upgrade planning for Aurora MySQL version 3 in the *Amazon Aurora User Guide*. For the upgrade procedure itself, see Upgrading to Aurora MySQL version 3 in the *Amazon Aurora User Guide*. For general information about Aurora MySQL upgrades, see Upgrading Amazon Aurora MySQL DB clusters in the *Amazon Aurora User Guide*.

For troubleshooting information, see Troubleshooting upgrade issues with Aurora MySQL version 3.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Improvements

Aurora MySQL version 3.01.0 is generally compatible with community MySQL 8.0.23. This version includes the security fixes for Common Vulnerabilities and Exposures (CVE) issues as of community MySQL 8.0.23.

Aurora MySQL version 3.01.0 contains all the Aurora-specific bug fixes through Aurora MySQL version 2.10.0.

For details of new features in Aurora MySQL version 3, see Features from community MySQL 8.0 and New parallel query optimizations in the *Amazon Aurora User Guide*.

Availability improvements:

- Fast insert isn't enabled in this Aurora MySQL version, due to an issue that can cause inconsistencies when running queries such as `INSERT INTO`, `SELECT`, and `FROM`. For more information on the fast insert optimization, see Amazon Aurora MySQL performance enhancements.

Database engine updates for Amazon Aurora MySQL version 2

The following are database engine updates for Amazon Aurora MySQL version 2.

- Aurora MySQL database engine updates 2024-03-19 (version 2.12.2, compatible with MySQL 5.7.44)
- Aurora MySQL database engine updates 2023-12-28 (version 2.12.1, compatible with MySQL 5.7.40)
- Aurora MySQL database engine updates 2023-10-25 (version 2.12.0.1, compatible with MySQL 5.7.40) Beta
- Aurora MySQL database engine updates 2023-07-25 (version 2.12.0, compatible with MySQL 5.7.40)
- Aurora MySQL database engine updates 2024-03-26 (version 2.11.5, compatible with MySQL 5.7.12) Default
- Aurora MySQL database engine updates 2023-10-17 (version 2.11.4, compatible with MySQL 5.7.12)
- Aurora MySQL database engine updates 2023-06-09 (version 2.11.3, compatible with MySQL 5.7.12)
- Aurora MySQL database engine updates 2023-03-24 (version 2.11.2, compatible with MySQL 5.7.12)
- Aurora MySQL database engine updates 2023-02-14 (version 2.11.1, compatible with MySQL 5.7.12)
- Aurora MySQL database engine updates 2022-10-25 (version 2.11.0, compatible with MySQL 5.7.12) This version isn't available for new creations.
- Aurora MySQL database engine updates 2022-11-01 (version 2.10.3) (Deprecated)
- Aurora MySQL database engine updates 2022-01-26 (version 2.10.2) (Deprecated)
- Aurora MySQL database engine updates 2021-10-21 (version 2.10.1) (Deprecated)
- Aurora MySQL database engine updates 2021-05-25 (version 2.10.0) (Deprecated)
- Aurora MySQL database engine updates 2021-11-12 (version 2.09.3) (Deprecated)
- Aurora MySQL database engine updates 2021-02-26 (version 2.09.2) (Deprecated)

- Aurora MySQL database engine updates 2020-12-11 (version 2.09.1) (Deprecated)
- Aurora MySQL database engine updates 2020-09-17 (version 2.09.0) (Deprecated)
- Aurora MySQL database engine updates 2022-01-06 (version 2.08.4) (Deprecated)
- Aurora MySQL database engine updates 2020-11-12 (version 2.08.3) (Deprecated)
- Aurora MySQL database engine updates 2020-08-28 (version 2.08.2) (Deprecated)
- Aurora MySQL database engine updates 2020-06-18 (version 2.08.1) (Deprecated)
- Aurora MySQL database engine updates 2020-06-02 (version 2.08.0) (Deprecated)
- Aurora MySQL database engine update 2023-08-15 (version 2.07.10, compatible with MySQL 5.7.12)
- Aurora MySQL database engine update 2023-05-04 (version 2.07.9, compatible with MySQL 5.7.12)
- Aurora MySQL database engine updates 2022-06-16 (version 2.07.8) (Deprecated)
- Aurora MySQL database engine updates 2021-11-24 (version 2.07.7) (Deprecated)
- Aurora MySQL database engine updates 2021-09-02 (version 2.07.6) (Deprecated)
- Aurora MySQL database engine updates 2021-07-06 (version 2.07.5) (Deprecated)
- Aurora MySQL database engine updates 2021-03-04 (version 2.07.4) (Deprecated)
- Aurora MySQL database engine updates 2020-11-10 (version 2.07.3) (Deprecated)
- Aurora MySQL database engine updates 2020-04-17 (version 2.07.2) (Deprecated)
- Aurora MySQL database engine updates 2019-12-23 (version 2.07.1) (Deprecated)
- Aurora MySQL database engine updates 2019-11-25 (version 2.07.0) (Deprecated)
- Aurora MySQL database engine updates 2019-11-22 (version 2.06.0) (Deprecated)
- Aurora MySQL database engine updates 2019-11-11 (version 2.05.0) (Deprecated)
- Aurora MySQL database engine updates 2020-08-14 (version 2.04.9) (Deprecated)
- Aurora MySQL database engine updates 2019-11-20 (version 2.04.8) (Deprecated)
- Aurora MySQL database engine updates 2019-11-14 (version 2.04.7) (Deprecated)
- Aurora MySQL database engine updates 2019-09-19 (version 2.04.6) (Deprecated)
- Aurora MySQL database engine updates 2019-07-08 (version 2.04.5) (Deprecated)
- Aurora MySQL database engine updates 2019-05-29 (version 2.04.4) (Deprecated)
- Aurora MySQL database engine updates 2019-05-09 (version 2.04.3) (Deprecated)
- Aurora MySQL database engine updates 2019-05-02 (version 2.04.2) (Deprecated)

- Aurora MySQL database engine updates 2019-03-25 (version 2.04.1) (Deprecated)
- Aurora MySQL database engine updates 2019-03-25 (version 2.04.0) (Deprecated)
- Aurora MySQL database engine updates 2019-02-07 (version 2.03.4) (Deprecated)
- Aurora MySQL database engine updates 2019-01-18 (version 2.03.3) (Deprecated)
- Aurora MySQL database engine updates 2019-01-09 (version 2.03.2) (Deprecated)
- Aurora MySQL database engine updates 2018-10-24 (version 2.03.1) (Deprecated)
- Aurora MySQL database engine updates 2018-10-11 (version 2.03) (Deprecated)
- Aurora MySQL database engine updates 2018-10-08 (version 2.02.5) (Deprecated)
- Aurora MySQL database engine updates 2018-09-21 (version 2.02.4) (Deprecated)
- Aurora MySQL database engine updates 2018-08-23 (version 2.02.3) (Deprecated)
- Aurora MySQL database engine updates 2018-06-04 (version 2.02.2) (Deprecated)
- Aurora MySQL database engine updates 2018-05-03 (version 2.02) (Deprecated)
- Aurora MySQL database engine updates 2018-03-13 (version 2.01.1) (Deprecated)
- Aurora MySQL database engine updates 2018-02-06 (version 2.01) (Deprecated)

Aurora MySQL database engine updates 2024-03-19 (version 2.12.2, compatible with MySQL 5.7.44)

Version: 2.12.2

Aurora MySQL 2.12.2 is generally available. Aurora MySQL 2.12 versions are compatible up to MySQL 5.7.44. For more information on community changes, see Changes in MySQL 5.7.44 (2022-10-11, General Availability).

Currently supported Aurora MySQL releases are 2.07.9, 2.07.10, 2.11.*, 2.12.*, 3.03.*, 3.04.*, 3.05.* and 3.06.*.

You can upgrade an existing Aurora MySQL 2.* database cluster to Aurora MySQL 2.12.2. You can also restore a snapshot from any currently supported Aurora MySQL release into Aurora MySQL 2.12.2.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

> **Note**
>
> For information on how to upgrade your Aurora MySQL database cluster, see Upgrading the minor version or patch level of an Aurora MySQL DB cluster in the *Amazon Aurora User Guide*.

Improvements

Fixed security issues and CVEs listed below:

This release includes all community CVE fixes up to and including MySQL 5.7.44. The following CVE fixes are included:

- CVE-2024-20963
- CVE-2023-39975
- CVE-2023-38545

Security issues:

- Added a fix that ensures binary log replicas default to using SSL/TLS if the source supports encrypted connections, irrespective of the `MASTER_SSL` setting.

Availability improvements:

- Fixed an issue that can prevent a read replica instance from launching successfully if there is a high workload on the writer instance.
- Fixed an issue which can cause an Aurora MySQL database writer instance to failover due to a defect in the component that communicates with Aurora storage. The defect occurs as a result of a breakdown in the communication between the database instance and the underlying storage following a software update of the Aurora storage instance.
- Fixed an issue which, in rare conditions, can cause the reader instances to restart.
- Fixed an issue in which a privileged user can modify the resource limits associated with the user, rdsadmin. When set incorrectly, these resource limits can impede the RDS monitoring agent's ability to monitor the health of the database instance leading to database unavailability.

Upgrades and migrations:

- Fixed an issue that occurred when attempting to start the binary log replication for Aurora MySQL clusters that had migrated from Amazon RDS MySQL 5.7 and that contained unsupported stored procedures.
- Disabled the database event scheduler during a major version upgrade to Aurora MySQL version 3. This update helps avoid any changes to the database by the event execution while the major version upgrade is in progress.

Integration of MySQL Community Edition bug fixes

This release includes all community bug fixes up to and including 5.7.44. For more information, see MySQL bugs fixed by Aurora MySQL 2.x database engine updates.

Features not supported in Aurora MySQL version 2

The following features are currently not supported in Aurora MySQL version 2 (compatible with MySQL 5.7).

- Scan batching

MySQL 5.7 compatibility

This Aurora MySQL version is wire-compatible with MySQL 5.7 and includes features such as JSON support, spatial indexes, and generated columns. Aurora MySQL uses a native implementation of spatial indexing using z-order curves to deliver >20x better write performance and >10x better read performance than MySQL 5.7 for spatial datasets.

This Aurora MySQL version does not currently support the following MySQL 5.7 features:

- The `CREATE TABLESPACE` SQL statement
- Group replication plugin
- Increased page size
- InnoDB buffer pool loading at startup
- InnoDB full-text parser plugin
- Multisource replication

- Online buffer pool resizing
- Password validation plugin
- Query rewrite plugins
- Replication filtering
- X Protocol

Aurora MySQL database engine updates 2023-12-28 (version 2.12.1, compatible with MySQL 5.7.40)

Version: 2.12.1

Aurora MySQL 2.12.1 is generally available. Aurora MySQL 2.12 versions are compatible up to MySQL 5.7.40. For more information on community changes, see Changes in MySQL 5.7.40 (2022-10-11, General Availability).

Currently supported Aurora MySQL releases are 2.07.*, 2.11.*, 2.12.*, 3.01.*, 3.02.*, 3.03.*, 3.04.*, and 3.05.*.

You can upgrade an existing Aurora MySQL 2.* database cluster to Aurora MySQL 2.12.1. You can also restore a snapshot from any currently supported Aurora MySQL release into Aurora MySQL 2.12.1.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

> **Note**
>
> For information on how to upgrade your Aurora MySQL database cluster, see Upgrading the minor version or patch level of an Aurora MySQL DB cluster in the *Amazon Aurora User Guide*.

Improvements

Fixed security issues and CVEs listed below:

This release includes all community CVEs fixes up to and including MySQL 5.7.44.

Fixes and other enhancements to fine-tune handling in a managed environment. Additional CVE fixes are below:

- CVE-2023-38546
- CVE-2023-38545
- CVE-2023-22053
- CVE-2023-22028
- CVE-2023-22026
- CVE-2023-22015
- CVE-2022-24407
- CVE-2020-11105
- CVE-2020-11104
- Fixed processing of single character tokens by a Full-Text Search (FTS) parser plugin (Bug #35432973)
- Fixed an issue where events that were reported while processing the audit log rotations might not be written to the audit log

New features:

- Added support for multi-threaded binary log (binlog) replication, where the SQL thread on the binlog replica would apply binary log events in parallel when possible. Learn more about the configuration options to help fine-tune your multithreaded replication in the Aurora User Guide.

Availability improvements:

- Fixed an issue where Aurora MySQL database instances using parallel query may experience a database restart when running a high number of concurrent parallel queries.
- Fixed an issue with lock contention caused by an audit logging thread that can lead to high CPU utilization and client application timeouts.
- Fixed an issue which can cause a database instance restart when attempting to read a database page that belongs to a dropped table.
- Fixed an issue which can cause the reader instance to restart when the writer instance grows the database volume to a multiple of 160GB.

- Fixed an issue in the lock manager that could cause a restart or failover, when handling two-phase commits with the isolation level set to `READ_COMMITED` or `READ_UNCOMMITED` and either XA transactions are used or binary log (binlog) is enabled.
- Fixed an issue which can cause database cluster unavailability if the writer instance restarts while the database is creating or dropping triggers on internal system tables.
- Fixed an issue that may cause the database instance to restart when the number of database connections approaches the value set by the `max_connections` parameter.
- Fixed an issue which can cause an Aurora reader instance to restart when executing Data Manipulation Language (DML) queries on a table containing a full-text index.
- Fast insert isn't enabled in this Aurora MySQL version, due to an issue that can cause inconsistencies when running queries such as `INSERT INTO`, `SELECT`, and `FROM`. For more information on the fast insert optimization, see Amazon Aurora MySQL performance enhancements.

General improvements:

- Fixed an issue which can cause a parallel query to fail due to transient network issues while reading data from the Aurora cluster volume.
- Fixed an issue related to audit log file management which can cause log files to be inaccessible for download or rotation, and in some cases increase CPU utilization.
- Fixed an issue where small read replica instances may experience increased replication lag after upgrading from versions below 2.11.*
- Fixed an issue that can cause excessive log messages when consulting the `procs_priv` grant table for verification of requests that involve stored routines.
- Fixed a memory management issue which can cause the database instance to use excessive memory while executing queries using the hash join optimization.
- Fixed an issue which can produce an incorrect value of the variable `Threads_running` in the `information_schema` and `performance_schema` global status tables when using write forwarding.
- Fixed an issue that caused a restart of the database when executing `SELECT` statements with partitioned tables (created in a version of MySQL supporting the old `ha_partition` partition handler) and parallel query is chosen by the query planner.
- Fixed an issue which can prevent new client connections from being established to the database when write forwarding is enabled.

- Reduced binary log (binlog) replication lag when an Aurora MySQL binlog replica is executing `QUERY` events written to the source's binlog file without a default database defined by the `USE` command.
- Fixed an issue which can cause the `CommitLatency` CloudWatch metric to be incorrectly reported when the `innodb_flush_log_at_trx_commit` parameter is not set to 1.
- Fixed an issue which can cause database connections to be closed before being established. This issue is more likely to affect database instances which open and close connections at a high rate.
- Fixed an issue which can cause a database restart when connected binary log (binlog) consumers are using duplicate binlog replication server IDs.

Integration of MySQL Community Edition bug fixes

This release includes all community bug fixes up to and including 5.7.40 in addition to the below. For more information, see MySQL bugs fixed by Aurora MySQL 2.x database engine updates.

- Fixed an issue which can cause existing and new remote connections to stall when run concurrently with `SHOW PROCESSLIST` statement (Community Bug #34857411)
- Replication: Some binary log events were not always handled correctly (Bug #34617506)

Features not supported in Aurora MySQL version 2

The following features are currently not supported in Aurora MySQL version 2 (compatible with MySQL 5.7).

- Scan batching

MySQL 5.7 compatibility

This Aurora MySQL version is wire-compatible with MySQL 5.7 and includes features such as JSON support, spatial indexes, and generated columns. Aurora MySQL uses a native implementation of spatial indexing using z-order curves to deliver >20x better write performance and >10x better read performance than MySQL 5.7 for spatial datasets.

This Aurora MySQL version does not currently support the following MySQL 5.7 features:

- The `CREATE TABLESPACE` SQL statement

- Group replication plugin
- Increased page size
- InnoDB buffer pool loading at startup
- InnoDB full-text parser plugin
- Multisource replication
- Online buffer pool resizing
- Password validation plugin
- Query rewrite plugins
- Replication filtering
- X Protocol

Aurora MySQL database engine updates 2023-10-25 (version 2.12.0.1, compatible with MySQL 5.7.40) Beta

Version: 2.12.0.1

Aurora MySQL 2.12.0.1 is generally available in the following regions: US East (N. Virginia), US East (Ohio), US West (N. California), US West (Oregon), AWS GovCloud (US-East), and AWS GovCloud (US-West). This is an early, security fix–only release. These fixes will be deployed more broadly across all Regions with the next patch release, 2.12.1. Aurora MySQL 2.12 versions are compatible with MySQL 5.7.40.

Currently supported Aurora MySQL releases are 2.07.*, 2.11.*, 2.12.*, 3.01.*, 3.02.*, 3.03.*, and 3.04.*.

You can upgrade an existing Aurora MySQL 2.* database cluster to Aurora MySQL 2.12.0.1. You can also restore a snapshot from any currently supported Aurora MySQL release into Aurora MySQL 2.12.0.1.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Improvements

Fixed security issues and CVEs listed below:

This release includes all community CVEs fixes up to and including MySQL 5.7.40.

- CVE-2023-38545

Availability improvements:

- Fast insert isn't enabled in this Aurora MySQL version, due to an issue that can cause inconsistencies when running queries such as `INSERT INTO`, `SELECT`, and `FROM`. For more information on the fast insert optimization, see Amazon Aurora MySQL performance enhancements.

Aurora MySQL database engine updates 2023-07-25 (version 2.12.0, compatible with MySQL 5.7.40)

Version: 2.12.0

Aurora MySQL 2.12.0 is generally available. Aurora MySQL 2.12 versions are compatible up to MySQL 5.7.40. For more information on community changes, see Changes in MySQL 5.7.40 (2022-10-11, General Availability).

Currently supported Aurora MySQL releases are 2.07.*, 2.11.*, 2.12.*, 3.01.*, 3.02.* and 3.03.*.

You can upgrade an existing Aurora MySQL 2.* database cluster to Aurora MySQL 2.12.0. You can also restore a snapshot from any currently supported Aurora MySQL release into Aurora MySQL 2.12.0.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

> **Note**
>
> For information on how to upgrade your Aurora MySQL database cluster, see Upgrading the minor version or patch level of an Aurora MySQL DB cluster in the *Amazon Aurora User Guide*.

Improvements

Fixed security issues and CVEs listed below:

This release includes all community CVEs fixes up to and including MySQL 5.7.40.

- Default SSL ciphers used by Aurora MySQL have been updated to exclude the less secure DES-CBC3-SHA values from the SSL_CIPHER database parameter. If you encounter SSL connection issues due to the removal of the DES-CBC3-SHA cipher, please use an applicable secure cipher from the following list, Configuring cipher suites for connections to Aurora MySQL DB clusters. More information on MySQL client Connection Cipher Configuration can be found in the MySQL documentation.
- CVE-2023-21963
- CVE-2023-21912
- CVE-2023-21840
- CVE-2023-0215
- CVE-2022-43551
- CVE-2022-37434
- CVE-2022-32221
- CVE-2021-36222
- CVE-2021-22926
- CVE-2021-2169

Availability improvements:

- Fixed an issue in the database activity streams event encryption which may cause database restarts
- Fixed two issues which can cause a database restart to fail if it occurred while executing a Data Definition Language (DDL) query
- Fixed an issue where connection surges can cause increased query latency or a database instance restart
- Fixed an issue which, in rare cases, can cause an Aurora replica to restart during simultaneous execution of large update operations or Data Definition Language (DDL) workloads on the writer instance and read operations on the same set of tables on the Aurora replica

- Fixed an issue where connection surges could cause the connection establishment process to take longer to complete or to fail with timeout errors
- Fixed an issue where the Advanced Auditing log rotation may reduce the freeable memory, which could lead to the database instance restarting
- Fixed an issue which can cause an Aurora MySQL reader instance to restart while executing a query which utilizes an Aurora parallel query execution plan
- Fixed an issue that can cause the writer instance to restart while executing the `OPTIMIZE TABLE` query on a table with a Full Text Search (FTS) index
- Fixed an issue which can cause the writer instance in an Aurora global database primary AWS Region to restart when a `SELECT FOR UPDATE` query is executed using global write forwarding from an Aurora global database secondary Region
- Fixed an issue which can cause an Aurora global database secondary AWS Region reader instance using global write forwarding to restart when a forwarded implicit commit statement encounters an error
- Fast insert isn't enabled in this Aurora MySQL version, due to an issue that can cause inconsistencies when running queries such as `INSERT INTO`, `SELECT`, and `FROM`. For more information on the fast insert optimization, see Amazon Aurora MySQL performance enhancements.

General improvements:

- Introduced file management performance optimizations on binlog replicas to help reduce contention when writing to relay log files
- Fixed an issue that can cause the `buffer_pool_read_requests` counter to be reported incorrectly in the `information_schema` metrics
- Fixed an issue that can cause the local storage to fill up when performing `LOAD FROM S3` or `SELECT INTO S3` operations. The issue can also lead to higher CPU utilization, database restarts due to low memory, and increased latency for these queries.
- Fixed an issue where DB instances using binary log replication may experience an increase in CPU utilization and connection failures when multiple binary log replication consumers are attached
- Fixed an issue where the SSL server status variables weren't being populated
- Fixed an issue where the Data Manipulation Language (DML) statements executing duplicate writes could lead to excessive error logging and increased query latencies
- Upgraded the time zone definitions to the IANA 2023c version

- Added support for enabling and disabling session-level binary logging. See Stored Procedures - Replicating in the Amazon Aurora User Guide
- Added support for setting the session-level binary log format. See Stored Procedures - Replicating in the Amazon Aurora User Guide
- Fixed an issue where setting the `aurora_disable_hash_join` parameter to `1` or `ON` might not prevent the optimizer from using a hash join
- Fixed an issue involving index scans where an inaccurate result might be returned when executing a `SELECT` query with the `GROUP BY` clause and the `aurora_parallel_query` parameter turned `ON`
- Fixed an issue which, in rare cases, can cause an Amazon Aurora reader instance to restart when accessing a table which has large update or Data Definition Language (DDL) operations running concurrently on the writer instance
- Fixed an issue that can cause the `buffer_pool_read_requests` counter to be reported incorrectly in the `information_schema` metrics
- Fixed an issue that can cause a binlog replica to restart if the system variable server uuid of the source is missing or has an invalid value
- Fixed an issue to prevent InnoDB statistics from getting stale, which can sometimes generate a sub-optimal query execution plan that may lead to an increase in the query execution time
- Fixed an issue wherein the `AuroraGlobalDBRPOLag` CloudWatch metrics always displayed zero regardless of the user workload

Upgrades and migrations:

- To perform a minor version upgrade for an Aurora global database from Aurora MySQL version 2.07 or 2.11 to Aurora MySQL version 2.12 or higher, refer to Upgrading Aurora MySQL by modifying the engine version.

Integration of MySQL Community Edition bug fixes

This release includes all community bug fixes up to and including 5.7.40 in addition to the below. For more information, see MySQL bugs fixed by Aurora MySQL 2.x database engine updates.

- Fixed an issue which can cause higher CPU utilization due to background TLS certificate rotation (Community Bug Fix #34284186)

Features not supported in Aurora MySQL version 2

The following features are currently not supported in Aurora MySQL version 2 (compatible with MySQL 5.7).

- Scan batching.

MySQL 5.7 compatibility

This Aurora MySQL version is wire-compatible with MySQL 5.7 and includes features such as JSON support, spatial indexes, and generated columns. Aurora MySQL uses a native implementation of spatial indexing using z-order curves to deliver >20x better write performance and >10x better read performance than MySQL 5.7 for spatial datasets.

This Aurora MySQL version does not currently support the following MySQL 5.7 features:

- The `CREATE TABLESPACE` SQL statement
- Group replication plugin
- Increased page size
- InnoDB buffer pool loading at startup
- InnoDB full-text parser plugin
- Multisource replication
- Online buffer pool resizing
- Password validation plugin
- Query rewrite plugins
- Replication filtering
- X Protocol

Aurora MySQL database engine updates 2024-03-26 (version 2.11.5, compatible with MySQL 5.7.12) Default

Version: 2.11.5

Aurora MySQL 2.11.5 is generally available. Aurora MySQL 2.11 versions are compatible with MySQL 5.7.12. For more information on community changes, see Changes in MySQL 5.7.12 (2016-04-11, General Availability).

The currently supported Aurora MySQL releases are 2.07.9, 2.07.10, 2.11.*, 2.12.*, 3.01.*, 3.02.*, 3.03.*, 3.04.*, 3.05.*, and 3.06.*.

You can upgrade an existing Aurora MySQL 2.* database cluster to Aurora MySQL 2.11.5. You can also restore a snapshot from any currently supported lower Aurora MySQL version 2 release into Aurora MySQL 2.11.5.

If you upgrade an Aurora MySQL global database to version 2.11.*, you must upgrade your primary and secondary DB clusters to the exact same version, including the patch level. For more information on upgrading the minor version of an Aurora global database, see Minor version upgrades.

Immediately after an in-place engine version upgrade to Aurora MySQL 2.11.* is performed, an operating system upgrade is applied automatically to all affected instances on the db.r4, db.r5, db.t2, and db.t3 DB instance classes, if the instances are running an old operating system version. In a Multi-AZ DB cluster, all of the reader instances apply the operating system upgrade first. When the operating system upgrade on the first reader instance is finished, a failover occurs and the previous writer instance is upgraded.

Note

The operating system upgrade isn't applied automatically to Aurora global databases during major version upgrades.

Note

For information on how to upgrade your Aurora MySQL database cluster, see Upgrading the minor version or patch level of an Aurora MySQL DB cluster in the *Amazon Aurora User Guide*.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Improvements

Fixed security issues and CVEs:

The following CVE fixes are included in this release:

- CVE-2020-11104
- CVE-2020-11105
- CVE-2023-22015
- CVE-2023-22026
- CVE-2023-22028
- CVE-2023-22084
- CVE-2023-38545
- CVE-2023-38546
- CVE-2024-20963

Availability improvements:

- Fixed an issue where an Aurora MySQL writer DB instance can fail over due to a defect in the component that communicates with Aurora storage. The defect occurs as a result of a breakdown in the communication between the DB instance and underlying storage following a software update.
- Fixed an issue that, in rare conditions, can cause the reader DB instances to restart.
- Fixed an issue with lock contention caused by an audit logging thread that can lead to high CPU utilization and client application timeouts.

General improvements:

- Fixed an issue that can cause a parallel query to fail due to transient network issues while reading data from the Aurora DB cluster volume.
- Fixed an issue related to audit log file management that can cause log files to be inaccessible for download or rotation, and in some cases increase CPU usage.
- Fixed an issue that can produce an incorrect value of the `Threads_running` variable in the `information_schema` and `performance_schema` global status tables when using write forwarding.

Upgrades and migrations:

- Fixed an issue that prevented initiation of binary log replication on Aurora MySQL DB clusters migrated from RDS for MySQL 5.7.
- Disabled the database event scheduler during major version upgrades to Aurora MySQL version 3. This helps avoid any changes to the database by event execution while the major version upgrade is in progress.

Features not supported in Aurora MySQL version 2

The following features are currently not supported in Aurora MySQL version 2 (compatible with MySQL 5.7).

- Scan batching. For more information, see Aurora MySQL database engine updates 2017-12-11 (version 1.16) (Deprecated).

MySQL 5.7 compatibility

This Aurora MySQL version is wire-compatible with MySQL 5.7 and includes features such as JSON support, spatial indexes, and generated columns. Aurora MySQL uses a native implementation of spatial indexing using z-order curves to deliver >20x better write performance and >10x better read performance than MySQL 5.7 for spatial datasets.

This Aurora MySQL version does not currently support the following MySQL 5.7 features:

- Group replication plugin
- Increased page size
- InnoDB buffer pool loading at startup
- InnoDB full-text parser plugin
- Multisource replication
- Online buffer pool resizing
- Password validation plugin
- Query rewrite plugins
- Replication filtering
- The `CREATE TABLESPACE` SQL statement

Aurora MySQL database engine updates 2023-10-17 (version 2.11.4, compatible with MySQL 5.7.12)

Version: 2.11.4

Aurora MySQL 2.11.4 is generally available. Aurora MySQL 2.11 versions are compatible with MySQL 5.7.12. For more information on community changes, see Changes in MySQL 5.7.12 (2016-04-11, General Availability).

The currently supported Aurora MySQL releases are 2.07.9, 2.07.10, 2.11.*, 2.12.*, 3.01.*, 3.02.*, 3.03.*, and 3.04.*.

You can upgrade an existing Aurora MySQL 2.* database cluster to Aurora MySQL 2.11.4. You can also restore a snapshot from any currently supported Aurora MySQL release into Aurora MySQL 2.11.4.

If you upgrade an Aurora MySQL global database to version 2.11.*, you must upgrade your primary and secondary DB clusters to the exact same version, including the patch level. For more information on upgrading the minor version of an Aurora global database, see Minor version upgrades.

Immediately after an in-place engine version upgrade to Aurora MySQL 2.11.* is performed, an operating system upgrade is applied automatically to all affected instances on the db.r4, db.r5, db.t2, and db.t3 DB instance classes, if the instances are running an old operating system version. In a Multi-AZ DB cluster, all of the reader instances apply the operating system upgrade first. When the operating system upgrade on the first reader instance is finished, a failover occurs and the previous writer instance is upgraded.

> **Note**
>
> The operating system upgrade isn't applied automatically to Aurora global databases during major version upgrades.

> **Note**
>
> For information on how to upgrade your Aurora MySQL database cluster, see Upgrading the minor version or patch level of an Aurora MySQL DB cluster in the *Amazon Aurora User Guide*.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Improvements

Fixed security issues and CVEs listed below:

- Fixed an issue where the events that were reported while processing audit log rotations might not be written to the audit log.
- CVE-2022-24407

Availability improvements:

- Fixed an issue where Aurora MySQL database instances using parallel query may experience a database restart when running a high number of concurrent parallel queries.
- Fixed an issue which can cause a database instance to restart while executing I/O intensive read workloads.
- Fixed an issue which can cause a database instance restart when attempting to read a database page that belongs to a dropped table.
- Fixed an issue which can cause the reader instance to restart when the writer instance grows the database volume to a multiple of 160GB.
- Fixed an issue which can cause database cluster unavailability if the writer instance restarts while the database is creating or dropping triggers on internal system tables.
- Fixed an issue which can cause a reader instance to restart when executing Data Manipulation Language (DML) queries on a table containing a full-text index.
- Fixed an issue which can cause a reader instance to restart while executing a query which utilizes an Aurora parallel query execution plan.

- Fixed an issue that can cause the writer instance to restart while executing the `OPTIMIZE TABLE` query on a table with a Full Text Search (FTS) index.
- Fast insert isn't enabled in this Aurora MySQL version, due to an issue that can cause inconsistencies when running queries such as `INSERT INTO`, `SELECT`, and `FROM`. For more information on the fast insert optimization, see Amazon Aurora MySQL performance enhancements.

General improvements:

- Fixed an issue where small read replica instances may experience increased replication lag after upgrading from versions below 2.11.*.
- Fixed an issue that can cause excessive log messages when consulting the procs_priv grant table for verification of requests that involve stored routines.
- Fixed a memory management issue which can cause the database instance to use excessive memory while executing queries using the hash join optimization.
- Fixed an issue that caused a restart of the database when executing `SELECT` statements with partitioned tables (created in a version of MySQL supporting the old `ha_partition` partition handler) and parallel query is chosen by the query planner.
- Fixed an issue which can prevent new client connections from being established to the database when write forwarding is enabled.
- Reduced binary log (binlog) replication lag when an Aurora MySQL binlog replica is executing `QUERY` events written to the source's binlog file without a default database defined by the `USE` command.
- Fixed an issue involving index scans where an inaccurate result might be returned when executing a `SELECT` query with the `GROUP BY` clause and the `aurora_parallel_query` parameter turned `ON`.
- Added support for enabling and disabling session-level binary logging. See Stored Procedures - Replicating in the *Amazon Aurora User Guide*.
- Fixed an issue that can cause a binlog replica to restart if the system variable server_uuid of the source is missing or has an invalid value.
- Added support for setting session-level binary log format. See Stored Procedures - Replicating in the *Amazon Aurora User Guide*.
- Fixed an issue which can cause the `CommitLatency` CloudWatch metric to be incorrectly reported when the `innodb_flush_log_at_trx_commit` parameter is not set to 1.

- Fixed an issue to prevent InnoDB statistics from getting stale, which can sometimes generate a sub-optimal query execution plan that may lead to an increase in query execution time.
- Fixed an issue which can cause a database restart when connected binary log (binlog) consumers are using duplicate binlog replication server IDs.

Integration of MySQL Community Edition bug fixes

This release includes all community bug fixes up to and including 5.7.12, in addition to the below. For more information, see MySQL bugs fixed by Aurora MySQL 2.x database engine updates.

- Replication: Some binary log events were not always handled correctly. (Bug #34617506)
- Fixed an issue which can cause higher CPU utilization due to background TLS certificate rotation (Community Bug Fix #34284186).
- In prepared statements, some types of subqueries could cause a server exit. (Bug #33100586)

Features not supported in Aurora MySQL version 2

The following features are currently not supported in Aurora MySQL version 2 (compatible with MySQL 5.7).

- Scan batching. For more information, see Aurora MySQL database engine updates 2017-12-11 (version 1.16) (Deprecated).

MySQL 5.7 compatibility

This Aurora MySQL version is wire-compatible with MySQL 5.7 and includes features such as JSON support, spatial indexes, and generated columns. Aurora MySQL uses a native implementation of spatial indexing using z-order curves to deliver >20x better write performance and >10x better read performance than MySQL 5.7 for spatial datasets.

This Aurora MySQL version does not currently support the following MySQL 5.7 features:

- Group replication plugin
- Increased page size
- InnoDB buffer pool loading at startup
- InnoDB full-text parser plugin

- Multisource replication
- Online buffer pool resizing
- Password validation plugin
- Query rewrite plugins
- Replication filtering
- The CREATE TABLESPACE SQL statement

Aurora MySQL database engine updates 2023-06-09 (version 2.11.3, compatible with MySQL 5.7.12)

Version: 2.11.3

Aurora MySQL 2.11.3 is generally available. Aurora MySQL 2.11 versions are compatible with MySQL 5.7.12. For more information on community changes, see Changes in MySQL 5.7.12 (2016-04-11, General Availability).

The currently supported Aurora MySQL releases are 2.07.*, 2.11.*, 3.01.*, 3.02.* and 3.03.*.

You can upgrade an existing Aurora MySQL 2.* database cluster to Aurora MySQL 2.11.3. You can also restore a snapshot from any currently supported Aurora MySQL release into Aurora MySQL 2.11.3.

If you upgrade an Aurora MySQL global database to version 2.11.*, you must upgrade your primary and secondary DB clusters to the exact same version, including the patch level. For more information on upgrading the minor version of an Aurora global database, see Minor version upgrades.

Immediately after an in-place engine version upgrade to Aurora MySQL 2.11.* is performed, an operating system upgrade is applied automatically to all affected instances on the db.r4, db.r5, db.t2, and db.t3 DB instance classes, if the instances are running an old operating system version. In a Multi-AZ DB cluster, all of the reader instances apply the operating system upgrade first. When the operating system upgrade on the first reader instance is finished, a failover occurs and the previous writer instance is upgraded.

> **Note**
>
> The operating system upgrade isn't applied automatically to Aurora global databases during major version upgrades.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

> **Note**
>
> For information on how to upgrade your Aurora MySQL database cluster, see Upgrading the minor version or patch level of an Aurora MySQL DB cluster in the *Amazon Aurora User Guide*.

Improvements

Fixed security issues and CVEs listed below:

- Updated the default SSL ciphers used by Aurora MySQL to exclude the less secure `DES-CBC3-SHA` values from the SSL_CIPHER database parameter. If you encounter SSL connection issues due to the removal of the `DES-CBC3-SHA` cipher, please use an applicable secure cipher from this list, ConfiguringCipherSuites. More information on the MySQL client Connection Cipher Configuration can be found in the MySQL documentation.
- CVE-2023-21963
- CVE-2023-21912
- CVE-2023-0215
- CVE-2022-43551
- CVE-2022-37434

Availability improvements:

- Fixed an issue in database activity streams (DAS) event encryption which may cause database restarts.

- Fixed two issues which can cause a database restart to fail if it occurred while executing a Data Definition Language (DDL) query.
- Fast insert isn't enabled in this Aurora MySQL version, due to an issue that can cause inconsistencies when running queries such as `INSERT INTO`, `SELECT`, and `FROM`. For more information on the fast insert optimization, see Amazon Aurora MySQL performance enhancements.

General improvements:

- Introduced file management performance optimizations on binlog replicas to help reduce contention when writing to relay log files.
- Fixed an issue where setting the `aurora_disable_hash_join` parameter to 1 or ON might not prevent the optimizer from using a hash join.
- Fixed an issue that can cause the `buffer_pool_read_requests` counter to be reported incorrectly in the `information_schema` metrics.
- Fixed an issue that can cause the local storage to fill up when performing `LOAD FROM S3` or `SELECT INTO S3` operations. The issue can also lead to higher CPU utilization, database restarts due to low memory, and increased latency for these queries.

Features not supported in Aurora MySQL version 2

The following features are currently not supported in Aurora MySQL version 2 (compatible with MySQL 5.7).

- Scan batching. For more information, see Aurora MySQL database engine updates 2017-12-11 (version 1.16) (Deprecated).

MySQL 5.7 compatibility

This Aurora MySQL version is wire-compatible with MySQL 5.7 and includes features such as JSON support, spatial indexes, and generated columns. Aurora MySQL uses a native implementation of spatial indexing using z-order curves to deliver >20x better write performance and >10x better read performance than MySQL 5.7 for spatial datasets.

This Aurora MySQL version does not currently support the following MySQL 5.7 features:

- Group replication plugin
- Increased page size
- InnoDB buffer pool loading at startup
- InnoDB full-text parser plugin
- Multisource replication
- Online buffer pool resizing
- Password validation plugin
- Query rewrite plugins
- Replication filtering
- The `CREATE TABLESPACE` SQL statement

Aurora MySQL database engine updates 2023-03-24 (version 2.11.2, compatible with MySQL 5.7.12)

Version: 2.11.2

Aurora MySQL 2.11.2 is generally available. Aurora MySQL 2.11 versions are compatible with MySQL 5.7.12. For more information on community changes, see Changes in MySQL 5.7.12 (2016-04-11, General Availability).

The currently supported Aurora MySQL releases are 2.07.*, 2.11.*, 3.01.*, 3.02.* and 3.03.*.

You can upgrade an existing Aurora MySQL 2.* database cluster to Aurora MySQL 2.11.2. You can also restore a snapshot from any currently supported Aurora MySQL release into Aurora MySQL 2.11.2.

If you upgrade an Aurora MySQL global database to version 2.11.*, you must upgrade your primary and secondary DB clusters to the exact same version, including the patch level. For more information on upgrading the minor version of an Aurora global database, see Minor version upgrades.

Immediately after an in-place engine version upgrade to Aurora MySQL 2.11.* is performed, an operating system upgrade is applied automatically to all affected instances on the db.r4, db.r5, db.t2, and db.t3 DB instance classes, if the instances are running an old operating system version.

In a Multi-AZ DB cluster, all of the reader instances apply the operating system upgrade first. When the operating system upgrade on the first reader instance is finished, a failover occurs and the previous writer instance is upgraded.

> **Note**
>
> The operating system upgrade isn't applied automatically to Aurora global databases during major version upgrades.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

> **Note**
>
> For information on how to upgrade your Aurora MySQL database cluster, see Upgrading the minor version or patch level of an Aurora MySQL DB cluster in the *Amazon Aurora User Guide*.

Improvements

General improvements:

- Fixed an issue where DB instances using binary log replication may experience an increase in CPU utilization and connection failures when multiple binary log replication consumers are attached.
- Fixed an issue that can cause a reader instance in a global database secondary Region to become out of sync after upgrading to Aurora MySQL version 2.11 if the primary database writer is on Aurora MySQL version 2.10.

Availability improvements:

- Fast insert isn't enabled in this Aurora MySQL version, due to an issue that can cause inconsistencies when running queries such as `INSERT INTO`, `SELECT`, and `FROM`. For more information on the fast insert optimization, see Amazon Aurora MySQL performance enhancements.

Features not supported in Aurora MySQL version 2

The following features are currently not supported in Aurora MySQL version 2 (compatible with MySQL 5.7).

- Scan batching. For more information, see Aurora MySQL database engine updates 2017-12-11 (version 1.16) (Deprecated).

MySQL 5.7 compatibility

This Aurora MySQL version is wire-compatible with MySQL 5.7 and includes features such as JSON support, spatial indexes, and generated columns. Aurora MySQL uses a native implementation of spatial indexing using z-order curves to deliver >20x better write performance and >10x better read performance than MySQL 5.7 for spatial datasets.

This Aurora MySQL version does not currently support the following MySQL 5.7 features:

- Group replication plugin
- Increased page size
- InnoDB buffer pool loading at startup
- InnoDB full-text parser plugin
- Multisource replication
- Online buffer pool resizing
- Password validation plugin
- Query rewrite plugins
- Replication filtering
- The `CREATE TABLESPACE` SQL statement

Aurora MySQL database engine updates 2023-02-14 (version 2.11.1, compatible with MySQL 5.7.12)

Version: 2.11.1

Aurora MySQL 2.11.1 is generally available. Aurora MySQL 2.11 versions are compatible with MySQL 5.7.12. For more information on community changes, see Changes in MySQL 5.7.12 (2016-04-11, General Availability).

Currently supported Aurora MySQL releases are 1.19.5, 1.19.6, 1.22.*, 1.23.*, 2.07.*, 2.09.*, 2.10.*, 2.11.*, 3.01.* and 3.02.*.

You can upgrade an existing Aurora MySQL 2.* database cluster to Aurora MySQL 2.11.1. For clusters running Aurora MySQL version 1, you can upgrade an existing Aurora MySQL 1.23 or higher cluster directly to 2.11.1. You can also restore a snapshot from any currently supported Aurora MySQL release into Aurora MySQL 2.11.1.

If you upgrade an Aurora MySQL global database to version 2.11.* and you have write forwarding turned on, you must upgrade your primary and secondary DB clusters to the exact same version, including the patch level, to continue using write forwarding. For more information on upgrading the minor version of an Aurora global database, see Minor version upgrades.

Immediately after an in-place engine version upgrade to Aurora MySQL 2.11.* is performed, an operating system upgrade is applied automatically to all affected instances on the db.r4, db.r5, db.t2, and db.t3 DB instance classes, if the instances are running an old operating system version. In a Multi-AZ DB cluster, all of the reader instances apply the operating system upgrade first. When the operating system upgrade on the first reader instance is finished, a failover occurs and the previous writer instance is upgraded.

Note

The operating system upgrade isn't applied automatically to Aurora global databases during major version upgrades.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Note

For information on how to upgrade your Aurora MySQL database cluster, see Upgrading the minor version or patch level of an Aurora MySQL DB cluster in the *Amazon Aurora User Guide*.

Improvements

Fixed security issues and CVEs listed below:

Fixes and other enhancements to fine-tune handling in a managed environment. Additional CVE fixes below:

- CVE-2022-32221
- CVE-2021-36222
- CVE-2021-22926
- CVE-2021-2169

Availability improvements:

- Fixed an issue where connection surges can cause increased query latency or a database instance restart.
- Fixed an issue which, in rare cases, can cause an Aurora replica to restart during simultaneous execution of large update operations or Data Definition Language (DDL) workloads on the writer instance and read operations on the same set of tables on the Aurora replica.
- Fixed an issue where connection surges could cause the connection establishment process to take longer to complete or to fail with timeout errors.
- Fixed an issue where the Advanced Audit log rotation may reduce the freeable memory, which could lead to the database instance restarting.
- Fast insert isn't enabled in this Aurora MySQL version, due to an issue that can cause inconsistencies when running queries such as `INSERT INTO`, `SELECT`, and `FROM`. For more information on the fast insert optimization, see Amazon Aurora MySQL performance enhancements.

General improvements:

- Fixed an issue where the SSL server status variables) weren't being populated.
- Fixed an issue where the Data Manipulation Language (DML) statements executing duplicate writes could lead to excessive error logging and increased query latencies.

Comparison with Aurora MySQL version 1

The following Amazon Aurora MySQL features are supported in Aurora MySQL version 1 (compatible with MySQL 5.6), but these features are currently not supported in Aurora MySQL version 2 (compatible with MySQL 5.7).

- Hash joins. For more information, see Optimizing large Aurora MySQL join queries with hash joins in the *Amazon Aurora User Guide*.
- Native functions for synchronously invoking AWS Lambda functions. For more information, see Invoking a Lambda function with an Aurora MySQL native function in the *Amazon Aurora User Guide*.
- Scan batching. For more information, see Aurora MySQL database engine updates 2017-12-11 (version 1.16) (Deprecated).
- Migrating data from MySQL using an Amazon S3 bucket. For more information, see Migrating data from MySQL by using an Amazon S3 bucket in the *Amazon Aurora User Guide*.

MySQL 5.7 compatibility

This Aurora MySQL version is wire-compatible with MySQL 5.7 and includes features such as JSON support, spatial indexes, and generated columns. Aurora MySQL uses a native implementation of spatial indexing using z-order curves to deliver >20x better write performance and >10x better read performance than MySQL 5.7 for spatial datasets.

This Aurora MySQL version does not currently support the following MySQL 5.7 features:

- Group replication plugin
- Increased page size
- InnoDB buffer pool loading at startup
- InnoDB full-text parser plugin
- Multisource replication
- Online buffer pool resizing
- Password validation plugin
- Query rewrite plugins
- Replication filtering
- The `CREATE TABLESPACE` SQL statement

Aurora MySQL database engine updates 2022-10-25 (version 2.11.0, compatible with MySQL 5.7.12) This version isn't available for new creations.

Version: 2.11.0

Aurora MySQL 2.11.0 is generally available. Aurora MySQL 2.x versions are compatible with MySQL 5.7.12. For more information on community changes, see Changes in MySQL 5.7.12 (2016-04-11, General Availability).

Currently supported Aurora MySQL releases are 1.19.5, 1.19.6, 1.22.*, 1.23.*, 2.04.*, 2.07.*, 2.08.*, 2.09.*, 2.10.*, 2.11.*, 3.01.* and 3.02.*.

You can upgrade an existing Aurora MySQL 2.* database cluster to Aurora MySQL 2.11.0. For clusters running Aurora MySQL version 1, you can upgrade an existing Aurora MySQL 1.23 or higher cluster directly to 2.11.0. You can also restore a snapshot from any currently supported Aurora MySQL release into Aurora MySQL 2.11.0.

If you upgrade an Aurora MySQL global database to version 2.11.* and you have write forwarding turned on, you must upgrade your primary and secondary DB clusters to the exact same version, including the patch level, to continue using write forwarding. For more information on upgrading the minor version of an Aurora global database, see Minor version upgrades.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

> **Note**
>
> For information on how to upgrade your Aurora MySQL database cluster, see Upgrading the minor version or patch level of an Aurora MySQL DB cluster in the *Amazon Aurora User Guide*.

Improvements

Fixed security issues and CVEs listed below:

Fixes and other enhancements to fine-tune handling in a managed environment. Additional CVE fixes below:

- CVE-2022-21460
- CVE-2022-21451
- CVE-2022-21444
- CVE-2022-21417
- CVE-2022-21304
- CVE-2022-21303
- CVE-2022-21245
- CVE-2021-36222
- CVE-2021-28196
- CVE-2021-23841
- CVE-2021-22926
- CVE-2021-3449
- CVE-2021-2307
- CVE-2021-2226
- CVE-2021-2202
- CVE-2021-2194
- CVE-2021-2179
- CVE-2021-2178
- CVE-2021-2174
- CVE-2021-2171
- CVE-2021-2169
- CVE-2021-2166
- CVE-2021-2160
- CVE-2021-2154

New features:

- With the release of Aurora MySQL version 2.11, a new operating system upgrade is available. We recommend that you apply this pending OS update to all your Aurora MySQL database

instances after upgrading to version 2.11. For more information, see Working with operating system updates.

- A new dynamic configuration option, innodb_deadlock_detect, may be used to disable deadlock detection. On high concurrency systems, deadlock detection can cause a slowdown when numerous threads wait for the same lock. At times, it may be more efficient to disable deadlock detection and rely on the innodb_lock_wait_timeout setting for transaction rollback when a deadlock occurs. (Bug #23477773) More information on Innodb deadlock detection can be found in the MySQL documentation.
- The `UUID_TO_BIN`, `BIN_TO_UUID` and `IS_UUID` functions from MySQL 8.0 have been added. More information on using these functions can be found in MySQL Miscellaneous function.
- Added support for optimizer hints allowing the user to enable or disable Aurora MySQL parallel query on a per-table or per-query basis.
  - Working with parallel query for Amazon Aurora MySQL
  - Aurora MySQL hints
- Removed R3 instance type support.
- Added support for R6i instances.

Availability improvements:

- Fixed an issue which can prevent cross region logical replication in a database cluster due to incorrect binlog file and position written to the error logs. This issue may occur when the engine is restarted after running a DDL statement.
- Fixed an issue which, in rare conditions, can cause Aurora reader instances to restart when running Access-Control List (ACL) statements such as GRANT and FLUSH on the writer instance. This issue is more likely to affect reader instances with a large number of users and ACL operations (e.g., permission changes).
- Fixed an issue which, in rare conditions, can cause the writer instance to restart or failover when a transaction accesses a row being deleted by another transaction.
- Improved the Fulltext phrase search performance to significantly reduce the time taken to search phrases in a table with fulltext indexes.
- Fixed an issue where, after a writer instance restarts, it would get stuck in slow recovery and subsequently restart again. This issue occurs when there are a large number of uncommitted rows in the database at the time of the initial restart.

- Fixed an issue which, in rare cases, causes the database server to restart due to a long semaphore wait when the deadlock detector thread gets stuck.
- Fixed an issue which, in rare cases, can cause the database to restart due to a long semaphore wait when I/O threads become deadlocked.
- Fast insert isn't enabled in this Aurora MySQL version, due to an issue that can cause inconsistencies when running queries such as `INSERT INTO`, `SELECT`, and `FROM`. For more information on the fast insert optimization, see Amazon Aurora MySQL performance enhancements.

General improvements:

- Fixed an issue which can cause the database server to restart when all of the following conditions are true:
  - ALLOW_INVALID_DATES is disabled in SQL MODE.
  - The database server is processing an INSERT, UPDATE, DELETE or SELECT statement with an invalid value of DATETIME type such that the month is not between 1 and 12.
- Fixed an issue where the binary log retention period was not honored when log-bin was set to OFF, leading to higher than expected storage utilization. After this fix, the binary logs will be purged based on your retention period. More information on how to configure your binary log retention period can be found in the Aurora MySQL User Guide.
- Fixed an issue which can cause the freeable memory on the database instance to reduce when certain Data Control Language (DCL) SQL statements such as GRANT, FLUSH PRIVILEGES etc. are run on that instance. Frequent use of such statements can cause the freeable memory to keep reducing and may cause the database instance to restart because of out-of-memory issues. Use of such statements on the writer instance can also cause the freeable memory on the reader instances to reduce.
- Introduced a larger read buffer size for reads performed from the relay logs to minimize the number of read I/O operations, which reduces contention between the I/O and SQL threads.
- Fixed an issue that can cause the mysql.rds_rotate_slow_log stored procedure to fail with the error message "Table 'mysql.slow_log_backup' doesn't exist".
- Fixed an issue where excessive query cache invalidation causes higher than expected CPU usage and latencies on the read replica due to the read replica having to read the data from the disk instead of from the query cache.

- Fixed an issue which allowed users to run the INSTALL PLUGIN and UNINSTALL PLUGIN commands on a reader instance, which can cause deadlock on LOCK_plugin, LOCK_system_variables_hash, LOCK_global_system_variables. These statements can now only be executed on the writer instance in a database cluster.
- Fixed an issue where clusters may experience higher than expected commit latency when binary logging is enabled. This affects all transactions that generate large binlog events (over 500MB in size).
- Fixed an issue that can cause the trx_active_transactions metric in the INFORMATION_SCHEMA.INNODB_METRICS table to have an incorrect value.
- Fixed an issue which can stop logical replication due to the binlog file becoming inconsistent while executing a rollback to savepoint for a large transaction.
- Masked credential hashes in general-log, slow-query-log, and audit-log by default using a consistent mask secret. This is configurable via the aurora_mask_password_hashes_type parameter.
- Fixed an issue where the Zero-Downtime-Restart (ZDR) duration is incorrectly reported in the customer observed events.
- Fixed an issue which can cause calls to mysql_rds_import_binlog_ssl_material to fail with MySQL server ERROR 1457.
- Fixed an issue where dump thread initialization could get deadlocked with the thread for purging binary logs. This can stop the active binlog file from rotating and instead continue growing or cause issues with new binlog replica connections.
- Fixed an issue where the query cache can return stale result on an Aurora read replica.

Integration of MySQL Community Edition bug fixes

This release includes all community bug fixes up to and including 5.7, in addition to the below. For more information, see MySQL bugs fixed by Aurora MySQL 2.x database engine updates.

- Fixed an issue where the code for reading character set information from Performance Schema statement events tables (for example, events_statements_current) did not prevent simultaneous writing to that character set information. As a result, the SQL query text character set could be invalid, which could result in a server exit. With this fix, an invalid character set causes SQL_TEXT column truncation and prevents server exits. (Bug #23540008)
- InnoDB: Backport of a fix for Community Bug #25189192, Bug #84038. Fixed an issue where after a RENAME TABLE operation that moved a table to a different schema, InnoDB failed to

update INNODB_SYS_DATAFILES data dictionary table. This resulted in an error on restart indicating that it could not locate the tablespace data file.

- InnoDB: Fixed an issue where the server dropped an internally defined foreign key index when adding a new index and attempted to use a secondary index defined on a virtual generated column as the foreign key index, causing a server exit. InnoDB now permits a foreign key constraint to reference a secondary index defined on a virtual generated column. (Bug #23533396)
- Fixed an issue where two sessions concurrently executing an INSERT ... ON DUPLICATE KEY UPDATE operation generated a deadlock. During partial rollback of a tuple, another session could update it. The fix for this bug reverts the fixes for Bug #11758237, Bug #17604730, and Bug #20040791. (Bug #25966845)
- Backport of a fix for Community Bug #27407480: Fixed an issue where the EXECUTE and ALTER ROUTINE privileges weren't correctly granted to routine creators even with automatic_sp_privileges enabled.
- Backport of fix for Community Bug#24671968: Fixed an issue where a query could produce incorrect results if the WHERE clause contained a dependent subquery, the table had a secondary index on the columns in the select list followed by the columns in the subquery, and `GROUP BY` or `DISTINCT` permitted the query to use a Loose Index Scan.
- Fixed an issue where replication breaks if a multi-table delete statement is issued against multiple tables with foreign keys. (Bug #80821)
- Fixed an issue where in special cases certain slave errors are not ignored even with slave_skip_errors enabled. In cases when opening and locking a table failed or when field conversions failed on a server running row-based replication, the error is considered critical and the state of slave_skip_errors is ignored. The fix ensures that with slave_skip_errors) enabled, all errors reported during applying a transaction are correctly handled. (Bug #70640, Bug #17653275)
- Fixed an issue where a `SET PASSWORD` statement was replicated from a MySQL 5.6 master to a MySQL 5.7 slave, or from a MySQL 5.7 master with the log_builtin_as_identified_by_password system variable set to ON to a MySQL 5.7 slave, the password hash was itself also hashed before being stored on the slave. The issue has now been fixed and the replicated password hash is stored as originally passed to the slave. (Bug#24687073)
- Fixed an issue where serialization of a JSON value consisting of a large sub-document wrapped in many levels of JSON arrays, objects, or both, sometimes required an excessive amount time to complete. (Bug #23031146)

- Statements that cannot be parsed (due, for example, to syntax errors) are no longer written to the slow query log. (Bug #33732907)

Comparison with Aurora MySQL version 1

The following Amazon Aurora MySQL features are supported in Aurora MySQL version 1 (compatible with MySQL 5.6), but these features are currently not supported in Aurora MySQL version 2 (compatible with MySQL 5.7).

- Hash joins. For more information, see Optimizing large Aurora MySQL join queries with hash joins in the *Amazon Aurora User Guide*.
- Native functions for synchronously invoking AWS Lambda functions. For more information, see Invoking a Lambda function with an Aurora MySQL native function in the *Amazon Aurora User Guide*.
- Scan batching. For more information, see Aurora MySQL database engine updates 2017-12-11 (version 1.16) (Deprecated).
- Migrating data from MySQL using an Amazon S3 bucket. For more information, see Migrating data from MySQL by using an Amazon S3 bucket in the *Amazon Aurora User Guide*.

MySQL 5.7 compatibility

This Aurora MySQL version is wire-compatible with MySQL 5.7 and includes features such as JSON support, spatial indexes, and generated columns. Aurora MySQL uses a native implementation of spatial indexing using z-order curves to deliver >20x better write performance and >10x better read performance than MySQL 5.7 for spatial datasets.

This Aurora MySQL version does not currently support the following MySQL 5.7 features:

- Group replication plugin
- Increased page size
- InnoDB buffer pool loading at startup
- InnoDB full-text parser plugin
- Multisource replication
- Online buffer pool resizing
- Password validation plugin

- Query rewrite plugins
- Replication filtering
- The CREATE TABLESPACE SQL statement

Aurora MySQL database engine updates 2022-11-01 (version 2.10.3) (Deprecated)

Version: 2.10.3

Aurora MySQL 2.10.3 is generally available. Aurora MySQL 2.x versions are compatible with MySQL 5.7, and Aurora MySQL 1.x versions are compatible with MySQL 5.6.

Currently supported Aurora MySQL releases are 1.19.5, 1.19.6, 1.22.*, 1.23.*, 2.04.*, 2.07.*, 2.08.*, 2.09.*, 2.10.*, 2.11.*, 3.01.* and 3.02.*.

You can upgrade an existing Aurora MySQL 2.* database cluster to Aurora MySQL 2.10.3. For clusters running Aurora MySQL version 1, you can upgrade an existing Aurora MySQL 1.23 or higher cluster directly to 2.10.3. You can also restore a snapshot from any currently supported Aurora MySQL release into Aurora MySQL 2.10.3.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

> **Note**
>
> For information on how to upgrade your Aurora MySQL database cluster, see Upgrading the minor version or patch level of an Aurora MySQL DB cluster in the *Amazon Aurora User Guide*.

Improvements

Fixed security issues and CVEs listed below:

Fixes and other enhancements to fine-tune handling in a managed environment. Additional CVE fixes below:

- CVE-2022-21444
- CVE-2022-21344
- CVE-2022-21304
- CVE-2022-21245
- CVE-2021-36222
- CVE-2021-22926

General improvements:

- Fixed an issue which, in rare conditions, causes the database server to restart due to a long semaphore wait when the deadlock detector thread gets stuck.
- Fixed an issue which can cause the freeable memory on the database instance to reduce when certain Data Control Language (DCL) SQL statements such as GRANT, FLUSH PRIVILEGES etc. are run on that instance. Frequent use of such statements can cause the freeable memory to keep reducing and may cause the database instance to restart because of out-of-memory issues. Use of such statements on the writer instance can also cause the freeable memory on the reader instances to reduce.
- Fixed an issue where queries against the "performance_schema.events_waits_summary_global_by_event_name" table may become slow when a database instance is under heavy load with the "wait/io/aurora_respond_to_client" performance_schema wait event enabled.
- Fixed an issue which, in rare conditions, can cause the database server to stall and restart when transactions partially roll back due to a constraint violation on the secondary indexes.
- Fixed an issue which, in rare conditions, can cause the writer instance to restart or failover when a transaction accesses a row being deleted by another transaction.
- Fixed an issue which, in rare conditions, can cause the database to restart due to a long semaphore wait when I/O threads become deadlocked.
- Fixed an issue which can cause the read replica to restart during failover in rare conditions when the Unix socket lock file is in use.
- Fixed an issue where excessive query cache invalidation causes higher than expected CPU usage and latencies on the read replica due to the read replica having to read the data from the disk instead of from the query cache.

Integration of MySQL Community Edition bug fixes

This release includes all community bug fixes up to and including 5.7, in addition to the below. For more information, see MySQL bugs fixed by Aurora MySQL 2.x database engine updates.

- Fixed an issue where the code for reading character set information from Performance Schema statement events tables (for example, events_statements_current) did not prevent simultaneous writing to that character set information. As a result, the SQL query text character set could be invalid, which could result in a server exit. With this fix, an invalid character set causes SQL_TEXT column truncation and prevents server exits. (Bug #23540008)
- Fixed an issue when an UPDATE required a temporary table having a primary key larger than 1024 bytes and that table was created using InnoDB, the server could exit. (Bug #25153670)
- Fixed an issue where two sessions concurrently executing an INSERT ... ON DUPLICATE KEY UPDATE operation generated a deadlock. During partial rollback of a tuple, another session could update it. The fix for this bug reverts the fixes for Bug #11758237, Bug #17604730, and Bug #20040791. (Bug #25966845)

Comparison with Aurora MySQL version 1

The following Amazon Aurora MySQL features are supported in Aurora MySQL version 1 (compatible with MySQL 5.6), but these features are currently not supported in Aurora MySQL version 2 (compatible with MySQL 5.7).

- Hash joins. For more information, see Optimizing large Aurora MySQL join queries with hash joins in the *Amazon Aurora User Guide*.
- Native functions for synchronously invoking AWS Lambda functions. For more information, see Invoking a Lambda function with an Aurora MySQL native function in the *Amazon Aurora User Guide*.
- Scan batching. For more information, see Aurora MySQL database engine updates 2017-12-11 (version 1.16) (Deprecated).
- Migrating data from MySQL using an Amazon S3 bucket. For more information, see Migrating data from MySQL by using an Amazon S3 bucket in the *Amazon Aurora User Guide*.

MySQL 5.7 compatibility

This Aurora MySQL version is wire-compatible with MySQL 5.7 and includes features such as JSON support, spatial indexes, and generated columns. Aurora MySQL uses a native implementation of spatial indexing using z-order curves to deliver >20x better write performance and >10x better read performance than MySQL 5.7 for spatial datasets.

This Aurora MySQL version does not currently support the following MySQL 5.7 features:

- Group replication plugin
- Increased page size
- InnoDB buffer pool loading at startup
- InnoDB full-text parser plugin
- Multisource replication
- Online buffer pool resizing
- Password validation plugin
- Query rewrite plugins
- Replication filtering
- The `CREATE TABLESPACE` SQL statement

Aurora MySQL database engine updates 2022-01-26 (version 2.10.2) (Deprecated)

Version: 2.10.2

Aurora MySQL 2.10.2 is generally available. Aurora MySQL 2.x versions are compatible with MySQL 5.7, and Aurora MySQL 1.x versions are compatible with MySQL 5.6.

Currently supported Aurora MySQL releases are 1.19.5, 1.19.6, 1.22.*, 1.23.*, 2.04.*, 2.07.*, 2.08.*, 2.09.*, 2.10.*, 3.01.* and 3.02.*.

You can upgrade an existing Aurora MySQL 2.* database cluster to Aurora MySQL 2.10.0. For clusters running Aurora MySQL version 1, you can upgrade an existing Aurora MySQL 1.23 or higher cluster directly to 2.10.0. You can also restore a snapshot from any currently supported Aurora MySQL release into Aurora MySQL 2.10.0.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

> **Note**
>
> For information on how to upgrade your Aurora MySQL database cluster, see Upgrading the minor version or patch level of an Aurora MySQL DB cluster in the *Amazon Aurora User Guide*.

Improvements

Fixed security issues and CVEs listed below:

Fixes and other enhancements to fine-tune handling in a managed environment. Additional CVE fixes below:

- CVE-2021-36222
- CVE-2021-35624
- CVE-2021-35604
- CVE-2021-22926
- CVE-2021-2390
- CVE-2021-2389
- CVE-2021-2385
- CVE-2021-2356
- CVE-2019-17543
- CVE-2019-2960

General improvements:

- Added a performance optimization to help reduce database IO latency in 24XL instance classes.
- Added support for ECDHE SSL ciphers. For more information on configuring your clients to use these SSL Ciphers please see the following MySQL documentation, encrypted connection protocols ciphers

- Fixed security issues related to Aurora MySQL integration with other AWS Services such as Amazon S3, Amazon ML, and AWS Lambda.
- Fixed an issue which can cause a database instance restart to fail when the database has approximately over 1GB of user and privilege combinations.
- Fixed an issue with Parallel Query which could cause the database to return incorrect groupings or sort order when executing queries with a GROUP BY clause and a WHERE clause that contain a range predicate.
- Fixed an issue which causes general_log and slow_log tables to become inaccessible after an in-place major version upgrade from Aurora MySQL 1.x (compatible with MySQL 5.6) to Aurora MySQL 2.x (compatible with MySQL 5.7).
- Fixed an issue which, in rare cases, causes the database instance to restart when innodb_trx, innodb_locks or innodb_lockwaits tables are queried while the database is under heavy workload. Monitoring tools such as Performance Insights may query such tables.
- Fixed an issue where the value of a TIMESTAMP column of an existing row is updated to the latest timestamp when all of the following conditions are satisfied:
  1. A trigger exists for the table.
  2. An INSERT is performed on the table that has an ON DUPLICATE KEY UPDATE clause.
  3. The inserted row causes a duplicate value violation in a UNIQUE index or PRIMARY KEY.
  4. One or more columns are of TIMESTAMP data type and have a default value of CURRENT_TIMESTAMP.
- Fixed an issue which, in rare cases, could prevent a binlog replica from connecting to an instance with binlog enabled.
- Fixed an issue where, in rare conditions, transactions were unable to commit when running on an instance with binlog enabled.
- Fixed an issue where new connections could not be established to an instance with binlog enabled.
- Fixed an issue which can cause excessive internal logging when attempting zero downtime patching and restart causing local storage to fill up.
- Fixed an issue that causes a binlog replica to stop with an HA_ERR_FOUND_DUPP_KEY error when replicating certain DDL and DCL statements. The issue occurs when the source instance is configured with MIXED binary logging format and READ COMMITTED or READ UNCOMMITTED isolation level.

- Fixed an issue where the binlog replication I/O thread is unable to keep up with the primary instance, when multi-threaded replication is enabled
- Fixed an issue where, in rare conditions, a high number of active connections to the database instance may cause the CloudWatch CommitLatency metric to be incorrectly reported.
- Fixed an issue which causes local storage on Graviton instances to fill up when performing LOAD FROM S3 or SELECT INTO S3.
- Fixed an issue which can cause wrong query results when querying a table with a foreign key and both of the following conditions are met:

  1. Query cache is enabled
  2. A transaction with a cascading delete or update on that table is rolled back
- Fixed an issue which, in rare conditions, can cause Aurora reader instances to restart. The chance of this issue occurring increases as the number of transaction rollbacks increases.
- Fixed an issue where the number of mutex 'LOCK_epoch_id_master' occurrences in Performance Schema increases when a session is opened and closed.
- Fixed an issue which can cause an increasing number of deadlocks for workloads which have many transactions updating the same set of rows concurrently.
- Fixed an issue which, in rare conditions, can cause the instances to restart when the database volume grows to a multiple of 160GB.
- Fixed an issue with Parallel Query which could cause the database to restart when executing SQL statements with a LIMIT clause.
- Fixed an issue which, in rare conditions, can cause the database instance to restart when using XA transactions with the READ COMMITTED isolation level.
- Fixed an issue where, after an Aurora Read instance restarts, it may restart again if there is a heavy DDL workload during the restart.
- Fixed an issue with incorrect reporting of Aurora reader replication lag.
- Fixed an issue which, in rare conditions, can cause a writer instance to restart when an in-memory data-integrity check fails.
- Fixed an issue which, in rare conditions, incorrectly shows the “Database Load” chart in Performance Insights (PI) sessions as actively using CPU even though the sessions have finished processing and are idle.
- Fixed an issue which, in rare conditions, can cause the database server to restart when a query is processed using Parallel Query.

- Fixed an issue which, in rare conditions, can cause the writer instance in a primary Global Database cluster to restart because of a race condition during Global Database replication.
- Fixed an issue that can occur during a database instance restart, which can cause more than one restart.

Integration of MySQL Community Edition bug fixes

- Fixed an issue in InnoDB where an error in code related to table statistics raised an assertion in the dict0stats.cc source file. (Bug #24585978)
- Fixed an issue where a secondary index over a virtual column became corrupted when the index was built online. For UPDATE statements, we fix this as follows: If the virtual column value of the index record is set to NULL, then we generate this value from the cluster index record. (Bug #30556595))
- Fixed an issue in InnoDB where deleting marked rows were able to acquire an external read lock before a partial rollback was completed. The external read lock prevented conversion of an implicit lock to an explicit lock during the partial rollback, causing an assertion failure. (Bug #29195848)
- Fixed an issue where the empty host names in accounts could cause the server to misbehave. (Bug #28653104)
- Fixed an issue in InnoDB where a query interruption during a lock wait caused an error. (Bug #28068293)
- Fixed an issue in replication where Interleaved transactions could sometimes deadlock the slave applier when the transaction isolation level was set to REPEATABLE READ. (Bug #25040331)
- Fixed an issue which can cause binlog replicas to stall due to lock wait timeout. (Bug #27189701)

Comparison with Aurora MySQL version 1

The following Amazon Aurora MySQL features are supported in Aurora MySQL version 1 (compatible with MySQL 5.6), but these features are currently not supported in Aurora MySQL version 2 (compatible with MySQL 5.7).

- Hash joins. For more information, see Optimizing large Aurora MySQL join queries with hash joins in the *Amazon Aurora User Guide*.

- Native functions for synchronously invoking AWS Lambda functions. For more information, see Invoking a Lambda function with an Aurora MySQL native function in the *Amazon Aurora User Guide*.
- Scan batching. For more information, see Aurora MySQL database engine updates 2017-12-11 (version 1.16) (Deprecated).
- Migrating data from MySQL using an Amazon S3 bucket. For more information, see Migrating data from MySQL by using an Amazon S3 bucket in the *Amazon Aurora User Guide*.

MySQL 5.7 compatibility

This Aurora MySQL version is wire-compatible with MySQL 5.7 and includes features such as JSON support, spatial indexes, and generated columns. Aurora MySQL uses a native implementation of spatial indexing using z-order curves to deliver >20x better write performance and >10x better read performance than MySQL 5.7 for spatial datasets.

This Aurora MySQL version does not currently support the following MySQL 5.7 features:

- Group replication plugin
- Increased page size
- InnoDB buffer pool loading at startup
- InnoDB full-text parser plugin
- Multisource replication
- Online buffer pool resizing
- Password validation plugin
- Query rewrite plugins
- Replication filtering
- The `CREATE TABLESPACE` SQL statement

Aurora MySQL database engine updates 2021-10-21 (version 2.10.1) (Deprecated)

Version: 2.10.1

Aurora MySQL 2.10.1 is generally available. Aurora MySQL 2.x versions are compatible with MySQL 5.7, and Aurora MySQL 1.x versions are compatible with MySQL 5.6.

Currently supported Aurora MySQL releases are 1.19.5, 1.19.6, 1.22.*, 1.23.*, 2.04.*, 2.07.*, 2.08.*, 2.09.*, 2.10.*, 3.01.* and 3.02.*.

You can upgrade an existing Aurora MySQL 2.* database cluster to Aurora MySQL 2.10.0. For clusters running Aurora MySQL version 1, you can upgrade an existing Aurora MySQL 1.23 or higher cluster directly to 2.10.0. You can also restore a snapshot from any currently supported Aurora MySQL release into Aurora MySQL 2.10.0.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

> **Note**
>
> For information on how to upgrade your Aurora MySQL database cluster, see Upgrading the minor version or patch level of an Aurora MySQL DB cluster in the *Amazon Aurora User Guide*.

Improvements

Fixed security issues and CVEs listed below:

Fixes and other enhancements to fine-tune handling in a managed environment. Additional CVE fixes below:

- CVE-2021-2307
- CVE-2021-2226
- CVE-2021-2194
- CVE-2021-2174
- CVE-2021-2171
- CVE-2021-2169
- CVE-2021-2166
- CVE-2021-2160

- CVE-2021-2154
- CVE-2021-2032
- CVE-2021-2001

Availability improvements:

- Added the ability to cleanly shut down the cluster for future major version upgrades.

General improvements:

- Fixed an issue that can cause high CPU consumption on the reader instances due to excessive logging of informational messages in internal diagnostic log files.
- Fixed an issue where the value of a TIMESTAMP column of an existing row is updated to the latest timestamp when all of the following conditions are satisfied:
  1. A trigger exists for the table.
  2. An INSERT is performed on the table that has an ON DUPLICATE KEY UPDATE clause.
  3. The inserted row causes a duplicate value violation in a UNIQUE index or PRIMARY KEY.
  4. One or more columns are of TIMESTAMP data type and have a default value of CURRENT_TIMESTAMP.
- Fixed an issue introduced in version 2.10.0 which causes use of json_merge function to raise an error code in certain cases. In particular, when json_merge function is used in a DDL containing generated columns, it can return error code 1305.
- Fixed an issue where, in rare conditions, read replicas restarts when a large object's update history is being validated for a transaction's read view on the read replica.
- Fixed an issue which, in rare conditions, causes a writer instance to restart when an in-memory data- integrity check fails.

Integration of MySQL community edition bug fixes

- CURRENT_TIMESTAMP PRODUCES ZEROS IN TRIGGER. (Bug #25209512)

Comparison with Aurora MySQL version 1

The following Amazon Aurora MySQL features are supported in Aurora MySQL version 1 (compatible with MySQL 5.6), but these features are currently not supported in Aurora MySQL version 2 (compatible with MySQL 5.7).

- Hash joins. For more information, see Optimizing large Aurora MySQL join queries with hash joins in the *Amazon Aurora User Guide*.
- Native functions for synchronously invoking AWS Lambda functions. For more information, see Invoking a Lambda function with an Aurora MySQL native function in the *Amazon Aurora User Guide*.
- Scan batching. For more information, see Aurora MySQL database engine updates 2017-12-11 (version 1.16) (Deprecated).
- Migrating data from MySQL using an Amazon S3 bucket. For more information, see Migrating data from MySQL by using an Amazon S3 bucket in the *Amazon Aurora User Guide*.

MySQL 5.7 compatibility

This Aurora MySQL version is wire-compatible with MySQL 5.7 and includes features such as JSON support, spatial indexes, and generated columns. Aurora MySQL uses a native implementation of spatial indexing using z-order curves to deliver >20x better write performance and >10x better read performance than MySQL 5.7 for spatial datasets.

This Aurora MySQL version does not currently support the following MySQL 5.7 features:

- Group replication plugin
- Increased page size
- InnoDB buffer pool loading at startup
- InnoDB full-text parser plugin
- Multisource replication
- Online buffer pool resizing
- Password validation plugin
- Query rewrite plugins
- Replication filtering
- The `CREATE TABLESPACE` SQL statement

Aurora MySQL database engine updates 2021-05-25 (version 2.10.0) (Deprecated)

Version: 2.10.0

Aurora MySQL 2.10.0 is generally available. Aurora MySQL 2.x versions are compatible with MySQL 5.7, and Aurora MySQL 1.x versions are compatible with MySQL 5.6.

Currently supported Aurora MySQL releases are 1.19.5, 1.19.6, 1.22.*, 1.23.*, 2.04.*, 2.07.*, 2.08.*, 2.09.*, 2.10.*, 3.01.* and 3.02.*.

You can upgrade an existing Aurora MySQL 2.* database cluster to Aurora MySQL 2.10.0. For clusters running Aurora MySQL version 1, you can upgrade an existing Aurora MySQL 1.23 or higher cluster directly to 2.10.0. You can also restore a snapshot from any currently supported Aurora MySQL release into Aurora MySQL 2.10.0.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

> **Note**
>
> For information on how to upgrade your Aurora MySQL database cluster, see Upgrading the minor version or patch level of an Aurora MySQL DB cluster in the *Amazon Aurora User Guide*.

Improvements

Fixed security issues and CVEs listed below:

Fixes and other enhancements to fine-tune handling in a managed environment. Additional CVE fixes below:

- CVE-2021-23841
- CVE-2021-3449
- CVE-2020-28196
- CVE-2020-14790
- CVE-2020-14776

- CVE-2020-14567
- CVE-2020-14559
- CVE-2020-14553
- CVE-2020-14547
- CVE-2020-14540
- CVE-2020-14539
- CVE-2018-3251
- CVE-2018-3156
- CVE-2018-3143
- CVE-2016-5440

New features:

- The `db.t3.large` instance class is now supported for Aurora MySQL.
- *Binary log replication:*
  - Introduced the binlog I/O cache to improve binlog performance by reducing contention between writer threads and dump threads. For more information, see Optimizing binary log replication in the *Amazon Aurora User Guide*.
  - In Aurora MySQL version 2.08, we introduced improved binary log (binlog) processing to reduce crash recovery time and commit time latency when very large transactions are involved. These improvements are now supported for clusters that have GTID enabled.
- *Improved reader instance availability:*
  - Previously, when a writer instance restarted, all reader instances in an Aurora MySQL cluster restarted as well. With today's launch, in-Region reader instances continue to serve read requests during a writer instance restart, improving read availability in the cluster. For more information, see Rebooting an Aurora MySQL cluster (version 2.10 and higher) in the *Amazon Aurora User Guide*.

    Important

    After you upgrade to Aurora MySQL 2.10, rebooting the writer instance doesn't perform a reboot of the entire cluster. If you want to reboot the entire cluster, now you reboot any reader instances in the cluster after rebooting the writer instance.

- Improved the performance of the read ahead page reads requested by logical read ahead (LRA) technique. This was done by batching the multiple page reads in a single request sent to Aurora storage. As a result, the queries that use the LRA optimization execute up to 3x faster.
- *Zero-downtime restarts and patching:*
  - Improved zero-downtime restart (ZDR) and zero-downtime patching (ZDP) to enable ZDR and ZDP in a wider range of scenarios, including the added support for cases when binary logging is enabled. Also, improved visibility into ZDR and ZDP events. See documentation for details: Zero-downtime restart (ZDR) for Amazon Aurora MySQL and Using zero-downtime patching in the *Amazon Aurora User Guide*.

Availability improvements:

- Improvements for faster startup when the database has a large number of temporary indexes and tables created during a prior interrupted DDL activity.
- Fixed multiple issues related to repeated restarts during the crash recovery of interrupted DDL operations, such as `DROP TRIGGER`, `ALTER TABLE`, and specifically `ALTER TABLE` that modifies the type of partitioning or number of partitions in a table.
- Fixed an issue that could cause a server restart during Database Activity Streams (DAS) log processing.
- Fixed an issue printing an error message while processing an `ALTER` query on system tables.

General improvements:

- Fixed an issue where the query cache could return stale results on a reader instance.
- Fixed an issue where some Aurora commit metrics were not being updated when the system variable `innodb_flush_log_at_trx_commit` was set to 0 or 2.
- Fixed an issue where a query result stored in the query cache was not refreshed by multistatement transactions.
- Fixed an issue that could cause the last-modified timestamp of binary log files to not be updated correctly. This could lead to binary log files being purged prematurely, before reaching the customer-configured retention period.
- Fixed incorrect reported binlog file name and position from InnoDB after crash recovery.
- Fixed an issue that could cause large transactions to generate incorrect binlog events if the `binlog_checksum` parameter was set to `NONE`.

- Fixed an issue that caused a binlog replica to stop with an error if the replicated transaction contained a DDL statement and a large number of row changes.
- Fixed an issue leading to a restart in a reader instance when dropping a table.
- Fixed an issue that caused open source connectors to fail when attempting to consume a binlog file with a large transaction.
- Fixed an issue that could lead to incorrect query results on the large geometry column after creating a spatial index on the table with the large geometry values.
- The database now recreates the temporary tablespace during restart, which allows the associated storage space to be freed and reclaimed.
- Fixed an issue that prevented `performance_schema` tables from being truncated on Aurora reader instances.
- Fixed an issue that caused a binlog replica to stop with an `HA_ERR_KEY_NOT_FOUND` error.
- Fixed an issue that caused the database to restart when running `FLUSH TABLES WITH READ LOCK` statement.
- Fixed an issue that prevented the use of user-level lock functions on Aurora reader instances.

Integration of MySQL community edition bug fixes

- Interleaved transactions could sometimes deadlock the replica applier when the transaction isolation level was set to REPEATABLE READ. (Bug #25040331)
- When a stored procedure contained a statement referring to a view which in turn referred to another view, the procedure could not be invoked successfully more than once. (Bug #87858, Bug #26864199)
- For queries with many `OR` conditions, the optimizer now is more memory-efficient and less likely to exceed the memory limit imposed by the range_optimizer_max_mem_size system variable. In addition, the default value for that variable has been raised from 1,536,000 to 8,388,608. (Bug #79450, Bug #22283790)
- *Replication:* In the `next_event()` function, which is called by a replica's SQL thread to read the next event from the relay log, the SQL thread did not release the `relaylog.log_lock` it acquired when it ran into an error (for example, due to a closed relay log), causing all other threads waiting to acquire a lock on the relay log to hang. With this fix, the lock is released before the SQL thread leaves the function under the situation. (Bug #21697821)
- Fixing a memory corruption for `ALTER TABLE` with virtual column. (Bug #24961167; Bug #24960450)

- *Replication:* Multithreaded replicas could not be configured with small queue sizes using slave_pending_jobs_size_max if they ever needed to process transactions larger than that size. Any packet larger than slave_pending_jobs_size_max was rejected with the error ER_MTS_EVENT_BIGGER_PENDING_JOBS_SIZE_MAX, even if the packet was smaller than the limit set by slave_max_allowed_packet. With this fix, slave_pending_jobs_size_max becomes a soft limit rather than a hard limit. If the size of a packet exceeds slave_pending_jobs_size_max but is less than slave_max_allowed_packet, the transaction is held until all the replica workers have empty queues, and then processed. All subsequent transactions are held until the large transaction has been completed. The queue size for replica workers can therefore be limited while still allowing occasional larger transactions. (Bug #21280753, Bug #77406)
- *Replication:* When using a multithreaded replica, applier errors displayed worker ID data that was inconsistent with data externalized in Performance Schema replication tables. (Bug #25231367)
- *Replication:* On a GTID-based replication replica running with -gtid-mode=ON, -log-bin=OFF, and using -slave-skip-errors, when an error was encountered that should be ignored `Exec_Master_Log_Pos` was not being correctly updated, causing `Exec_Master_Log_Pos` to loose synchrony with `Read_master_log_pos`. If a `GTID_NEXT` was not specified, the replica would never update its GTID state when rolling back from a single statement transaction. The `Exec_Master_Log_Pos` would not be updated because even though the transaction was finished, its GTID state would show otherwise. The fix removes the restraint of updating the GTID state when a transaction is rolled back only if `GTID_NEXT` is specified. (Bug #22268777)
- *Replication:* A partially failed statement was not correctly consuming an auto-generated or specified GTID when binary logging was disabled. The fix ensures that a partially failed DROP TABLE, a partially failed DROP USER, or a partially failed DROP VIEW consume respectively the relevant GTID and save it into `@@GLOBAL.GTID_EXECUTED` and `mysql.gtid_executed` table when binary logging is disabled. (Bug #21686749)
- *Replication:* Replicas running MySQL 5.7 could not connect to a MySQL 5.5 source due to an error retrieving the server_uuid, which is not part of MySQL 5.5. This was caused by changes in the method of retrieving the `server_uuid`. (Bug #22748612)
- *Replication:* The GTID transaction skipping mechanism that silently skips a GTID transaction that was previously executed did not work properly for XA transactions. (Bug #25041920)
- ">XA ROLLBACK statements that failed because an incorrect transaction ID was given, could be recorded in the binary log with the correct transaction ID, and could therefore be actioned by replication replicas. A check is now made for the error situation before binary logging takes place, and failed XA `ROLLBACK` statements are not logged. (Bug #26618925)

- *Replication:* If a replica was set up using a CHANGE MASTER TO statement that did not specify the source log file name and source log position, then shut down before START SLAVE was issued, then restarted with the option -relay-log-recovery set, replication did not start. This happened because the receiver thread had not been started before relay log recovery was attempted, so no log rotation event was available in the relay log to provide the source log file name and source log position. In this situation, the replica now skips relay log recovery and logs a warning, then proceeds to start replication. (Bug #28996606, Bug #93397)
- *Replication:* In row-based replication, a message that incorrectly displayed field lengths was returned when replicating from a table with a `utf8mb3` column to a table of the same definition where the column was defined with a `utf8mb4` character set. (Bug #25135304, Bug #83918)
- *Replication:* When a RESET SLAVE statement was issued on a replication replica with GTIDs in use, the existing relay log files were purged, but the replacement new relay log file was generated before the set of received GTIDs for the channel had been cleared. The former GTID set was therefore written to the new relay log file as the `PREVIOUS_GTIDS` event, causing a fatal error in replication stating that the replica had more GTIDs than the source, even though the gtid_executed set for both servers was empty. Now, when `RESET SLAVE` is issued, the set of received GTIDs is cleared before the new relay log file is generated, so that this situation does not occur. (Bug #27411175)
- *Replication:* With GTIDs in use for replication, transactions including statements that caused a parsing error (ER_PARSE_ERROR) could not be skipped manually by the recommended method of injecting an empty or replacement transaction with the same GTID. This action should result in the replica identifying the GTID as already used, and therefore skipping the unwanted transaction that shared its GTID. However, in the case of a parsing error, because the statement was parsed before the GTID was checked to see if it needed to be skipped, the replication applier thread stopped due to the parsing error, even though the intention was for the transaction to be skipped anyway. With this fix, the replication applier thread now ignores parsing errors if the transaction concerned needs to be skipped because the GTID was already used. Note that this behavior change does not apply in the case of workloads consisting of binary log output produced by `mysqlbinlog`. In that situation, there would be a risk that a transaction with a parsing error that immediately follows a skipped transaction would also be silently skipped, when it ought to raise an error. (Bug #27638268)
- *Replication:* Enable the SQL thread to GTID skip a partial transaction. (Bug #25800025)
- *Replication:* When a negative or fractional timeout parameter was supplied to `WAIT_UNTIL_SQL_THREAD_AFTER_GTIDS()`, the server behaved in unexpected ways. With this fix:

  - A fractional timeout value is read as-is, with no round-off.
  - A negative timeout value is rejected with an error if the server is on a strict SQL mode; if the server is not on a strict SQL mode, the value makes the function return `NULL` immediately without any waiting and then issue a warning. (Bug #24976304, Bug #83537)
- *Replication:* If the `WAIT_FOR_EXECUTED_GTID_SET()` function was used with a timeout value including a fractional part (for example, 1.5), an error in the casting logic meant that the timeout was rounded down to the nearest whole second, and to zero for values less than 1 second (for example, 0.1). The casting logic has now been corrected so that the timeout value is applied as originally specified with no rounding. Thanks to Dirkjan Bussink for the contribution. (Bug #29324564, Bug #94247)
- With GTIDs enabled, XA COMMIT on a disconnected XA transaction within a multiple-statement transaction raised an assertion. (Bug #22173903)
- *Replication:* An assertion was raised in debug builds if an XA ROLLBACK statement was issued for an unknown transaction identifier when the gtid_next value had been set manually. The server now does not attempt to update the GTID state if an XA `ROLLBACK` statement fails with an error. (Bug #27928837, Bug #90640)
- Fix wrong sorting order issue when multiple `CASE` functions are used in `ORDER BY` clause (Bug#22810883).
- Some queries that used ordering could access an uninitialized column during optimization and cause a server exit. (Bug #27389294)

Comparison with Aurora MySQL version 1

The following Amazon Aurora MySQL features are supported in Aurora MySQL version 1 (compatible with MySQL 5.6), but these features are currently not supported in Aurora MySQL version 2 (compatible with MySQL 5.7).

- Hash joins. For more information, see Optimizing large Aurora MySQL join queries with hash joins in the *Amazon Aurora User Guide*.
- Native functions for synchronously invoking AWS Lambda functions. For more information, see Invoking a Lambda function with an Aurora MySQL native function in the *Amazon Aurora User Guide*.
- Scan batching. For more information, see Aurora MySQL database engine updates 2017-12-11 (version 1.16) (Deprecated).

- Migrating data from MySQL using an Amazon S3 bucket. For more information, see Migrating data from MySQL by using an Amazon S3 bucket in the *Amazon Aurora User Guide*.

MySQL 5.7 compatibility

This Aurora MySQL version is wire-compatible with MySQL 5.7 and includes features such as JSON support, spatial indexes, and generated columns. Aurora MySQL uses a native implementation of spatial indexing using z-order curves to deliver >20x better write performance and >10x better read performance than MySQL 5.7 for spatial datasets.

This Aurora MySQL version does not currently support the following MySQL 5.7 features:

- Group replication plugin
- Increased page size
- InnoDB buffer pool loading at startup
- InnoDB full-text parser plugin
- Multisource replication
- Online buffer pool resizing
- Password validation plugin
- Query rewrite plugins
- Replication filtering
- The `CREATE TABLESPACE` SQL statement

Aurora MySQL database engine updates 2021-11-12 (version 2.09.3) (Deprecated)

Version: 2.09.3

Aurora MySQL 2.09.3 is generally available. Aurora MySQL 2.x versions are compatible with MySQL 5.7, and Aurora MySQL 1.x versions are compatible with MySQL 5.6.

Currently supported Aurora MySQL releases are 1.19.5, 1.19.6, 1.22.*, 1.23.*, 2.04.*, 2.07.*, 2.08.*, 2.09.*, 2.10.*, 3.01.* and 3.02.*.

You can upgrade an existing Aurora MySQL 2.* database cluster to Aurora MySQL 2.10.0. For clusters running Aurora MySQL version 1, you can upgrade an existing Aurora MySQL 1.23 or

higher cluster directly to 2.10.0. You can also restore a snapshot from any currently supported Aurora MySQL release into Aurora MySQL 2.10.0.

To create a cluster with an older version of Aurora MySQL, specify the engine version through the AWS Management Console, the AWS CLI, or the Amazon RDS API.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

> **Note**
>
> For information on how to upgrade your Aurora MySQL database cluster, see Upgrading the minor version or patch level of an Aurora MySQL DB cluster in the *Amazon Aurora User Guide*.

Improvements

Security fixes:

Fixes and other enhancements to fine-tune handling in a managed environment. Additional CVE fixes below:

- CVE-2021-23841
- CVE-2021-3712
- CVE-2021-3449
- CVE-2021-2307
- CVE-2021-2226
- CVE-2021-2174
- CVE-2021-2171
- CVE-2021-2169
- CVE-2021-2166
- CVE-2021-2154
- CVE-2021-2060

- CVE-2021-2032
- CVE-2021-2001
- CVE-2020-28196
- CVE-2020-14769
- CVE-2019-17543
- CVE-2019-2960

Availability improvements:

- Introduced an optimization which can reduce contention for queries that are executed on tables in information_schema.
- Add support for ECDHE SSL ciphers.

General improvements:

- Fixed an issue which, in rare conditions, can cause a writer instance to restart when an in-memory data-integrity check fails.
- Fixed an issue which, in rare conditions, can cause the database instance to restart when the cluster volume is expanding while binary logging is enabled.
- Fixed a rare race condition during a database instance restart, which can cause more than one restart.
- Fixed an issue which can cause a database instance restart to fail when the database has a large number of user and privilege combinations.
- Fixed an issue with parallel query which can cause the database to restart when executing SQL statements with LIMIT clause.
- Fixed an issue with incorrect reporting of aurora replication lag.
- Fixed an issue which can cause general_log and slow_log tables to become inaccessible after in-place major version upgrade from Aurora-MySQL 1.x (based on MySQL 5.6) to Aurora-MySQL 2.x (based on MySQL 5.7).
- Fixed an issue which, in rare cases, can cause the database instance to restart when innodb_trx, innodb_locks or innodb_lockwaits tables are queried while the database is under heavy workload. Monitoring tools and features such as performance insights may query such tables.
- Fixed an issue which can cause a database instance to restart when "FLUSH TABLES WITH READ LOCK" SQL statement is executed.

- Fixed an issue where the InnoDB purge process pauses during the deletion of a reader instance leading to a temporary increase in history list length.
- Fixed an issue with parallel query which can cause the database to restart when executing a SQL statement against a table containing a virtual column.
- Fixed an issue with parallel query which can cause the database to return incorrect groupings or sort order when executing queries with GROUP BY clause and a WHERE clause containing a range predicate.
- Fixed an issue in parallel query which, in rare conditions, can cause the database to restart when executing SQL statements with JSON functions.
- Fixed an issue which, in rare conditions, can cause the writer instance in primary Global Database cluster to restart because of a race condition during Global Database Replication.
- Fixed an issue that can cause a Binlog replica to stop with an HA_ERR_FOUND_DUPP_KEY error when replicating certain DDL and DCL statements. The issue occurs when the source instance is configured with MIXED binary logging format and READ COMMITTED or READ UNCOMMITTED isolation level.
- Fixed an issue which, in rare conditions, can cause the database instance to restart when using XA transactions in READ COMMITTED isolation level.
- Fixed an issue where the value of a TIMESTAMP column of an existing row is updated to the latest timestamp when all of the following conditions are satisfied: 1. a trigger exists for the table; 2. an INSERT is performed on the table that has an ON DUPLICATE KEY UPDATE clause; 3. the inserted row can cause a duplicate value violation in a UNIQUE index or PRIMARY KEY; and 4. one or more columns are of TIMESTAMP data type and have a default value of CURRENT_TIMESTAMP.
- Fixed an issue which, in rare conditions, can cause a reader instance to restart due to an incorrect check processing.
- Fixed an issue which can cause the reader instance to restart when the writer instance grows the database volume to cross specific volume size boundaries.
- Fixed an issue which can cause longer restart times for database instances using cloned cluster volumes.
- Fixed an issue where a database instance restart may fail one or more times after a TRUNCATE TABLE operation was performed on the writer instance.
- Fixed an issue which, in rare conditions, can cause the database instance to restart.
- Fixed an issue which, in rare conditions, can cause the writer instance to restart when the database volume grows to a multiple of 160GB.

Integration of MySQL community edition bug fixes

- Bug #23533396 - When adding a new index, the server dropped an internally defined foreign key index and attempted to use a secondary index defined on a virtual generated column as the foreign key index, causing a server exit. InnoDB now permits a foreign key constraint to reference a secondary index defined on a virtual generated column.
- Bug #29550513 - Replication: A locking issue in the WAIT_FOR_EXECUTED_GTID_SET() function can cause the server to hang in certain circumstances. The issue has now been corrected.

Comparison with Aurora MySQL version 1

The following Amazon Aurora MySQL features are supported in Aurora MySQL version 1 (compatible with MySQL 5.6), but these features are currently not supported in Aurora MySQL version 2 (compatible with MySQL 5.7).

- Asynchronous key prefetch (AKP). For more information, see Optimizing Aurora indexed join queries with asynchronous key prefetch in the *Amazon Aurora User Guide*.
- Hash joins. For more information, see Optimizing large Aurora MySQL join queries with hash joins in the *Amazon Aurora User Guide*.
- Native functions for synchronously invoking AWS Lambda functions. For more information, see Invoking a Lambda function with an Aurora MySQL native function in the *Amazon Aurora User Guide*.
- Scan batching. For more information, see Aurora MySQL database engine updates 2017-12-11 (version 1.16) (Deprecated).
- Migrating data from MySQL using an Amazon S3 bucket. For more information, see Migrating data from MySQL by using an Amazon S3 bucket in the *Amazon Aurora User Guide*.

MySQL 5.7 compatibility

This Aurora MySQL version is wire-compatible with MySQL 5.7 and includes features such as JSON support, spatial indexes, and generated columns. Aurora MySQL uses a native implementation of spatial indexing using z-order curves to deliver >20x better write performance and >10x better read performance than MySQL 5.7 for spatial datasets.

This Aurora MySQL version does not currently support the following MySQL 5.7 features:

- Group replication plugin
- Increased page size
- InnoDB buffer pool loading at startup
- InnoDB full-text parser plugin
- Multisource replication
- Online buffer pool resizing
- Password validation plugin
- Query rewrite plugins
- Replication filtering
- The `CREATE TABLESPACE` SQL statement

Aurora MySQL database engine updates 2021-02-26 (version 2.09.2) (Deprecated)

Version: 2.09.2

Aurora MySQL 2.09.2 is generally available. Aurora MySQL 2.x versions are compatible with MySQL 5.7 and Aurora MySQL 1.x versions are compatible with MySQL 5.6.

Currently supported Aurora MySQL releases are 1.19.5, 1.19.6, 1.22.*, 1.23.*, 2.04.*, 2.07.*, 2.08.*, 2.09.*, 2.10.*, 3.01.* and 3.02.*.

You can upgrade an existing Aurora MySQL 2.* database cluster to Aurora MySQL 2.09.2. For clusters running Aurora MySQL version 1, you can upgrade an existing Aurora MySQL 1.23 or higher cluster directly to 2.09.2. You can also restore a snapshot from any currently supported Aurora MySQL release into Aurora MySQL 2.09.2.

To create a cluster with an older version of Aurora MySQL, specify the engine version through the AWS Management Console, the AWS CLI, or the Amazon RDS API.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

> **Note**
>
> For information on how to upgrade your Aurora MySQL database cluster, see Upgrading the minor version or patch level of an Aurora MySQL DB cluster in the *Amazon Aurora User Guide*.

Improvements

New features:

- Aurora MySQL clusters now support the following EC2 R6g instances powered by Arm-based AWS Graviton2 processors: `r6g.large`, `r6g.xlarge`, `r6g.2xlarge`, `r6g.4xlarge`, `r6g.8xlarge`, `r6g.12xlarge`, `r6g.16xlarge`. For more information, see Aurora DB instance classes in the *Amazon Aurora User Guide*.

Security fixes:

Fixes and other enhancements to fine-tune handling in a managed environment. Additional CVE fixes below:

- CVE-2020-14775
- CVE-2020-14793
- CVE-2020-14765
- CVE-2020-14769
- CVE-2020-14812
- CVE-2020-14760
- CVE-2020-14672
- CVE-2020-14790
- CVE-2020-1971

Availability improvements:

- Fixed an issue introduced in 2.09.0 that can cause elevated write latency during the scaling of the cluster storage volume.

- Fixed an issue in the dynamic resizing feature that could cause Aurora Read Replicas to restart.
- Fixed an issue that could cause longer downtime during upgrade from 1.23.* to 2.09.*.
- Fixed an issue where a DDL or DML could cause engine restart during a page prefetch request.
- Fixed an issue that caused a binlog replica to stop with an error if the replicated transaction contains a DDL statement and a large number of row changes.
- Fixed an issue where a database acting as a binlog replica could restart while replicating a DDL event on the mysql `time_zone` table.
- Fixed an issue that could cause large transactions to generate incorrect binlog events if the `binlog_checksum` parameter was set to `NONE`.
- Fixed an issue that caused a binlog replica to stop with an `HA_ERR_KEY_NOT_FOUND` error.
- Improved overall stability.

Comparison with Aurora MySQL version 1

The following Amazon Aurora MySQL features are supported in Aurora MySQL Version 1 (compatible with MySQL 5.6), but these features are currently not supported in Aurora MySQL Version 2 (compatible with MySQL 5.7).

- Asynchronous key prefetch (AKP). For more information, see Optimizing Aurora indexed join queries with asynchronous key prefetch in the *Amazon Aurora User Guide*.
- Hash joins. For more information, see Optimizing large Aurora MySQL join queries with hash joins in the *Amazon Aurora User Guide*.
- Native functions for synchronously invoking AWS Lambda functions. For more information, see Invoking a Lambda function with an Aurora MySQL native function in the *Amazon Aurora User Guide*.
- Scan batching. For more information, see Aurora MySQL database engine updates 2017-12-11 (version 1.16) (Deprecated).
- Migrating data from MySQL using an Amazon S3 bucket. For more information, see Migrating data from MySQL by using an Amazon S3 bucket in the *Amazon Aurora User Guide*.

MySQL 5.7 compatibility

This Aurora MySQL version is wire-compatible with MySQL 5.7 and includes features such as JSON support, spatial indexes, and generated columns. Aurora MySQL uses a native implementation of

spatial indexing using z-order curves to deliver >20x better write performance and >10x better read performance than MySQL 5.7 for spatial datasets.

This Aurora MySQL version does not currently support the following MySQL 5.7 features:

- Group replication plugin
- Increased page size
- InnoDB buffer pool loading at startup
- InnoDB full-text parser plugin
- Multisource replication
- Online buffer pool resizing
- Password validation plugin
- Query rewrite plugins
- Replication filtering
- The `CREATE TABLESPACE` SQL statement

Aurora MySQL database engine updates 2020-12-11 (version 2.09.1) (Deprecated)

Version: 2.09.1

Aurora MySQL 2.09.1 is generally available. Aurora MySQL 2.x versions are compatible with MySQL 5.7 and Aurora MySQL 1.x versions are compatible with MySQL 5.6.

Currently supported Aurora MySQL releases are 1.19.5, 1.19.6, 1.22.*, 1.23.*, 2.04.*, 2.07.*, 2.08.*, 2.09.*, 2.10.*, 3.01.* and 3.02.*.

You can upgrade an existing Aurora MySQL 2.* database cluster to Aurora MySQL 2.09.1. For clusters running Aurora MySQL version 1, you can upgrade an existing Aurora MySQL 1.23 or higher cluster directly to 2.09.1. You can also restore a snapshot from any currently supported Aurora MySQL release into Aurora MySQL 2.09.1.

To create a cluster with an older version of Aurora MySQL, specify the engine version through the AWS Management Console, the AWS CLI, or the RDS API.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

> **Note**
>
> For information on how to upgrade your Aurora MySQL database cluster, see Upgrading the minor version or patch level of an Aurora MySQL DB cluster in the *Amazon Aurora User Guide*.

Improvements

Security fixes:

Fixes and other enhancements to fine-tune handling in a managed environment. Additional CVE fixes below:

- CVE-2020-14567
- CVE-2020-14559
- CVE-2020-14553
- CVE-2020-14547
- CVE-2020-14540
- CVE-2020-2812
- CVE-2020-2806
- CVE-2020-2780
- CVE-2020-2765
- CVE-2020-2763
- CVE-2020-2760
- CVE-2020-2579

Incompatible changes:

This version introduces a permission change that affects the behavior of the `mysqldump` command. Users must have the `PROCESS` privilege to access the `INFORMATION_SCHEMA.FILES` table. To run the `mysqldump` command without any changes, grant the `PROCESS` privilege to the database

user that the `mysqldump` command connects to. You can also run the `mysqldump` command with the `--no-tablespaces` option. With that option, the `mysqldump` output doesn't include any `CREATE LOGFILE GROUP` or `CREATE TABLESPACE` statements. In that case, the `mysqldump` command doesn't access the `INFORMATION_SCHEMA.FILES` table, and you don't need to grant the `PROCESS` permission.

Availability improvements:

- Fixed an issue that might cause a client session to hang when the database engine encounters an error while reading from or writing to the network.
- Fixed a memory leak in dynamic resizing feature, introduced in 2.09.0.

Global databases:

- Fixed multiple issues where a global database secondary Region's replicas might restart when upgraded to release 2.09.0 while the primary Region writer was on an older release version.

Integration of MySQL community edition bug fixes

- **Replication:** Interleaved transactions could sometimes deadlock the slave applier when the transaction isolation level was set to REPEATABLE READ. (Bug #25040331)
- For a table having a TIMESTAMP or DATETIME column having a default of CURRENT_TIMESTAMP, the column could be initialized to `0000-00-00 00:00:00` if the table had a `BEFORE INSERT` trigger. (Bug #25209512, Bug #84077)
- For an INSERT statement for which the `VALUES` list produced values for the second or later row using a subquery containing a join, the server could exit after failing to resolve the required privileges. (Bug #23762382)

Comparison with Aurora MySQL version 1

The following Amazon Aurora MySQL features are supported in Aurora MySQL Version 1 (compatible with MySQL 5.6), but these features are currently not supported in Aurora MySQL Version 2 (compatible with MySQL 5.7).

- Asynchronous key prefetch (AKP). For more information, see Optimizing Aurora indexed join queries with asynchronous key prefetch in the *Amazon Aurora User Guide*.

- Hash joins. For more information, see Optimizing large Aurora MySQL join queries with hash joins in the *Amazon Aurora User Guide*.
- Native functions for synchronously invoking AWS Lambda functions. For more information, see Invoking a Lambda function with an Aurora MySQL native function in the *Amazon Aurora User Guide*.
- Scan batching. For more information, see Aurora MySQL database engine updates 2017-12-11 (version 1.16) (Deprecated).
- Migrating data from MySQL using an Amazon S3 bucket. For more information, see Migrating data from MySQL by using an Amazon S3 bucket in the *Amazon Aurora User Guide*.

MySQL 5.7 compatibility

This Aurora MySQL version is wire-compatible with MySQL 5.7 and includes features such as JSON support, spatial indexes, and generated columns. Aurora MySQL uses a native implementation of spatial indexing using z-order curves to deliver >20x better write performance and >10x better read performance than MySQL 5.7 for spatial datasets.

This Aurora MySQL version does not currently support the following MySQL 5.7 features:

- Group replication plugin
- Increased page size
- InnoDB buffer pool loading at startup
- InnoDB full-text parser plugin
- Multisource replication
- Online buffer pool resizing
- Password validation plugin
- Query rewrite plugins
- Replication filtering
- The `CREATE TABLESPACE` SQL statement

Aurora MySQL database engine updates 2020-09-17 (version 2.09.0) (Deprecated)

Version: 2.09.0

Aurora MySQL 2.09.0 is generally available. Aurora MySQL 2.x versions are compatible with MySQL 5.7 and Aurora MySQL 1.x versions are compatible with MySQL 5.6.

Currently supported Aurora MySQL releases are 1.19.5, 1.19.6, 1.22.*, 1.23.*, 2.04.*, 2.07.*, 2.08.*, 2.09.*, 2.10.*, 3.01.* and 3.02.*.

You can restore a snapshot from Aurora MySQL 1.23.* into Aurora MySQL 2.09.0. You also have the option to upgrade existing Aurora MySQL 2.* database clusters to Aurora MySQL 2.09.0. You can't upgrade an existing Aurora MySQL 1.23.* cluster directly to 2.09.0; however, you can restore its snapshot to Aurora MySQL 2.09.0.

Important

The improvements to Aurora storage in this version limit the available upgrade paths from Aurora MySQL 1.* to Aurora MySQL 2.09. When you upgrade an Aurora MySQL 1.* cluster to 2.09, you must upgrade from Aurora MySQL 1.23.

To create a cluster with an older version of Aurora MySQL, specify the engine version through the AWS Management Console, the AWS CLI, or the RDS API.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Note

For information on how to upgrade your Aurora MySQL database cluster, see Upgrading the minor version or patch level of an Aurora MySQL DB cluster in the *Amazon Aurora User Guide*.

Improvements

New features:

- With this release, you can create Amazon Aurora MySQL database instances with up to 128 tebibytes (TiB) of storage. The new storage limit is an increase from the prior 64 TiB. The 128 TiB

storage size supports larger databases. This capability is not supported on small instances sizes (db.t2 or db.t3). A single tablespace cannot grow beyond 64 TiB due to InnoDB limitations with 16 KB page size.

Aurora alerts you when the cluster volume size is near 128 TiB, so that you can take action prior to hitting the size limit. The alerts appear in the mysql log and RDS Events in the AWS Management Console.

- You can now turn parallel query on or off for an existing cluster by changing the value of the DB cluster parameter `aurora_parallel_query`. You don't need to use the `parallelquery` setting for the `--engine-mode` parameter when creating the cluster.

  Parallel query is now expanded to be available in all regions where Aurora MySQL is available.

  There are a number of other functionality enhancements and changes to the procedures for upgrading and enabling parallel query in an Aurora cluster. For more information, see Working with parallel query for Amazon Aurora MySQL in the *Amazon Aurora User Guide*.
- Aurora dynamically resizes your cluster storage space. With dynamic resizing, the storage space for your Aurora DB cluster automatically decreases when you remove data from the DB cluster. For more information, see Storage scaling in the *Amazon Aurora User Guide*.

  > **Note**
  >
  > The dynamic resizing feature is being deployed in phases to the AWS Regions where Aurora is available. Depending on the Region where your cluster is, this feature might not be available yet. For more information, see the What's New announcement.

High priority fixes:

- Backport of Community Bug #27659490: SELECT USING DYNAMIC RANGE AND INDEX MERGE USE TOO MUCH MEMORY (OOM)
- Bug #26881508: MYSQL #1: DISABLE_ABORT_ON_ERROR IN AUTH_COMMON.H
- Backport of Community Bug #24437124: POSSIBLE BUFFER OVERFLOW ON CREATE TABLE
- Backport of Bug #27158030: INNODB ONLINE ALTER CRASHES WITH CONCURRENT DML
- Bug #29770705: SERVER CRASHED WHILE EXECUTING SELECT WITH SPECIFIC WHERE CLAUSE
- Backport of BUG #26502135: MYSQLD SEGFAULTS IN MDL_CONTEXT::TRY_ACQUIRE_LOCK_IMPL

- Backport of Bug #26935001: ALTER TABLE AUTO_INCREMENT TRIES TO READ INDEX FROM DISCARDED TABLESPACE
- Bug #28491099: [FATAL] MEMORY BLOCK IS INVALID | INNODB: ASSERTION FAILURE: UT0UT.CC:670
- Bug #30499288: GCC 9.2.1 REPORTS A NEW WARNING FOR OS_FILE_GET_PARENT_DIR
- Bug #29952565 where MYSQLD GOT SIGNAL 11 WHILE EXECUTING A QUERY(UNION + ORDER BY + SUB-QUERY)
- Bug #30628268: OUT OF MEMORY CRASH
- Bug #30441969: BUG #29723340: MYSQL SERVER CRASH AFTER SQL QUERY WITH DATA ?AST
- Bug #30569003: 5.7 REPLICATION BREAKAGE WITH SYNTAX ERROR WITH GRANT MANAGEMENT
- Bug #29915479: RUNNING COM_REGISTER_SLAVE WITHOUT COM_BINLOG_DUMP CAN RESULTS IN SERVER EXIT
- Bug #30569003: 5.7 REPLICATION BREAKAGE WITH SYNTAX ERROR WITH GRANT MANAGEMENT
- Bug #29915479: RUNNING COM_REGISTER_SLAVE WITHOUT COM_BINLOG_DUMP CAN RESULTS IN SERVER EXIT
- Bug #20712046: SHOW PROCESSLIST AND PERFORMANCE_SCHEMA TABLES DO NOT MASK PASSWORD FROM QUERY
- Backport bug #18898433: EXTREMELY SLOW PERFORMANCE WITH OUTER JOINS AND JOIN BUFFER (fixed in 5.7.21). Queries with many left joins were slow if join buffering was used (for example, using the block nested loop algorithm). (Bug #18898433, Bug #72854)"
- Backport bug #26402045: MYSQLD CRASHES ON QUERY (fixed in MySQL 5.7.23). Certain cases of subquery materialization could cause a server exit. These queries now produce an error suggesting that materialization be disabled. (Bug #26402045)
- [Backport from MySQL] users other than rdsadmin is disallowed to update pfs table in the reader replica.
- Fix the issue where the customer can not update the perfschema in the reader replica
- Bug #26666274: INFINITE LOOP IN PERFORMANCE SCHEMA BUFFER CONTAINER
- Bug #26997096: relay_log_space value is not updated in a synchronized manner so that its value sometimes much higher than the actual disk space used by relay logs.
- BUG #25082593: FOREIGN KEY VALIDATION DOESN'T NEED TO ACQUIRE GAP LOCK IN READ COMMITTED

- CVE-2019-2731
- CVE-2018-2645
- CVE-2019-2581
- CVE-2018-2787
- CVE-2019-2482
- CVE-2018-2640
- CVE-2018-2784
- CVE-2019-2628
- CVE-2019-2911
- CVE-2019-2628
- CVE-2018-3284
- CVE-2018-3065
- CVE-2019-2537
- CVE-2019-2948
- CVE-2019-2434
- CVE-2019-2420

Availability improvements:

- Enable lock manager ABA fix by default.
- Fixed an issue in the lock manager where a race condition can cause a lock to be shared by two transactions, causing the database to restart.
- Fixed an issue when creating a temporary table with compressed row format might result in a restart.
- Fix default value of `table_open_cache` on 16XL and 24XL instances which could cause repeated failovers and high CPU utilization on large instances classes (R4/R5-16XL, R5-12XL, R5-24XL). This impacted 2.07.x.
- Fixed an issue where restoring a cluster from Amazon S3 to Aurora MySQL version 2.08.0 takes longer than expected when the S3 backup didn't include the `mysql.host` table.
- Fixed an issue which might cause repeated failovers due to updates of virtual columns with secondary indexes.

- Fixed an issue related to transaction lock memory management with long-running write transactions resulting in a database restart.
- Fixed multiple issues where the engine might crash during zero-downtime patching while checking for safe point for patching.
- Fixed an issue to skip redo logging for temporary tables, which was previously causing a crash.
- Fixed a race condition in the lock manager between killing connection/query and the session killed.
- Fixed an issue where the database could crash if it is a binlog replica and receives a DDL event over the MySQL `time_zone` table.

Global databases:

- MySQL `INFORMATION_SCHEMA.REPLICA_HOST_STATUS` view in a secondary Region now shows the entries for the replicas belonging to that Region.
- Fixed unexpected query failures that could occur in a Global DB secondary Region after temporary network connectivity issues between the primary and secondary Regions.
-

Parallel query:

- Fixed the `EXPLAIN` plan for a Parallel Query query, which is incorrect for a simple single-table query.
- Fixed self-deadlatch that may occur when Parallel Query is enabled.

General improvements:

- Export to S3 now supports the `ENCRYPTION` keyword.
- The `aurora_binlog_replication_max_yield_seconds` parameter now has a max value of 36,000. The previous maximum accepted value was 45. This parameter works only when the parameter `aurora_binlog_use_large_read_buffer` is set to 1.
- Changed the behavior to map `MIXED` `binlog_format` to `ROW` instead of `STATEMENT` when executing `LOAD DATA FROM INFILE | S3`.

- Fixed an issue where a binlog replica connected to an Aurora MySQL binlog primary might show incomplete data when the primary executed `LOAD DATA FROM S3` and `binlog_format` is set to `STATEMENT`.
- Increased maximum allowable length for audit system variables `server_audit_incl_users` and `server_audit_excl_users` from 1024 bytes to 2000 bytes.
- Fixed an issue where users may lose access to the database when lowering the `max_connections` parameter in the parameter group when the current connections is greater than the value being set.
- Fixed an issue in Data Activity Streams where a single quote and backslash were not escaped properly.

Integration of MySQL community edition bug fixes

- Bug #27659490: SELECT USING DYNAMIC RANGE AND INDEX MERGE USE TOO MUCH MEMORY(OOM)
- Bug #26881508: MYSQL #1: DISABLE_ABORT_ON_ERROR IN AUTH_COMMON.H
- Bug #24437124: POSSIBLE BUFFER OVERFLOW ON CREATE TABLE
- Bug #27158030: INNODB ONLINE ALTER CRASHES WITH CONCURRENT DML
- Bug #29770705: SERVER CRASHED WHILE EXECUTING SELECT WITH SPECIFIC WHERE CLAUSE
- Bug #26502135: MYSQLD SEGFAULTS IN MDL_CONTEXT::TRY_ACQUIRE_LOCK_IMPL
- Bug #26935001: ALTER TABLE AUTO_INCREMENT TRIES TO READ INDEX FROM DISCARDED TABLESPACE
- Bug #28491099: [FATAL] MEMORY BLOCK IS INVALID | INNODB: ASSERTION FAILURE: UT0UT.CC:670
- Bug #30499288: GCC 9.2.1 REPORTS A NEW WARNING FOR OS_FILE_GET_PARENT_DIR
- Bug #29952565: where MYSQLD GOT SIGNAL 11 WHILE EXECUTING A QUERY(UNION + ORDER BY + SUB-QUERY)
- Bug #30628268: OUT OF MEMORY CRASH
- Bug #30441969: BUG #29723340: MYSQL SERVER CRASH AFTER SQL QUERY WITH DATA ?AST
- Bug #30569003: 5.7 REPLICATION BREAKAGE WITH SYNTAX ERROR WITH GRANT MANAGEMENT
- Bug #29915479: RUNNING COM_REGISTER_SLAVE WITHOUT COM_BINLOG_DUMP CAN RESULTS IN SERVER EXIT

- Bug #30569003: 5.7 REPLICATION BREAKAGE WITH SYNTAX ERROR WITH GRANT MANAGEMENT
- Bug #29915479: RUNNING COM_REGISTER_SLAVE WITHOUT COM_BINLOG_DUMP CAN RESULTS IN SERVER EXIT
- Bug #20712046: SHOW PROCESSLIST AND PERFORMANCE_SCHEMA TABLES DO NOT MASK PASSWORD FROM QUERY
- Bug #18898433: EXTREMELY SLOW PERFORMANCE WITH OUTER JOINS AND JOIN BUFFER (fixed in 5.7.21)
- Bug #26402045: MYSQLD CRASHES ON QUERY (fixed in MySQL 5.7.23)
- Bug #23103937: PS_TRUNCATE_ALL_TABLES() DOES NOT WORK IN SUPER_READ_ONLY MODE
- Bug #26666274: INFINITE LOOP IN PERFORMANCE SCHEMA BUFFER CONTAINER
- Bug #26997096: relay_log_space value is not updated in a synchronized manner so that its value sometimes much higher than the actual disk space used by relay logs. (https://github.com/mysql/mysql-server/commit/78f25d2809ad457e81f90342239c9bc32a36cdfa)
- Bug #25082593: FOREIGN KEY VALIDATION DOESN'T NEED TO ACQUIRE GAP LOCK IN READ COMMITTED
- Bug #24764800: REPLICATION FAILING ON SLAVE WITH XAER_RMFAIL ERROR.
- Bug #81441: WARNING ABOUT LOCALHOST WHEN USING SKIP-NAME-RESOLVE.

Comparison with Aurora MySQL version 1

The following Amazon Aurora MySQL features are supported in Aurora MySQL Version 1 (compatible with MySQL 5.6), but these features are currently not supported in Aurora MySQL Version 2 (compatible with MySQL 5.7).

- Asynchronous key prefetch (AKP). For more information, see Optimizing Aurora indexed join queries with asynchronous key prefetch in the *Amazon Aurora User Guide*.
- Hash joins. For more information, see Optimizing large Aurora MySQL join queries with hash joins in the *Amazon Aurora User Guide*.
- Native functions for synchronously invoking AWS Lambda functions. For more information, see Invoking a Lambda function with an Aurora MySQL native function in the *Amazon Aurora User Guide*.
- Scan batching. For more information, see Aurora MySQL database engine updates 2017-12-11 (version 1.16) (Deprecated).

- Migrating data from MySQL using an Amazon S3 bucket. For more information, see Migrating data from MySQL by using an Amazon S3 bucket in the *Amazon Aurora User Guide*.

MySQL 5.7 compatibility

This Aurora MySQL version is wire-compatible with MySQL 5.7 and includes features such as JSON support, spatial indexes, and generated columns. Aurora MySQL uses a native implementation of spatial indexing using z-order curves to deliver >20x better write performance and >10x better read performance than MySQL 5.7 for spatial datasets.

This Aurora MySQL version does not currently support the following MySQL 5.7 features:

- Group replication plugin
- Increased page size
- InnoDB buffer pool loading at startup
- InnoDB full-text parser plugin
- Multisource replication
- Online buffer pool resizing
- Password validation plugin
- Query rewrite plugins
- Replication filtering
- The `CREATE TABLESPACE` SQL statement

Aurora MySQL database engine updates 2022-01-06 (version 2.08.4) (Deprecated)

Version: 2.08.4

Aurora MySQL 2.08.4 is generally available. Aurora MySQL 2.x versions are compatible with MySQL 5.7 and Aurora MySQL 1.x versions are compatible with MySQL 5.6.

Currently supported Aurora MySQL releases are 1.19.5, 1.19.6, 1.22.*, 1.23.*, 2.04.*, 2.07.*, 2.08.*, 2.09.*, 2.10.*, 3.01.* and 3.02.*.

You can upgrade an existing Aurora MySQL 2.* database cluster to Aurora MySQL 2.10.0. For clusters running Aurora MySQL version 1, you can upgrade an existing Aurora MySQL 1.23 or

higher cluster directly to 2.10.0. You can also restore a snapshot from any currently supported Aurora MySQL release into Aurora MySQL 2.10.0.

To create a cluster with an older version of Aurora MySQL, specify the engine version through the AWS Management Console, the AWS CLI, or the Amazon RDS API.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

> **Note**
>
> For information on how to upgrade your Aurora MySQL database cluster, see Upgrading the minor version or patch level of an Aurora MySQL DB cluster in the *Amazon Aurora User Guide*.

Improvements

Security fixes and general improvements:

- Fixed security issues related to Aurora MySQL integration with other AWS Services such as Amazon S3, Amazon ML, and AWS Lambda.
- Fixed an issue where the value of a TIMESTAMP column of an existing row is updated to the latest timestamp when all of the following conditions are satisfied: 1. a trigger exists for the table; 2. an INSERT is performed on the table that has an ON DUPLICATE KEY UPDATE clause; 3. the inserted row can cause a duplicate value violation in a UNIQUE index or PRIMARY KEY; and 4. one or more columns are of TIMESTAMP data type and have a default value of CURRENT_TIMESTAMP.
- Fixed an issue which, in rare conditions, causes a writer instance to restart when an in-memory data-integrity check fails.
- Fixed an issue with parallel query which could cause the database to restart when executing SQL statements with a LIMIT clause.

Comparison with Aurora MySQL version 1

The following Amazon Aurora MySQL features are supported in Aurora MySQL Version 1 (compatible with MySQL 5.6), but these features are currently not supported in Aurora MySQL Version 2 (compatible with MySQL 5.7).

- Asynchronous key prefetch (AKP). For more information, see Optimizing Aurora indexed join queries with asynchronous key prefetch in the *Amazon Aurora User Guide*.
- Hash joins. For more information, see Optimizing large Aurora MySQL join queries with hash joins in the *Amazon Aurora User Guide*.
- Native functions for synchronously invoking AWS Lambda functions. For more information, see Invoking a Lambda function with an Aurora MySQL native function in the *Amazon Aurora User Guide*.
- Scan batching. For more information, see Aurora MySQL database engine updates 2017-12-11 (version 1.16) (Deprecated).
- Migrating data from MySQL using an Amazon S3 bucket. For more information, see Migrating data from MySQL by using an Amazon S3 bucket in the *Amazon Aurora User Guide*.

MySQL 5.7 compatibility

This Aurora MySQL version is wire-compatible with MySQL 5.7 and includes features such as JSON support, spatial indexes, and generated columns. Aurora MySQL uses a native implementation of spatial indexing using z-order curves to deliver >20x better write performance and >10x better read performance than MySQL 5.7 for spatial datasets.

This Aurora MySQL version does not currently support the following MySQL 5.7 features:

- Group replication plugin
- Increased page size
- InnoDB buffer pool loading at startup
- InnoDB full-text parser plugin
- Multisource replication
- Online buffer pool resizing
- Password validation plugin
- Query rewrite plugins

- Replication filtering
- The CREATE TABLESPACE SQL statement

Aurora MySQL database engine updates 2020-11-12 (version 2.08.3) (Deprecated)

Version: 2.08.3

Aurora MySQL 2.08.3 is generally available. Aurora MySQL 2.x versions are compatible with MySQL 5.7 and Aurora MySQL 1.x versions are compatible with MySQL 5.6.

Currently supported Aurora MySQL releases are 1.19.5, 1.19.6, 1.22.*, 1.23.*, 2.04.*, 2.07.*, 2.08.*, 2.09.*, 2.10.*, 3.01.* and 3.02.*.

You can upgrade existing Aurora MySQL 2.* database clusters directly to Aurora MySQL 2.08.3. You can upgrade an existing Aurora MySQL 1.* cluster directly to 2.07.3 or higher and then directly upgrade to 2.08.3.

To create a cluster with an older version of Aurora MySQL, specify the engine version through the AWS Management Console, the AWS CLI, or the RDS API.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

> **Note**
>
> For information on how to upgrade your Aurora MySQL database cluster, see Upgrading the minor version or patch level of an Aurora MySQL DB cluster in the *Amazon Aurora User Guide*.

Improvements

Security fixes:

Fixes and other enhancements to fine-tune handling in a managed environment. Additional CVE fixes below:

- CVE-2020-14567
- CVE-2020-14559
- CVE-2020-14553
- CVE-2020-14547
- CVE-2020-14540
- CVE-2020-2812
- CVE-2020-2806
- CVE-2020-2780
- CVE-2020-2765
- CVE-2020-2763
- CVE-2020-2760
- CVE-2020-2579

Incompatible changes:

This version introduces a permission change that affects the behavior of the `mysqldump` command. Users must have the `PROCESS` privilege to access the `INFORMATION_SCHEMA.FILES` table. To run the `mysqldump` command without any changes, grant the `PROCESS` privilege to the database user that the `mysqldump` command connects to. You can also run the `mysqldump` command with the `--no-tablespaces` option. With that option, the `mysqldump` output doesn't include any `CREATE LOGFILE GROUP` or `CREATE TABLESPACE` statements. In that case, the `mysqldump` command doesn't access the `INFORMATION_SCHEMA.FILES` table, and you don't need to grant the `PROCESS` permission.

Integration of MySQL community edition bug fixes

- Bug #23762382 - INSERT VALUES QUERY WITH JOIN IN A SELECT CAUSES INCORRECT BEHAVIOR.
- Bug #25209512 - CURRENT_TIMESTAMP PRODUCES ZEROS IN TRIGGER.

Comparison with Aurora MySQL version 1

The following Amazon Aurora MySQL features are supported in Aurora MySQL Version 1 (compatible with MySQL 5.6), but these features are currently not supported in Aurora MySQL Version 2 (compatible with MySQL 5.7).

- Asynchronous key prefetch (AKP). For more information, see Optimizing Aurora indexed join queries with asynchronous key prefetch in the *Amazon Aurora User Guide*.
- Hash joins. For more information, see Optimizing large Aurora MySQL join queries with hash joins in the *Amazon Aurora User Guide*.
- Native functions for synchronously invoking AWS Lambda functions. For more information, see Invoking a Lambda function with an Aurora MySQL native function in the *Amazon Aurora User Guide*.
- Scan batching. For more information, see Aurora MySQL database engine updates 2017-12-11 (version 1.16) (Deprecated).
- Migrating data from MySQL using an Amazon S3 bucket. For more information, see Migrating data from MySQL by using an Amazon S3 bucket in the *Amazon Aurora User Guide*.

MySQL 5.7 compatibility

This Aurora MySQL version is wire-compatible with MySQL 5.7 and includes features such as JSON support, spatial indexes, and generated columns. Aurora MySQL uses a native implementation of spatial indexing using z-order curves to deliver >20x better write performance and >10x better read performance than MySQL 5.7 for spatial datasets.

This Aurora MySQL version does not currently support the following MySQL 5.7 features:

- Group replication plugin
- Increased page size
- InnoDB buffer pool loading at startup
- InnoDB full-text parser plugin
- Multisource replication
- Online buffer pool resizing
- Password validation plugin
- Query rewrite plugins

- Replication filtering
- The CREATE TABLESPACE SQL statement

Aurora MySQL database engine updates 2020-08-28 (version 2.08.2) (Deprecated)

Version: 2.08.2

Aurora MySQL 2.08.2 is generally available. Aurora MySQL 2.x versions are compatible with MySQL 5.7 and Aurora MySQL 1.x versions are compatible with MySQL 5.6.

Currently supported Aurora MySQL releases are 1.19.5, 1.19.6, 1.22.*, 1.23.*, 2.04.*, 2.07.*, 2.08.*, 2.09.*, 2.10.*, 3.01.* and 3.02.*.

You can restore a snapshot from a currently supported Aurora MySQL release into Aurora MySQL 2.08.2. You also have the option to upgrade existing Aurora MySQL 2.* database clusters to Aurora MySQL 2.08.2. You can't upgrade an existing Aurora MySQL 1.* cluster directly to 2.08.2; however, you can restore its snapshot to Aurora MySQL 2.08.2. For more information about restoring snapshots, see Restoring from a DB cluster snapshot in the *Amazon Aurora User Guide*.

To create a cluster with an older version of Aurora MySQL, specify the engine version through the AWS Management Console, the AWS CLI, or the RDS API.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

> **Note**
>
> For information on how to upgrade your Aurora MySQL database cluster, see Upgrading the minor version or patch level of an Aurora MySQL DB cluster in the *Amazon Aurora User Guide*.

Improvements

Critical fixes:

- Fixed an issue that might cause an unplanned outage and affect database availability.

Availability fixes:

- Fixed an issue where the Aurora MySQL database could restart if it is a binlog replica and replicates a DDL event over the mysql `time_zone` table.

Comparison with Aurora MySQL version 1

The following Amazon Aurora MySQL features are supported in Aurora MySQL Version 1 (compatible with MySQL 5.6), but these features are currently not supported in Aurora MySQL Version 2 (compatible with MySQL 5.7).

- Asynchronous key prefetch (AKP). For more information, see Optimizing Aurora indexed join queries with asynchronous key prefetch in the *Amazon Aurora User Guide*.
- Hash joins. For more information, see Optimizing large Aurora MySQL join queries with hash joins in the *Amazon Aurora User Guide*.
- Native functions for synchronously invoking AWS Lambda functions. For more information, see Invoking a Lambda function with an Aurora MySQL native function in the *Amazon Aurora User Guide*.
- Scan batching. For more information, see Aurora MySQL database engine updates 2017-12-11 (version 1.16) (Deprecated).
- Migrating data from MySQL using an Amazon S3 bucket. For more information, see Migrating data from MySQL by using an Amazon S3 bucket in the *Amazon Aurora User Guide*.

MySQL 5.7 compatibility

This Aurora MySQL version is wire-compatible with MySQL 5.7 and includes features such as JSON support, spatial indexes, and generated columns. Aurora MySQL uses a native implementation of spatial indexing using z-order curves to deliver >20x better write performance and >10x better read performance than MySQL 5.7 for spatial datasets.

This Aurora MySQL version does not currently support the following MySQL 5.7 features:

- Group replication plugin
- Increased page size
- InnoDB buffer pool loading at startup
- InnoDB full-text parser plugin

- Multisource replication
- Online buffer pool resizing
- Password validation plugin
- Query rewrite plugins
- Replication filtering
- The CREATE TABLESPACE SQL statement

Aurora MySQL database engine updates 2020-06-18 (version 2.08.1) (Deprecated)

Version: 2.08.1

Aurora MySQL 2.08.1 is generally available. Aurora MySQL 2.x versions are compatible with MySQL 5.7 and Aurora MySQL 1.x versions are compatible with MySQL 5.6.

Currently supported Aurora MySQL releases are 1.19.5, 1.19.6, 1.22.*, 1.23.*, 2.04.*, 2.07.*, 2.08.*, 2.09.*, 2.10.*, 3.01.* and 3.02.*.

You can restore a snapshot from a currently supported Aurora MySQL release into Aurora MySQL 2.08.1. You also have the option to upgrade existing Aurora MySQL 2.* database clusters to Aurora MySQL 2.08.1. You can't upgrade an existing Aurora MySQL 1.* cluster directly to 2.08.1; however, you can restore its snapshot to Aurora MySQL 2.08.1.

To create a cluster with an older version of Aurora MySQL, specify the engine version through the AWS Management Console, the AWS CLI, or the RDS API.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

> **Note**
>
> For information on how to upgrade your Aurora MySQL database cluster, see Upgrading the minor version or patch level of an Aurora MySQL DB cluster in the *Amazon Aurora User Guide*.

Improvements

New features:

- Global database write forwarding. In an Aurora global database, now you can perform certain write operations, such as DML statements, while connected to a secondary cluster. The write operations are forwarded to the primary cluster, and any changes are replicated back to the secondary clusters. For more information, see Using write forwarding in an Amazon Aurora global database in the *Amazon Aurora User Guide*.

General stability fixes:

- Fixed an issue where restoring a cluster from Amazon S3 to Aurora MySQL version 2.08.0 took longer than expected if the S3 backup didn't include the `mysql.host` table.

Comparison with Aurora MySQL version 1

The following Amazon Aurora MySQL features are supported in Aurora MySQL Version 1 (compatible with MySQL 5.6), but these features are currently not supported in Aurora MySQL Version 2 (compatible with MySQL 5.7).

- Asynchronous key prefetch (AKP). For more information, see Optimizing Aurora indexed join queries with asynchronous key prefetch in the *Amazon Aurora User Guide*.
- Hash joins. For more information, see Optimizing large Aurora MySQL join queries with hash joins in the *Amazon Aurora User Guide*.
- Native functions for synchronously invoking AWS Lambda functions. For more information, see Invoking a Lambda function with an Aurora MySQL native function in the *Amazon Aurora User Guide*.
- Scan batching. For more information, see Aurora MySQL database engine updates 2017-12-11 (version 1.16) (Deprecated).
- Migrating data from MySQL using an Amazon S3 bucket. For more information, see Migrating data from MySQL by using an Amazon S3 bucket in the *Amazon Aurora User Guide*.

MySQL 5.7 compatibility

This Aurora MySQL version is wire-compatible with MySQL 5.7 and includes features such as JSON support, spatial indexes, and generated columns. Aurora MySQL uses a native implementation of spatial indexing using z-order curves to deliver >20x better write performance and >10x better read performance than MySQL 5.7 for spatial datasets.

This Aurora MySQL version does not currently support the following MySQL 5.7 features:

- Group replication plugin
- Increased page size
- InnoDB buffer pool loading at startup
- InnoDB full-text parser plugin
- Multisource replication
- Online buffer pool resizing
- Password validation plugin
- Query rewrite plugins
- Replication filtering
- The `CREATE TABLESPACE` SQL statement

Aurora MySQL database engine updates 2020-06-02 (version 2.08.0) (Deprecated)

Version: 2.08.0

Aurora MySQL 2.08.0 is generally available. Aurora MySQL 2.x versions are compatible with MySQL 5.7 and Aurora MySQL 1.x versions are compatible with MySQL 5.6.

Currently supported Aurora MySQL releases are 1.19.5, 1.19.6, 1.22.*, 1.23.*, 2.04.*, 2.07.*, 2.08.*, 2.09.*, 2.10.*, 3.01.* and 3.02.*.

You can restore a snapshot from a currently supported Aurora MySQL release into Aurora MySQL 2.08.0. You also have the option to upgrade existing Aurora MySQL 2.* database clusters to Aurora MySQL 2.08.0. You can't upgrade an existing Aurora MySQL 1.* cluster directly to 2.08.0; however, you can restore its snapshot to Aurora MySQL 2.08.0.

To create a cluster with an older version of Aurora MySQL, specify the engine version through the AWS Management Console, the AWS CLI, or the RDS API.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

> **Note**
>
> For information on how to upgrade your Aurora MySQL database cluster, see Upgrading the minor version or patch level of an Aurora MySQL DB cluster in the *Amazon Aurora User Guide*.

Improvements

New features:

- Improved binary log (binlog) processing to reduce crash recovery time and commit time latency when very large transactions are involved.
- Launching Database Activity Streams (DAS) feature for Aurora MySQL. This feature provides a near-real-time data stream of the database activity in your relational database to help you monitor activity. For more information, see Monitoring Amazon Aurora with Database Activity Streams in the *Amazon Aurora User Guide*.
- Updated timezone files to support the latest Brazil timezone change.
- Introduced new keywords in SQL to exercise the hash join functionality for a specific table and/or inner table: `HASH_JOIN`, `HASH_JOIN_PROBING`, and `HASH_JOIN_BUILDING`. For additional details, see Aurora MySQL hints in the *Amazon Aurora User Guide*.
- Introduced join order hint support in Aurora MySQL 5.7 by backporting a MySQL 8.0 feature. The new hints are `JOIN_FIXED_ORDER`, `JOIN_ORDER`, `JOIN_PREFIX`, and `JOIN_SUFFIX`. For detailed documentation of join order hint support, see WL#9158: Join order hints.
- Aurora Machine Learning now supports user-defined functions with `MEDIUMINT` as the return type.
- The `lambda_async()` stored procedure now supports all MySQL `utf8` characters.

High priority fixes:

- Fixed an issue that could cause a reader DB instance to return incomplete results for an FTS query after the `INFORMATION_SCHEMA.INNODB_SYS_TABLES` table is queried on the writer DB instance.
- CVE-2019-5443
- CVE-2019-3822

Availability improvements:

- Fixed an issue that resulted in a database restart after a multi-query statement that accesses multiple tables or databases is executed with the query cache enabled.
- Fixed a race condition in the lock manager that resulted in a database restart or failover during transaction rollback.
- Fixed an issue that triggered database restart or failover when multiple connections are trying to update the same table with a Full-Text Search index.
- Fixed an issue that could trigger a database restart or failover during a `kill session` command. If you encounter this issue, contact AWS support to enable this fix on your instance.
- Fixed an issue that caused reader DB instance to restart during a multi-statement transaction with multiple `SELECT` statements and a heavy write workload on the writer DB instance with `AUTOCOMMIT` enabled.
- Fixed an issue that caused reader DB instance to restart after executing long-running queries while the writer DB instance is under a heavy OLTP write workload.

General improvements:

- Improved database recovery time and commit latency for long running transactions when binlog is enabled.
- Improved the algorithm to generate better statistics for estimating distinct value counts on indexed columns, including columns with skewed data distributions.
- Reduced the response time and CPU utilization of join queries that access MyISAM temporary tables and the results spill to local storage.
- Fixed an issue that prevented Aurora MySQL 5.6 snapshots with database or table names containing spaces from being restored to a new Aurora MySQL 5.7 cluster.
- Included victim transaction info when deadlock is resolved in `show engine innodb status`.

- Fixed an issue that caused connections to get stuck when clients of multiple different versions are connected to the same database and are accessing the query cache.
- Fixed a memory leak resulting from multiple invocations of the Zero-Downtime Patch (ZDP) or Zero-Downtime Restart (ZDR) workflow throughout the lifetime of a database instance.
- Fixed an error message in Zero-Downtime Patch (ZDP) or Zero-Downtime Restart (ZDR) operations wrongly stating that the last transaction was aborted if the auto-commit flag is turned off.
- Fixed an issue in Zero-Downtime Patch (ZDP) operations that could lead to a server failure error message when restoring user session variables in the new database process.
- Fixed an issue in Zero Downtime Patch (ZDP) operations that might cause intermittent database failures when there are long running queries during patching.
- Fixed an issue where queries including an Aurora Machine Learning function returned empty error messages due to an incorrectly handled error response from Machine Learning services such as Amazon SageMaker and Amazon Comprehend.
- Fixed an issue in the out-of-memory monitoring functionality that did not honor a custom value of the `table_definition_cache` parameter.
- The error message "Query execution was interrupted" is returned if an Aurora Machine Learning query is interrupted. Previously, the generic message "Internal error in processing ML request" was returned instead.
- Fixed an issue that could cause a binlog worker to experience a connection timeout when the `slave_net_timeout` parameter is less than the `aurora_binlog_replication_max_yield_seconds` parameter and there is low workload on the binlog master cluster.
- Improved monitoring of the binlog recovery progress by outputting informational messages in the error log at a frequency of one message per minute.
- Fixed an issue that could cause active transactions not to be reported by the `SHOW ENGINE INNODB STATUS` query.

Integration of MySQL community edition bug fixes

- Bug #25289359: A full-text cache lock taken when data is synchronized was not released if the full-text cache size exceeded the full-text cache size limit.
- Bug #29138644: Manually changing the system time while the MySQL server was running caused page cleaner thread delays.

- Bug #25222337: A NULL virtual column field name in a virtual index caused a server exit during a field name comparison that occurs while populating virtual columns affected by a foreign key constraint.
- Bug #25053286: Executing a stored procedure containing a query that accessed a view could allocate memory that was not freed until the session ended.
- Bug #25586773: Executing a stored procedure containing a statement that created a table from the contents of certain SELECT statements could result in a memory leak.
- Bug #28834208: During log application, after an OPTIMIZE TABLE operation, InnoDB did not populate virtual columns before checking for virtual column index updates.
- Bug #26666274: Infinite loop in performance schema buffer container due to 32-bit unsigned integer overflow.

Comparison with Aurora MySQL version 1

The following Amazon Aurora MySQL features are supported in Aurora MySQL Version 1 (compatible with MySQL 5.6), but these features are currently not supported in Aurora MySQL Version 2 (compatible with MySQL 5.7).

- Asynchronous key prefetch (AKP). For more information, see Optimizing Aurora indexed join queries with asynchronous key prefetch in the *Amazon Aurora User Guide*.
- Hash joins. For more information, see Optimizing large Aurora MySQL join queries with hash joins in the *Amazon Aurora User Guide*.
- Native functions for synchronously invoking AWS Lambda functions. For more information, see Invoking a Lambda function with an Aurora MySQL native function in the *Amazon Aurora User Guide*.
- Scan batching. For more information, see Aurora MySQL database engine updates 2017-12-11 (version 1.16) (Deprecated).
- Migrating data from MySQL using an Amazon S3 bucket. For more information, see Migrating data from MySQL by using an Amazon S3 bucket in the *Amazon Aurora User Guide*.

MySQL 5.7 compatibility

This Aurora MySQL version is wire-compatible with MySQL 5.7 and includes features such as JSON support, spatial indexes, and generated columns. Aurora MySQL uses a native implementation of

spatial indexing using z-order curves to deliver >20x better write performance and >10x better read performance than MySQL 5.7 for spatial datasets.

This Aurora MySQL version does not currently support the following MySQL 5.7 features:

- Group replication plugin
- Increased page size
- InnoDB buffer pool loading at startup
- InnoDB full-text parser plugin
- Multisource replication
- Online buffer pool resizing
- Password validation plugin
- Query rewrite plugins
- Replication filtering
- The `CREATE TABLESPACE` SQL statement

Aurora MySQL database engine update 2023-08-15 (version 2.07.10, compatible with MySQL 5.7.12)

Version: 2.07.10

Aurora MySQL 2.07.10 is generally available. Aurora MySQL 2.07 versions are compatible with MySQL 5.7.12. For more information on community changes, see Changes in MySQL 5.7.12 (2016-04-11, General Availability).

> **Note**
>
> This version is designated as a long-term support (LTS) release. For more information, see Aurora MySQL long-term support (LTS) releases in the *Amazon Aurora User Guide*.

Currently supported Aurora MySQL releases are 2.07.*, 2.11.*, 3.01.*, 3.02.*, 3.03.*, and 3.04.*.

You can restore a snapshot from a currently supported Aurora MySQL release into Aurora MySQL 2.07.10. You also have the option to upgrade existing Aurora MySQL 2.* database clusters to

Aurora MySQL 2.07.10. In-place upgrade is available for Aurora MySQL 1.* clusters to Aurora MySQL 2.* (see Upgrading from Aurora MySQL 1.x to 2.x). It's also available for Aurora MySQL 2.* clusters to Aurora MySQL 3.* (see Upgrading from Aurora MySQL 2.x to 3.x).

Immediately after an in-place engine version upgrade to Aurora MySQL 2.07.10 is performed, an operating system upgrade is applied automatically to all of the affected instances on the db.r4, db.r5, db.t2, and db.t3 DB instance classes, if the instances are running an old operating system version. In a Multi-AZ DB cluster, all of the reader instances apply the operating system upgrade first. When the operating system upgrade on the first reader instance is finished, a failover occurs and the previous writer instance is upgraded.

> **Note**
>
> The operating system upgrade isn't applied automatically to Aurora global databases during major version upgrades.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

For information on how to upgrade your Aurora MySQL database cluster, see Upgrading the minor version or patch level of an Aurora MySQL DB cluster in the *Amazon Aurora User Guide*.

Improvements

Fixed security issues and CVEs listed below:

Fixes and other enhancements to fine-tune handling in a managed environment. Additional CVE fixes below:

- CVE-2023-21963
- CVE-2023-21912
- CVE-2023-0215
- CVE-2022-43551
- CVE-2022-37434
- Fixed an issue where events that were reported while processing audit log rotations might not be written to the audit log.

- Updated the default SSL ciphers used by Aurora MySQL to exclude the less secure DES-CBC3-SHA values from the SSL_CIPHER database parameter. If you encounter SSL connection issues due to the removal of the DES-CBC3-SHA cipher, please use an applicable secure cipher from the following information, Configuring cipher suites for connections to Aurora MySQL DB clusters.
- OpenSSL was upgraded to version 1.0.2zh.

General improvements:

- Added support for ECDHE-RSA SSL ciphers that use smaller key sizes for encryption.
- Fixed a memory management issue when executing queries with hash joins.

Features not supported in Aurora MySQL version 2

The following features are currently not supported in Aurora MySQL version 2 (compatible with MySQL 5.7).

- Asynchronous key prefetch (AKP).
- Scan batching. For more information, see Aurora MySQL database engine updates 2017-12-11 (version 1.16) (Deprecated).

MySQL 5.7 compatibility

This Aurora MySQL version is wire-compatible with MySQL 5.7 and includes features such as JSON support, spatial indexes, and generated columns. Aurora MySQL uses a native implementation of spatial indexing using z-order curves to deliver >20x better write performance and >10x better read performance than MySQL 5.7 for spatial datasets.

This Aurora MySQL version does not currently support the following MySQL 5.7 features:

- The `CREATE TABLESPACE` SQL statement
- Group replication plugin
- Increased page size
- InnoDB buffer pool loading at startup
- InnoDB full-text parser plugin
- Multisource replication

- Online buffer pool resizing
- Password validation plugin
- Query rewrite plugins
- Replication filtering
- X Protocol

Aurora MySQL database engine update 2023-05-04 (version 2.07.9, compatible with MySQL 5.7.12)

Version: 2.07.9

Aurora MySQL 2.07.9 is generally available. Aurora MySQL 2.07 versions are compatible with MySQL 5.7.12. For more information on community changes, see Changes in MySQL 5.7.12 (2016-04-11, General Availability).

> **Note**
>
> This version is designated as a long-term support (LTS) release. For more information, see Aurora MySQL long-term support (LTS) releases in the *Amazon Aurora User Guide*.

Currently supported Aurora MySQL releases are 2.07.*, 2.11.*, 3.01.*, 3.02.* and 3.03.*.

You can restore a snapshot from a currently supported Aurora MySQL release into Aurora MySQL 2.07.9. You also have the option to upgrade existing Aurora MySQL 2.* database clusters to Aurora MySQL 2.07.9. In-place upgrade is available for Aurora MySQL 1.* clusters to Aurora MySQL 2.* (see Upgrading from Aurora MySQL 1.x to 2.x). It's also available for Aurora MySQL 2.* clusters to Aurora MySQL 3.* (see Upgrading from Aurora MySQL 2.x to 3.x).

Immediately after an in-place engine version upgrade to Aurora MySQL 2.07.9 is performed, an operating system upgrade is applied automatically to all of the affected instances on the db.r4, db.r5, db.t2, and db.t3 DB instance classes, if the instances are running an old operating system version. In a Multi-AZ DB cluster, all of the reader instances apply the operating system upgrade first. When the operating system upgrade on the first reader instance is finished, a failover occurs and the previous writer instance is upgraded.

> **Note**
>
> The operating system upgrade isn't applied automatically to Aurora global databases during major version upgrades.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

For information on how to upgrade your Aurora MySQL database cluster, see Upgrading the minor version or patch level of an Aurora MySQL DB cluster in the *Amazon Aurora User Guide*.

Improvements

Fixed security issues and CVEs listed below:

Fixes and other enhancements to fine-tune handling in a managed environment. Additional CVE fixes below:

- CVE-2022-32221

Availability improvements:

- Fixed an issue where the Advanced Audit log rotation may reduce the freeable memory, which could lead to the DB instance restarting.
- Fixed an issue which can occur during database restarts and results in the database not starting up successfully for an extended duration.

General improvements:

- Fixed an issue which, in rare conditions, can cause the instances to restart when the database volume grows to a multiple of 160GB.

Features not supported in Aurora MySQL version 2

The following features are currently not supported in Aurora MySQL version 2 (compatible with MySQL 5.7).

- Asynchronous key prefetch (AKP).
- Scan batching. For more information, see Aurora MySQL database engine updates 2017-12-11 (version 1.16) (Deprecated).

MySQL 5.7 compatibility

This Aurora MySQL version is wire-compatible with MySQL 5.7 and includes features such as JSON support, spatial indexes, and generated columns. Aurora MySQL uses a native implementation of spatial indexing using z-order curves to deliver >20x better write performance and >10x better read performance than MySQL 5.7 for spatial datasets.

This Aurora MySQL version does not currently support the following MySQL 5.7 features:

- Group replication plugin
- Increased page size
- InnoDB buffer pool loading at startup
- InnoDB full-text parser plugin
- Multisource replication
- Online buffer pool resizing
- Password validation plugin
- Query rewrite plugins
- Replication filtering
- The `CREATE TABLESPACE` SQL statement

Aurora MySQL database engine updates 2022-06-16 (version 2.07.8) (Deprecated)

Version: 2.07.8

Aurora MySQL 2.07.8 is generally available. Aurora MySQL 2.* versions are compatible with MySQL 5.7 and Aurora MySQL 1.* versions are compatible with MySQL 5.6.

> **Note**
>
> This version is designated as a long-term support (LTS) release. For more information, see Aurora MySQL long-term support (LTS) releases in the *Amazon Aurora User Guide*.

Currently supported Aurora MySQL releases are 1.19.5, 1.19.6, 1.22.*, 1.23.*, 2.04.*, 2.07.*, 2.08.*, 2.09.*, 2.10.*, 3.01.* and 3.02.*.

You can restore a snapshot from a currently supported Aurora MySQL release into Aurora MySQL 2.07.8. You also have the option to upgrade existing Aurora MySQL 2.* database clusters to Aurora MySQL 2.07.8. In-place upgrade is available for Aurora MySQL 1.* clusters to Aurora MySQL 2.* (see Upgrading from Aurora MySQL 1.x to 2.x). It's also available for Aurora MySQL 2.* clusters to Aurora MySQL 3.* (see Upgrading from Aurora MySQL 2.x to 3.x).

To create a cluster with an older version of Aurora MySQL, specify the engine version through the AWS Management Console, the AWS CLI, or the RDS API.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

> **Note**
>
> For information on how to upgrade your Aurora MySQL database cluster, see Upgrading the minor version or patch level of an Aurora MySQL DB cluster in the *Amazon Aurora User Guide*.

Improvements

Security fixes:

Fixes and other enhancements to fine-tune handling in a managed environment. Additional CVE fixes below:

- CVE-2022-21245
- CVE-2021-36222
- CVE-2021-22926

General improvements:

- Fixed an issue which, in rare cases, causes the database server to restart when the deadlock detector thread gets stuck because of a race condition.

Integration of MySQL community edition bug fixes

- When an UPDATE required a temporary table having a primary key larger than 1024 bytes and that table was created using InnoDB, the server could exit. (Bug #25153670)

Comparison with Aurora MySQL version 1

The following Amazon Aurora MySQL features are supported in Aurora MySQL Version 1 (compatible with MySQL 5.6), but these features are currently not supported in Aurora MySQL Version 2 (compatible with MySQL 5.7).

- Asynchronous key prefetch (AKP). For more information, see Optimizing Aurora indexed join queries with asynchronous key prefetch in the *Amazon Aurora User Guide*.
- Hash joins. For more information, see Optimizing large Aurora MySQL join queries with hash joins in the *Amazon Aurora User Guide*.
- Native functions for synchronously invoking AWS Lambda functions. For more information, see Invoking a Lambda function with an Aurora MySQL native function in the *Amazon Aurora User Guide*.
- Scan batching. For more information, see Aurora MySQL database engine updates 2017-12-11 (version 1.16) (Deprecated).
- Migrating data from MySQL using an Amazon S3 bucket. For more information, see Migrating data from MySQL by using an Amazon S3 bucket in the *Amazon Aurora User Guide*.

MySQL 5.7 compatibility

This Aurora MySQL version is wire-compatible with MySQL 5.7 and includes features such as JSON support, spatial indexes, and generated columns. Aurora MySQL uses a native implementation of spatial indexing using z-order curves to deliver >20x better write performance and >10x better read performance than MySQL 5.7 for spatial datasets.

This Aurora MySQL version does not currently support the following MySQL 5.7 features:

- Group replication plugin
- Increased page size
- InnoDB buffer pool loading at startup
- InnoDB full-text parser plugin
- Multisource replication
- Online buffer pool resizing
- Password validation plugin
- Query rewrite plugins
- Replication filtering
- The `CREATE TABLESPACE` SQL statement

Aurora MySQL database engine updates 2021-11-24 (version 2.07.7) (Deprecated)

Version: 2.07.7

Aurora MySQL 2.07.7 is generally available. Aurora MySQL 2.* versions are compatible with MySQL 5.7 and Aurora MySQL 1.* versions are compatible with MySQL 5.6.

Currently supported Aurora MySQL releases are 1.19.5, 1.19.6, 1.22.*, 1.23.*, 2.04.*, 2.07.*, 2.08.*, 2.09.*, 2.10.*, 3.01.* and 3.02.*.

You can restore a snapshot from a currently supported Aurora MySQL release into Aurora MySQL 2.07.7. You also have the option to upgrade existing Aurora MySQL 2.* database clusters to Aurora MySQL 2.07.7. You can't upgrade an existing Aurora MySQL 1.* cluster directly to 2.07.7; however, you can restore its snapshot to Aurora MySQL 2.07.7.

To create a cluster with an older version of Aurora MySQL, specify the engine version through the AWS Management Console, the AWS CLI, or the RDS API.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

> **Note**
>
> For information on how to upgrade your Aurora MySQL database cluster, see Upgrading the minor version or patch level of an Aurora MySQL DB cluster in the *Amazon Aurora User Guide*.

Improvements

Security fixes:

Fixes and other enhancements to fine-tune handling in a managed environment. Additional CVE fixes below:

- CVE-2019-17543
- CVE-2019-2960

General improvements:

- Fixed security issues related to Aurora MySQL integration with other AWS Services such as Amazon S3, Amazon ML, Lambda.
- Fixed an issue with incorrect reporting of an Aurora replication lag.
- Fixed an issue which can cause a database instance restart to fail when the database has a large number of user and privilege combinations.
- Fixed an issue which can cause general_log and slow_log tables to become inaccessible after in-place major version upgrade from Aurora MySQL 1.x (based on MySQL 5.6) to Aurora MySQL 2.x (based on MySQL 5.7).
- Fixed an issue which, in rare conditions, can cause a reader instance to restart due to an incorrect check processing.
- Fixed an issue which, in rare conditions, shows the "Database Load" chart in Performance Insights (PI) sessions as actively using CPU even though the sessions have finished processing and are idle.
- Fixed an issue with parallel query which can cause the database to restart when executing SQL statements with a LIMIT clause.
- Fixed an issue where the value of a TIMESTAMP column of an existing row is updated to the latest timestamp when all of the following conditions are satisfied: 1. A trigger exists for

the table; 2. an INSERT is performed on the table that has an ON DUPLICATE KEY UPDATE clause; 3. the inserted row can cause a duplicate value violation in a UNIQUE index or PRIMARY KEY; and, 4. one or more columns are of TIMESTAMP data type and have a default value of CURRENT_TIMESTAMP.

- Fixed an issue which, in rare conditions, can cause the database instance to restart when using XA transactions in READ COMMITTED isolation level.

Comparison with Aurora MySQL version 1

The following Amazon Aurora MySQL features are supported in Aurora MySQL Version 1 (compatible with MySQL 5.6), but these features are currently not supported in Aurora MySQL Version 2 (compatible with MySQL 5.7).

- Asynchronous key prefetch (AKP). For more information, see Optimizing Aurora indexed join queries with asynchronous key prefetch in the *Amazon Aurora User Guide*.
- Hash joins. For more information, see Optimizing large Aurora MySQL join queries with hash joins in the *Amazon Aurora User Guide*.
- Native functions for synchronously invoking AWS Lambda functions. For more information, see Invoking a Lambda function with an Aurora MySQL native function in the *Amazon Aurora User Guide*.
- Scan batching. For more information, see Aurora MySQL database engine updates 2017-12-11 (version 1.16) (Deprecated).
- Migrating data from MySQL using an Amazon S3 bucket. For more information, see Migrating data from MySQL by using an Amazon S3 bucket in the *Amazon Aurora User Guide*.

MySQL 5.7 compatibility

This Aurora MySQL version is wire-compatible with MySQL 5.7 and includes features such as JSON support, spatial indexes, and generated columns. Aurora MySQL uses a native implementation of spatial indexing using z-order curves to deliver >20x better write performance and >10x better read performance than MySQL 5.7 for spatial datasets.

This Aurora MySQL version does not currently support the following MySQL 5.7 features:

- Group replication plugin
- Increased page size

- InnoDB buffer pool loading at startup
- InnoDB full-text parser plugin
- Multisource replication
- Online buffer pool resizing
- Password validation plugin
- Query rewrite plugins
- Replication filtering
- The `CREATE TABLESPACE` SQL statement

Aurora MySQL database engine updates 2021-09-02 (version 2.07.6) (Deprecated)

Version: 2.07.6

Aurora MySQL 2.07.6 is generally available. Aurora MySQL 2.* versions are compatible with MySQL 5.7 and Aurora MySQL 1.* versions are compatible with MySQL 5.6.

Currently supported Aurora MySQL releases are 1.19.5, 1.19.6, 1.22.*, 1.23.*, 2.04.*, 2.07.*, 2.08.*, 2.09.*, 2.10.*, 3.01.* and 3.02.*.

You can restore a snapshot from a currently supported Aurora MySQL release into Aurora MySQL 2.07.6. You also have the option to upgrade existing Aurora MySQL 2.* database clusters to Aurora MySQL 2.07.6. You can't upgrade an existing Aurora MySQL 1.* cluster directly to 2.07.6; however, you can restore its snapshot to Aurora MySQL 2.07.6.

To create a cluster with an older version of Aurora MySQL, please specify the engine version through the AWS Management Console, the AWS CLI, or the RDS API.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Integration of MySQL community edition bug fixes

- INSERTING 64K SIZE RECORDS TAKE TOO MUCH TIME. (Bug#23031146)

Comparison with Aurora MySQL version 1

The following Amazon Aurora MySQL features are supported in Aurora MySQL Version 1 (compatible with MySQL 5.6), but these features are currently not supported in Aurora MySQL Version 2 (compatible with MySQL 5.7).

- Asynchronous key prefetch (AKP). For more information, see Optimizing Aurora indexed join queries with asynchronous key prefetch in the *Amazon Aurora User Guide*.
- Hash joins. For more information, see Optimizing large Aurora MySQL join queries with hash joins in the *Amazon Aurora User Guide*.
- Native functions for synchronously invoking AWS Lambda functions. For more information, see Invoking a Lambda function with an Aurora MySQL native function in the *Amazon Aurora User Guide*.
- Scan batching. For more information, see Aurora MySQL database engine updates 2017-12-11 (version 1.16) (Deprecated).
- Migrating data from MySQL using an Amazon S3 bucket. For more information, see Migrating data from MySQL by using an Amazon S3 bucket in the *Amazon Aurora User Guide*.

MySQL 5.7 compatibility

This Aurora MySQL version is wire-compatible with MySQL 5.7 and includes features such as JSON support, spatial indexes, and generated columns. Aurora MySQL uses a native implementation of spatial indexing using z-order curves to deliver >20x better write performance and >10x better read performance than MySQL 5.7 for spatial datasets.

This Aurora MySQL version does not currently support the following MySQL 5.7 features:

- Group replication plugin
- Increased page size
- InnoDB buffer pool loading at startup
- InnoDB full-text parser plugin
- Multisource replication
- Online buffer pool resizing
- Password validation plugin
- Query rewrite plugins

- Replication filtering
- The CREATE TABLESPACE SQL statement

Aurora MySQL database engine updates 2021-07-06 (version 2.07.5) (Deprecated)

Version: 2.07.5

Aurora MySQL 2.07.5 is generally available. Aurora MySQL 2.* versions are compatible with MySQL 5.7 and Aurora MySQL 1.* versions are compatible with MySQL 5.6.

Currently supported Aurora MySQL releases are 1.19.5, 1.19.6, 1.22.*, 1.23.*, 2.04.*, 2.07.*, 2.08.*, 2.09.*, 2.10.*, 3.01.* and 3.02.*.

You can restore a snapshot from a currently supported Aurora MySQL release into Aurora MySQL 2.07.5. You also have the option to upgrade existing Aurora MySQL 2.* database clusters to Aurora MySQL 2.07.5. You can't upgrade an existing Aurora MySQL 1.* cluster directly to 2.07.5; however, you can restore its snapshot to Aurora MySQL 2.07.5.

To create a cluster with an older version of Aurora MySQL, please specify the engine version through the AWS Management Console, the AWS CLI, or the RDS API.

> **Note**
>
> This version is designated as a long-term support (LTS) release. For more information, see Aurora MySQL long-term support (LTS) releases in the *Amazon Aurora User Guide*.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Improvements

Availability improvements:

- Fixed an issue that user-level locks are not allowed on an Aurora Replica.

- Fixed an issue that could cause a restart of a database when using XA transactions in `READ COMMITTED` isolation level.
- Extended maximum allowable length to 2000 for the `server_audit_incl_users` and `server_audit_excl_users` global parameters.

Comparison with Aurora MySQL version 1

The following Amazon Aurora MySQL features are supported in Aurora MySQL Version 1 (compatible with MySQL 5.6), but these features are currently not supported in Aurora MySQL Version 2 (compatible with MySQL 5.7).

- Asynchronous key prefetch (AKP). For more information, see Optimizing Aurora indexed join queries with asynchronous key prefetch in the *Amazon Aurora User Guide*.
- Hash joins. For more information, see Optimizing large Aurora MySQL join queries with hash joins in the *Amazon Aurora User Guide*.
- Native functions for synchronously invoking AWS Lambda functions. For more information, see Invoking a Lambda function with an Aurora MySQL native function in the *Amazon Aurora User Guide*.
- Scan batching. For more information, see Aurora MySQL database engine updates 2017-12-11 (version 1.16) (Deprecated).
- Migrating data from MySQL using an Amazon S3 bucket. For more information, see Migrating data from MySQL by using an Amazon S3 bucket in the *Amazon Aurora User Guide*.

MySQL 5.7 compatibility

This Aurora MySQL version is wire-compatible with MySQL 5.7 and includes features such as JSON support, spatial indexes, and generated columns. Aurora MySQL uses a native implementation of spatial indexing using z-order curves to deliver >20x better write performance and >10x better read performance than MySQL 5.7 for spatial datasets.

This Aurora MySQL version does not currently support the following MySQL 5.7 features:

- Group replication plugin
- Increased page size
- InnoDB buffer pool loading at startup

- InnoDB full-text parser plugin
- Multisource replication
- Online buffer pool resizing
- Password validation plugin
- Query rewrite plugins
- Replication filtering
- The CREATE TABLESPACE SQL statement

Aurora MySQL database engine updates 2021-03-04 (version 2.07.4) (Deprecated)

Version: 2.07.4

Aurora MySQL 2.07.4 is generally available. Aurora MySQL 2.* versions are compatible with MySQL 5.7 and Aurora MySQL 1.* versions are compatible with MySQL 5.6.

Currently supported Aurora MySQL releases are 1.19.5, 1.19.6, 1.22.*, 1.23.*, 2.04.*, 2.07.*, 2.08.*, 2.09.*, 2.10.*, 3.01.* and 3.02.*.

You can restore a snapshot from a currently supported Aurora MySQL release into Aurora MySQL 2.07.4. You also have the option to upgrade existing Aurora MySQL 2.* database clusters to Aurora MySQL 2.07.4. You can't upgrade an existing Aurora MySQL 1.* cluster directly to 2.07.4; however, you can restore its snapshot to Aurora MySQL 2.07.4.

To create a cluster with an older version of Aurora MySQL, please specify the engine version through the AWS Management Console, the AWS CLI, or the RDS API.

> **Note**
>
> This version is designated as a long-term support (LTS) release. For more information, see Aurora MySQL long-term support (LTS) releases in the *Amazon Aurora User Guide*.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Improvements

Security fixes:

- CVE-2020-14812
- CVE-2020-14793
- CVE-2020-14790
- CVE-2020-14775
- CVE-2020-14769
- CVE-2020-14765
- CVE-2020-14760
- CVE-2020-14672
- CVE-2020-1971

Availability improvements:

- Fixed an issue that could cause a client to hang in case of a network error while reading or writing a network packet.
- Improved engine restart times in some cases after interrupted DDL.
- Fixed an issue where a DDL or DML could cause engine restart during a page prefetch request.
- Fixed an issue where a replica could restart while performing a reverse scan of a table/index on an Aurora Read Replica.
- Fixed an issue in clone cluster operation that could cause the clone to take longer.
- Fixed an issue that could cause a restart of a database when using parallel query optimization for geospatial columns.
- Fixed an issue that caused a binlog replica to stop with an `HA_ERR_KEY_NOT_FOUND` error.

Integration of MySQL community edition bug fixes

- Fixed an issue in the Full-text ngram parser when dealing with tokens containing ' ' (space), '%', or ','. Customers should rebuild their FTS indexes if using ngram parser. (Bug #25873310)
- Fixed an issue that could cause engine restart during query execution with nested SQL views. (Bug #27214153, Bug #26864199)

Comparison with Aurora MySQL version 1

The following Amazon Aurora MySQL features are supported in Aurora MySQL Version 1 (compatible with MySQL 5.6), but these features are currently not supported in Aurora MySQL Version 2 (compatible with MySQL 5.7).

- Asynchronous key prefetch (AKP). For more information, see Optimizing Aurora indexed join queries with asynchronous key prefetch in the *Amazon Aurora User Guide*.
- Hash joins. For more information, see Optimizing large Aurora MySQL join queries with hash joins in the *Amazon Aurora User Guide*.
- Native functions for synchronously invoking AWS Lambda functions. For more information, see Invoking a Lambda function with an Aurora MySQL native function in the *Amazon Aurora User Guide*.
- Scan batching. For more information, see Aurora MySQL database engine updates 2017-12-11 (version 1.16) (Deprecated).
- Migrating data from MySQL using an Amazon S3 bucket. For more information, see Migrating data from MySQL by using an Amazon S3 bucket in the *Amazon Aurora User Guide*.

MySQL 5.7 compatibility

This Aurora MySQL version is wire-compatible with MySQL 5.7 and includes features such as JSON support, spatial indexes, and generated columns. Aurora MySQL uses a native implementation of spatial indexing using z-order curves to deliver >20x better write performance and >10x better read performance than MySQL 5.7 for spatial datasets.

This Aurora MySQL version does not currently support the following MySQL 5.7 features:

- Group replication plugin
- Increased page size
- InnoDB buffer pool loading at startup
- InnoDB full-text parser plugin
- Multisource replication
- Online buffer pool resizing
- Password validation plugin
- Query rewrite plugins

- Replication filtering
- The CREATE TABLESPACE SQL statement

Aurora MySQL database engine updates 2020-11-10 (version 2.07.3) (Deprecated)

Version: 2.07.3

Aurora MySQL 2.07.3 is generally available. Aurora MySQL 2.* versions are compatible with MySQL 5.7 and Aurora MySQL 1.* versions are compatible with MySQL 5.6.

Currently supported Aurora MySQL releases are 1.19.5, 1.19.6, 1.22.*, 1.23.*, 2.04.*, 2.07.*, 2.08.*, 2.09.*, 2.10.*, 3.01.* and 3.02.*.

You can restore a snapshot from a currently supported Aurora MySQL release into Aurora MySQL 2.07.3. You also have the option to upgrade existing Aurora MySQL 2.* database clusters to Aurora MySQL 2.07.3. You can't upgrade an existing Aurora MySQL 1.* cluster directly to 2.07.3; however, you can restore its snapshot to Aurora MySQL 2.07.3.

To create a cluster with an older version of Aurora MySQL, please specify the engine version through the AWS Management Console, the AWS CLI, or the RDS API.

> **Note**
>
> This version is designated as a long-term support (LTS) release. For more information, see Aurora MySQL long-term support (LTS) releases in the *Amazon Aurora User Guide*.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Improvements

Security fixes:

Fixes and other enhancements to fine-tune handling in a managed environment.

- CVE-2021-2144
- CVE-2020-14567
- CVE-2020-14559
- CVE-2020-14553
- CVE-2020-14547
- CVE-2020-14540
- CVE-2020-2812
- CVE-2020-2806
- CVE-2020-2780
- CVE-2020-2765
- CVE-2020-2763
- CVE-2020-2760
- CVE-2020-2579
- CVE-2019-2740

Incompatible changes:

This version introduces a permission change that affects the behavior of the `mysqldump` command. Users must have the `PROCESS` privilege to access the `INFORMATION_SCHEMA.FILES` table. To run the `mysqldump` command without any changes, grant the `PROCESS` privilege to the database user that the `mysqldump` command connects to. You can also run the `mysqldump` command with the `--no-tablespaces` option. With that option, the `mysqldump` output doesn't include any `CREATE LOGFILE GROUP` or `CREATE TABLESPACE` statements. In that case, the `mysqldump` command doesn't access the `INFORMATION_SCHEMA.FILES` table, and you don't need to grant the `PROCESS` permission.

Availability improvements:

- Fixed a race condition in the lock manager between the killing of a connection/query and the termination of the session resulting in a database restart.
- Fixed an issue that results in a database restart after a multi-query statement that accesses multiple tables or databases is executed with the query cache enabled.
- Fixed an issue that might cause repeated restarts due to updates of virtual columns with secondary indexes.

Integration of MySQL community edition bug fixes

- *InnoDB:* Concurrent XA transactions that ran successfully to the XA prepare stage on the master conflicted when replayed on the slave, resulting in a lock wait timeout in the applier thread. The conflict was due to the GAP lock range which differed when the transactions were replayed serially on the slave. To prevent this type of conflict, GAP locks taken by XA transactions in READ COMMITTED isolation level are now released (and no longer inherited) when XA transactions reach the prepare stage. (Bug #27189701, Bug #25866046)
- *InnoDB:* A gap lock was taken unnecessarily during foreign key validation while using the READ COMMITTED isolation level. (Bug #25082593)
- *Replication:* When using XA transactions, if a lock wait timeout or deadlock occurred for the applier (SQL) thread on a replication slave, the automatic retry did not work. The cause was that while the SQL thread would do a rollback, it would not roll the XA transaction back. This meant that when the transaction was retried, the first event was XA START which was invalid as the XA transaction was already in progress, leading to an XAER_RMFAIL error. (Bug #24764800)
- *Replication:* Interleaved transactions could sometimes deadlock the slave applier when the transaction isolation level was set to REPEATABLE READ. (Bug #25040331)
- *Replication:* The value returned by a SHOW SLAVE STATUS statement for the total combined size of all existing relay log files (Relay_Log_Space) could become much larger than the actual disk space used by the relay log files. The I/O thread did not lock the variable while it updated the value, so the SQL thread could automatically delete a relay log file and write a reduced value before the I/O thread finished updating the value. The I/O thread then wrote its original size calculation, ignoring the SQL thread's update and so adding back the space for the deleted file. The Relay_Log_Space value is now locked during updates to prevent concurrent updates and ensure an accurate calculation. (Bug #26997096, Bug #87832)
- For an INSERT statement for which the VALUES list produced values for the second or later row using a subquery containing a join, the server could exit after failing to resolve the required privileges. (Bug #23762382)
- For a table having a TIMESTAMP or DATETIME column having a default of CURRENT_TIMESTAMP, the column could be initialized to `0000-00-00 00:00:00` if the table had a `BEFORE INSERT` trigger. (Bug #25209512, Bug #84077)
- A server exit could result from simultaneous attempts by multiple threads to register and deregister metadata Performance Schema objects. (Bug #26502135)
- Executing a stored procedure containing a statement that created a table from the contents of certain SELECT statements could result in a memory leak. (Bug #25586773)

- Executing a stored procedure containing a query that accessed a view could allocate memory that was not freed until the session ended. (Bug #25053286)
- Certain cases of subquery materialization could cause a server exit. These queries now produce an error suggesting that materialization be disabled. (Bug #26402045)
- Queries with many left joins were slow if join buffering was used (for example, using the block nested loop algorithm). (Bug #18898433, Bug #72854)
- The optimizer skipped the second column in a composite index when executing an inner join with a `LIKE` clause against the second column. (Bug #28086754)

Comparison with Aurora MySQL version 1

The following Amazon Aurora MySQL features are supported in Aurora MySQL Version 1 (compatible with MySQL 5.6), but these features are currently not supported in Aurora MySQL Version 2 (compatible with MySQL 5.7).

- Asynchronous key prefetch (AKP). For more information, see Optimizing Aurora indexed join queries with asynchronous key prefetch in the *Amazon Aurora User Guide*.
- Hash joins. For more information, see Optimizing large Aurora MySQL join queries with hash joins in the *Amazon Aurora User Guide*.
- Native functions for synchronously invoking AWS Lambda functions. For more information, see Invoking a Lambda function with an Aurora MySQL native function in the *Amazon Aurora User Guide*.
- Scan batching. For more information, see Aurora MySQL database engine updates 2017-12-11 (version 1.16) (Deprecated).
- Migrating data from MySQL using an Amazon S3 bucket. For more information, see Migrating data from MySQL by using an Amazon S3 bucket in the *Amazon Aurora User Guide*.

MySQL 5.7 compatibility

This Aurora MySQL version is wire-compatible with MySQL 5.7 and includes features such as JSON support, spatial indexes, and generated columns. Aurora MySQL uses a native implementation of spatial indexing using z-order curves to deliver >20x better write performance and >10x better read performance than MySQL 5.7 for spatial datasets.

This Aurora MySQL version does not currently support the following MySQL 5.7 features:

- Group replication plugin
- Increased page size
- InnoDB buffer pool loading at startup
- InnoDB full-text parser plugin
- Multisource replication
- Online buffer pool resizing
- Password validation plugin
- Query rewrite plugins
- Replication filtering
- The `CREATE TABLESPACE` SQL statement

Aurora MySQL database engine updates 2020-04-17 (version 2.07.2) (Deprecated)

Version: 2.07.2

Aurora MySQL 2.07.2 is generally available. Aurora MySQL 2.* versions are compatible with MySQL 5.7 and Aurora MySQL 1.* versions are compatible with MySQL 5.6.

Currently supported Aurora MySQL releases are 1.19.5, 1.19.6, 1.22.*, 1.23.*, 2.04.*, 2.07.*, 2.08.*, 2.09.*, 2.10.*, 3.01.* and 3.02.*.

You can restore a snapshot from a currently supported Aurora MySQL release into Aurora MySQL 2.07.2. You also have the option to upgrade existing Aurora MySQL 2.* database clusters to Aurora MySQL 2.07.2. You can't upgrade an existing Aurora MySQL 1.* cluster directly to 2.07.2; however, you can restore its snapshot to Aurora MySQL 2.07.2.

To create a cluster with an older version of Aurora MySQL, please specify the engine version through the AWS Management Console, the AWS CLI, or the RDS API.

> **Note**
>
> This version is designated as a long-term support (LTS) release. For more information, see Aurora MySQL long-term support (LTS) releases in the *Amazon Aurora User Guide*.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Improvements

Security fixes:

- CVE-2016-8287
- CVE-2016-5634

High priority fixes:

- Fixed an issue that caused cloning to take longer on some database clusters with high write loads.
- Fixed an issue that could cause queries on a reader DB instance with execution plans using secondary indexes to return uncommitted data. The issue is limited to data affected by data manipulation language (DML) operations that modify primary or secondary index key columns.

General improvements:

- Fixed an issue that resulted in a slow restore of an Aurora 1.x DB cluster containing FTS (Full Text Search) indexes to an Aurora 2.x DB cluster.
- Fixed an issue that caused slower restores of an Aurora 1.x database snapshot containing partitioned tables with special characters in table names to an Aurora 2.x DB cluster.
- Fixed an issue that caused errors when querying slow query logs and general logs in reader DB instances.

Integration of MySQL community edition bug fixes

- Bug #23104498: Fixed an issue in Performance Schema in reporting total memory used. (https://github.com/mysql/mysql-server/commit/20b6840df5452f47313c6f9a6ca075bfbc00a96b)
- Bug #22551677: Fixed an issue in Performance Schema that could lead to the database engine crashing when attempting to take it offline. (https://github.com/mysql/mysql-server/commit/05e2386eccd32b6b444b900c9f8a87a1d8d531e9)

- Bug #23550835, Bug #23298025, Bug #81464: Fixed an issue in Performance Schema that causes a database engine crash due to exceeding the capacity of an internal buffer. (https://github.com/mysql/mysql-server/commit/b4287f93857bf2f99b18fd06f555bbe5b12debfc)

Comparison with Aurora MySQL version 1

The following Amazon Aurora MySQL features are supported in Aurora MySQL Version 1 (compatible with MySQL 5.6), but these features are currently not supported in Aurora MySQL Version 2 (compatible with MySQL 5.7).

- Asynchronous key prefetch (AKP). For more information, see Optimizing Aurora indexed join queries with asynchronous key prefetch in the *Amazon Aurora User Guide*.
- Hash joins. For more information, see Optimizing large Aurora MySQL join queries with hash joins in the *Amazon Aurora User Guide*.
- Native functions for synchronously invoking AWS Lambda functions. For more information, see Invoking a Lambda function with an Aurora MySQL native function in the *Amazon Aurora User Guide*.
- Scan batching. For more information, see Aurora MySQL database engine updates 2017-12-11 (version 1.16) (Deprecated).
- Migrating data from MySQL using an Amazon S3 bucket. For more information, see Migrating data from MySQL by using an Amazon S3 bucket in the *Amazon Aurora User Guide*.

MySQL 5.7 compatibility

This Aurora MySQL version is wire-compatible with MySQL 5.7 and includes features such as JSON support, spatial indexes, and generated columns. Aurora MySQL uses a native implementation of spatial indexing using z-order curves to deliver >20x better write performance and >10x better read performance than MySQL 5.7 for spatial datasets.

This Aurora MySQL version does not currently support the following MySQL 5.7 features:

- Group replication plugin
- Increased page size
- InnoDB buffer pool loading at startup
- InnoDB full-text parser plugin

- Multisource replication
- Online buffer pool resizing
- Password validation plugin
- Query rewrite plugins
- Replication filtering
- The CREATE TABLESPACE SQL statement

Aurora MySQL database engine updates 2019-12-23 (version 2.07.1) (Deprecated)

Version: 2.07.1

Aurora MySQL 2.07.1 is generally available. Aurora MySQL 2.x versions are compatible with MySQL 5.7 and Aurora MySQL 1.x versions are compatible with MySQL 5.6.

Currently supported Aurora MySQL releases are 1.19.5, 1.19.6, 1.22.*, 1.23.*, 2.04.*, 2.07.*, 2.08.*, 2.09.*, 2.10.*, 3.01.* and 3.02.*.

You can restore a snapshot from a currently supported Aurora MySQL release into Aurora MySQL 2.07.1. You also have the option to upgrade existing Aurora MySQL 2.* database clusters to Aurora MySQL 2.07.1. You cannot upgrade an existing Aurora MySQL 1.* cluster directly to 2.07.1; however, you can restore its snapshot to Aurora MySQL 2.07.1.

To create a cluster with an older version of Aurora MySQL, please specify the engine version through the AWS Management Console, the AWS CLI, or the RDS API.

> **Note**
>
> This version is currently not available in the following AWS Regions: AWS GovCloud (US-East) [us-gov-east-1], AWS GovCloud (US-West) [us-gov-west-1], China (Ningxia) [cn-northwest-1], Asia Pacific (Hong Kong) [ap-east-1], and Middle East (Bahrain) [me-south-1]. There will be a separate announcement once it is made available.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

> **Note**
>
> For information on how to upgrade your Aurora MySQL database cluster, see Upgrading the minor version or patch level of an Aurora MySQL DB cluster in the *Amazon Aurora User Guide*.

Improvements

High priority fixes:

- Fixed a slow memory leak in Aurora specific database tracing and logging sub-system that lowers the freeable memory.

General Stability fixes:

- Fixed a crash during execution of a complex query involving multi-table joins and aggregation that use intermediate tables internally.

Comparison with Aurora MySQL Version 1

The following Amazon Aurora MySQL features are supported in Aurora MySQL Version 1 (compatible with MySQL 5.6), but these features are currently not supported in Aurora MySQL Version 2 (compatible with MySQL 5.7).

- Asynchronous key prefetch (AKP). For more information, see Optimizing Aurora indexed join queries with asynchronous key prefetch in the *Amazon Aurora User Guide*.
- Scan batching. For more information, see Aurora MySQL database engine updates 2017-12-11 (version 1.16) (Deprecated).
- Migrating data from MySQL using an Amazon S3 bucket. For more information, see Migrating data from MySQL by using an Amazon S3 bucket in the *Amazon Aurora User Guide*.

MySQL 5.7 compatibility

This Aurora MySQL version is wire-compatible with MySQL 5.7 and includes features such as JSON support, spatial indexes, and generated columns. Aurora MySQL uses a native implementation of

spatial indexing using z-order curves to deliver >20x better write performance and >10x better read performance than MySQL 5.7 for spatial datasets.

This Aurora MySQL version does not currently support the following MySQL 5.7 features:

- Group replication plugin
- Increased page size
- InnoDB buffer pool loading at startup
- InnoDB full-text parser plugin
- Multisource replication
- Online buffer pool resizing
- Password validation plugin
- Query rewrite plugins
- Replication filtering
- The `CREATE TABLESPACE` SQL statement

Aurora MySQL database engine updates 2019-11-25 (version 2.07.0) (Deprecated)

Version: 2.07.0

Aurora MySQL 2.07.0 is generally available. Aurora MySQL 2.x versions are compatible with MySQL 5.7 and Aurora MySQL 1.x versions are compatible with MySQL 5.6.

Currently supported Aurora MySQL releases are 1.19.5, 1.19.6, 1.22.*, 1.23.*, 2.04.*, 2.07.*, 2.08.*, 2.09.*, 2.10.*, 3.01.* and 3.02.*.

You can restore a snapshot from a currently supported Aurora MySQL release into Aurora MySQL 2.07.0. You also have the option to upgrade existing Aurora MySQL 2.* database clusters to Aurora MySQL 2.07.0. You cannot upgrade an existing Aurora MySQL 1.* cluster directly to 2.07.0; however, you can restore its snapshot to Aurora MySQL 2.07.0.

To create a cluster with an older version of Aurora MySQL, please specify the engine version through the AWS Management Console, the AWS CLI, or the RDS API.

> **Note**
>
> This version is currently not available in the following AWS Regions: AWS GovCloud (US-East) [us-gov-east-1], AWS GovCloud (US-West) [us-gov-west-1], China (Ningxia) [cn-northwest-1], Asia Pacific (Hong Kong) [ap-east-1], Middle East (Bahrain) [me-south-1], and South America (São Paulo) [sa-east-1]. There will be a separate announcement once it is made available.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

> **Note**
>
> For information on how to upgrade your Aurora MySQL database cluster, see Upgrading the minor version or patch level of an Aurora MySQL DB cluster in the *Amazon Aurora User Guide*.

Improvements

New features:

- Global Databases now allow adding secondary read-only replica regions for database clusters deployed in these AWS Regions: regions: US East (N. Virginia) [us-east-1], US East (Ohio) [us-east-2], US West (N. California) [us-west-1], US West (Oregon) [us-west-2], Europe (Ireland) [eu-west-1], Europe (London) [eu-west-2], Europe (Paris) [eu-west-3], Asia Pacific (Tokyo) [ap-northeast-1], Asia Pacific (Seoul) [ap-northeast-2], Asia Pacific (Singapore) [ap-southeast-1], Asia Pacific (Sydney) [ap-southeast-2], Canada (Central) [ca-central-1], Europe (Frankfurt) [eu-central-1], and Asia Pacific (Mumbai) [ap-south-1].
- Amazon Aurora machine learning is a highly optimized integration between the Aurora MySQL database and AWS machine learning (ML) services. Aurora machine learning allows developers to add a variety of ML-based predictions to their database applications by invoking ML models using the familiar SQL programming language they already use for database development, without having to build custom integrations or learn separate tools. For more information, see Using machine learning (ML) capabilities with Amazon Aurora.

- Added support for the ANSI `READ COMMITTED` isolation level on the read replicas. This isolation level enables long-running queries on the read replica to execute without impacting the high throughput of writes on the writer node. For more information, see Aurora MySQL isolation levels.

Critical fixes:

- CVE-2019-2922
- CVE-2019-2923
- CVE-2019-2924
- CVE-2019-2910

High-priority fixes:

- Fixed an issue in the DDL recovery that resulted in prolonged database downtime. Clusters that become unavailable after executing multi-table drop statement, for example `DROP TABLE t1, t2, t3`, should be updated to this version.
- Fixed an issue in the DDL recovery that resulted in prolonged database downtime. Clusters that become unavailable after executing `INPLACE ALTER TABLE` DDL statements should be updated to this version.

General stability fixes:

- Fixed an issue that generated inconsistent data in the `information_schema.replica_host_status` table.

Integration of MySQL community edition bug fixes

- Bug #26251621: INCORRECT BEHAVIOR WITH TRIGGER AND GCOL
- Bug #22574695: ASSERTION `!TABLE || (!TABLE->READ_SET || BITMAP_IS_SET(TABLE->READ_SET, FIEL
- Bug #25966845: INSERT ON DUPLICATE KEY GENERATE A DEADLOCK
- Bug #23070734: CONCURRENT TRUNCATE TABLES CAUSE STALL
- Bug #26191879: FOREIGN KEY CASCADES USE EXCESSIVE MEMORY

- Bug #20989615: INNODB AUTO_INCREMENT PRODUCES SAME VALUE TWICE

Comparison with Aurora MySQL version 1

The following Amazon Aurora MySQL features are supported in Aurora MySQL Version 1 (compatible with MySQL 5.6), but these features are currently not supported in Aurora MySQL Version 2 (compatible with MySQL 5.7).

- Asynchronous key prefetch (AKP). For more information, see Optimizing Aurora indexed join queries with asynchronous key prefetch in the *Amazon Aurora User Guide*.
- Scan batching. For more information, see Aurora MySQL database engine updates 2017-12-11 (version 1.16) (Deprecated).
- Migrating data from MySQL using an Amazon S3 bucket. For more information, see Migrating data from MySQL by using an Amazon S3 bucket in the *Amazon Aurora User Guide*.

MySQL 5.7 compatibility

Aurora MySQL 2.07.0 is wire-compatible with MySQL 5.7 and includes features such as JSON support, spatial indexes, and generated columns. Aurora MySQL uses a native implementation of spatial indexing using z-order curves to deliver >20x better write performance and >10x better read performance than MySQL 5.7 for spatial datasets.

Aurora MySQL 2.07.0 does not currently support the following MySQL 5.7 features:

- Group replication plugin
- Increased page size
- InnoDB buffer pool loading at startup
- InnoDB full-text parser plugin
- Multisource replication
- Online buffer pool resizing
- Password validation plugin
- Query rewrite plugins
- Replication filtering
- The `CREATE TABLESPACE` SQL statement

Aurora MySQL database engine updates 2019-11-22 (version 2.06.0) (Deprecated)

Version: 2.06.0

Aurora MySQL 2.06.0 is generally available. Aurora MySQL 2.x versions are compatible with MySQL 5.7 and Aurora MySQL 1.x versions are compatible with MySQL 5.6.

Currently supported Aurora MySQL releases are 1.14.*, 1.15.*, 1.16.*, 1.17.*, 1.18.*, 1.19.*, 2.01.*, 2.02.*, 2.03.*, 2.04.*, 2.05.*, and 2.06.*.

You can restore a snapshot from a currently supported Aurora MySQL release into Aurora MySQL 2.06.0. You also have the option to upgrade existing Aurora MySQL 2.* database clusters to Aurora MySQL 2.06.0. You cannot upgrade an existing Aurora MySQL 1.* cluster directly to 2.06.0; however, you can restore its snapshot to Aurora MySQL 2.06.0.

To create a cluster with an older version of Aurora MySQL, please specify the engine version through the AWS Management Console, the AWS CLI, or the RDS API.

Note

This version is currently not available in the following AWS Regions: AWS GovCloud (US-East) [us-gov-east-1], AWS GovCloud (US-West) [us-gov-west-1], China (Ningxia) [cn-northwest-1], Asia Pacific (Hong Kong) [ap-east-1], and Middle East (Bahrain) [me-south-1]. There will be a separate announcement once it is made available.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Note

For information on how to upgrade your Aurora MySQL database cluster, see Upgrading the minor version or patch level of an Aurora MySQL DB cluster in the *Amazon Aurora User Guide*.

Improvements

New features:

- Aurora MySQL clusters now support the instance types db.r5.8xlarge, db.r5.16xlarge, and db.r5.24xlarge. For more information about instance types for Aurora MySQL clusters, see Aurora DB instance classes in the *Amazon Aurora User Guide*.
- The hash join feature is now generally available and does not require the Aurora lab mode setting to be ON. This feature can improve query performance when you need to join a large amount of data by using an equi-join. For more information about using this feature, see Using the Data API for Aurora Serverless in the *Amazon Aurora User Guide*.
- The hot row contention feature is now generally available and does not require the Aurora lab mode setting to be ON. This feature substantially improves throughput for workloads with many transactions contending for rows on the same page.
- Aurora MySQL 2.06 and higher support "rewinding" a DB cluster to a specific time, without restoring data from a backup. This feature, known as Backtrack, provides a quick way to recover from user errors, such as dropping the wrong table or deleting the wrong row. Backtrack completes within seconds, even for large databases. Read the AWS blog for an overview, and refer to Backtracking an Aurora DB cluster in the *Amazon Aurora User Guide* for more details.
- Aurora 2.06 and higher support synchronous AWS Lambda invocations through the native function `lambda_sync()`. Also available is native function `lambda_async()`, which can be used as an alternative to the existing stored procedure for asynchronous Lambda invocation. For information about calling Lambda functions, see Invoking a Lambda function with an Aurora MySQL native function in the *Amazon Aurora User Guide*.

Critical fixes:

None.

High-priority fixes:

Security fixes

- CVE-2019-2805
- CVE-2019-2791
- CVE-2019-2778
- CVE-2019-2758

- CVE-2019-2739
- CVE-2019-2730
- CVE-2018-3064
- CVE-2018-3058
- CVE-2018-2786
- CVE-2017-3653
- CVE-2017-3465
- CVE-2017-3455
- CVE-2017-3244
- CVE-2016-5612

Connection handling

- Database availability has been improved to better service a surge in client connections while executing one or more DDLs. It is handled by temporarily creating additional threads when needed. You are advised to upgrade if the database becomes unresponsive following a surge in connections while processing DDL.

Engine restart

- Fixed an issue of prolonged unavailability while restarting the engine. This addresses an issue in the buffer pool initialization. This issue occurs rarely but can potentially impact any supported release.
- Fixed an issue that causes a database configured as a binary log (binlog) master to restart while a heavy write workload is running.

General stability fixes:

- Made improvements where queries accessing uncached data could be slower than usual. Customers experiencing unexplained elevated read latency while accessing uncached data are encouraged to upgrade as they may be experiencing this issue.
- Fixed an issue that failed to restore partitioned tables from a database snapshot. Customers who encounter errors when accessing partitioned tables in a database that has been restored from the snapshot of an Aurora MySQL 1.* database are advised to use this version.

- Improved stability of the Aurora Replicas by fixing lock contention between threads serving read queries and the one applying schema changes while a DDL query is in progress on the writer DB instance.
- Fixed a stability issue related to `mysql.innodb_table_stats` table update triggered by DDL operations.
- Fixed an issue that incorrectly reported `ERROR 1836` when a nested query is executed against a temporary table on the Aurora Replica.

Performance enhancements:

- Improved performance of binlog replication by preventing unnecessary API calls to the cache if the query cache has been disabled on the binlog worker.

Comparison with Aurora MySQL version 1

The following Amazon Aurora MySQL features are supported in Aurora MySQL Version 1 (compatible with MySQL 5.6), but these features are currently not supported in Aurora MySQL Version 2 (compatible with MySQL 5.7).

- Asynchronous key prefetch (AKP). For more information, see Optimizing Aurora indexed join queries with asynchronous key prefetch in the *Amazon Aurora User Guide*.
- Scan batching. For more information, see Aurora MySQL database engine updates 2017-12-11 (version 1.16) (Deprecated).
- Migrating data from MySQL using an Amazon S3 bucket. For more information, see Migrating data from MySQL by using an Amazon S3 bucket in the *Amazon Aurora User Guide*.

MySQL 5.7 compatibility

Aurora MySQL 2.06.0 is wire-compatible with MySQL 5.7 and includes features such as JSON support, spatial indexes, and generated columns. Aurora MySQL uses a native implementation of spatial indexing using z-order curves to deliver >20x better write performance and >10x better read performance than MySQL 5.7 for spatial datasets.

Aurora MySQL 2.06.0 does not currently support the following MySQL 5.7 features:

- Group replication plugin

- Increased page size
- InnoDB buffer pool loading at startup
- InnoDB full-text parser plugin
- Multisource replication
- Online buffer pool resizing
- Password validation plugin
- Query rewrite plugins
- Replication filtering
- The `CREATE TABLESPACE` SQL statement

Aurora MySQL database engine updates 2019-11-11 (version 2.05.0) (Deprecated)

Version: 2.05.0

Aurora MySQL 2.05.0 is generally available. Aurora MySQL 2.x versions are compatible with MySQL 5.7 and Aurora MySQL 1.x versions are compatible with MySQL 5.6.

Currently supported Aurora MySQL releases are 1.14.*, 1.15.*, 1.16.*, 1.17.*, 1.18.*, 1.19.*, 2.01.*, 2.02.*, 2.03.* and 2.04.*.

You can restore a snapshot from a currently supported Aurora MySQL release into Aurora MySQL 2.05.0. You also have the option to upgrade existing Aurora MySQL 2.* database clusters, up to 2.04.6, to Aurora MySQL 2.05.0. You cannot upgrade an existing Aurora MySQL 1.* cluster directly to 2.05.0; however, you can restore its snapshot to Aurora MySQL 2.05.0.

To create a cluster with an older version of Aurora MySQL, please specify the engine version through the AWS Management Console, the AWS CLI, or the RDS API.

> **Note**
>
> This version is currently not available in the following AWS Regions: AWS GovCloud (US-East) [us-gov-east-1], AWS GovCloud (US-West) [us-gov-west-1], China (Ningxia) [cn-northwest-1], Asia Pacific (Hong Kong) [ap-east-1], Europe (Stockholm) [eu-north-1], and Middle East (Bahrain) [me-south-1]. There will be a separate announcement once it is made available.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

> **Note**
>
> For information on how to upgrade your Aurora MySQL database cluster, see Upgrading the minor version or patch level of an Aurora MySQL DB cluster in the *Amazon Aurora User Guide*.

Improvements

Critical fixes:

- CVE-2018-0734
- CVE-2019-2534
- CVE-2018-3155
- CVE-2018-2612
- CVE-2017-3599
- CVE-2018-3056
- CVE-2018-2562
- CVE-2017-3329
- CVE-2018-2696
- Fixed an issue where the events in current binlog file on the master were not replicated on the worker if the value of the parameter `sync_binlog` was not set to 1.

High-priority fixes:

- Customers with database size close to 64 tebibytes (TiB) are strongly advised to upgrade to this version to avoid downtime due to stability bugs affecting volumes close to the Aurora storage limit.
- The default value of the parameter `aurora_binlog_replication_max_yield_seconds` has been changed to zero to prevent an increase in replication lag in favor of foreground query performance on the binlog master.

Integration of MySQL bug fixes

- Bug#23054591: PURGE BINARY LOGS TO is reading the whole binlog file and causing MySql to Stall

Comparison with Aurora MySQL version 1

The following Amazon Aurora MySQL features are supported in Aurora MySQL Version 1 (compatible with MySQL 5.6), but these features are currently not supported in Aurora MySQL Version 2 (compatible with MySQL 5.7).

- Asynchronous key prefetch (AKP). For more information, see Optimizing Aurora indexed join queries with asynchronous key prefetch in the *Amazon Aurora User Guide*.
- Hash joins. For more information, see Optimizing large Aurora MySQL join queries with hash joins in the *Amazon Aurora User Guide*.
- Native functions for synchronously invoking AWS Lambda functions. For more information, see Invoking a Lambda function with an Aurora MySQL native function in the *Amazon Aurora User Guide*.
- Scan batching. For more information, see Aurora MySQL database engine updates 2017-12-11 (version 1.16) (Deprecated).
- Migrating data from MySQL using an Amazon S3 bucket. For more information, see Migrating data from MySQL by using an Amazon S3 bucket in the *Amazon Aurora User Guide*.

MySQL 5.7 compatibility

Aurora MySQL 2.05.0 is wire-compatible with MySQL 5.7 and includes features such as JSON support, spatial indexes, and generated columns. Aurora MySQL uses a native implementation of spatial indexing using z-order curves to deliver >20x better write performance and >10x better read performance than MySQL 5.7 for spatial datasets.

Aurora MySQL 2.05.0 does not currently support the following MySQL 5.7 features:

- Group replication plugin
- Increased page size
- InnoDB buffer pool loading at startup

- InnoDB full-text parser plugin
- Multisource replication
- Online buffer pool resizing
- Password validation plugin
- Query rewrite plugins
- Replication filtering
- The CREATE TABLESPACE SQL statement

Aurora MySQL database engine updates 2020-08-14 (version 2.04.9) (Deprecated)

Version: 2.04.9

Aurora MySQL 2.04.9 is generally available. Aurora MySQL 2.x versions are compatible with MySQL 5.7 and Aurora MySQL 1.x versions are compatible with MySQL 5.6.

Currently supported Aurora MySQL releases are 1.19.5, 1.19.6, 1.22.*, 1.23.*, 2.04.*, 2.07.*, 2.08.*, 2.09.*, 2.10.*, 3.01.* and 3.02.*.

To create a cluster with an older version of Aurora MySQL, please specify the engine version through the AWS Management Console, the AWS CLI, or the RDS API.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

> **Note**
>
> This version is currently not available in the following AWS Regions: AWS GovCloud (US-East) [us-gov-east-1], AWS GovCloud (US-West) [us-gov-west-1], Asia Pacific (Hong Kong) [ap-east-1], and Middle East (Bahrain) [me-south-1]. There will be a separate announcement once it is made available.

> **Note**
>
> For information on how to upgrade your Aurora MySQL database cluster, see Upgrading the minor version or patch level of an Aurora MySQL DB cluster in the *Amazon Aurora User Guide*.

Improvements

High-priority fixes:

Critical fixes:

- CVE-2020-2760
- CVE-2019-5443
- CVE-2019-3822
- CVE-2019-2924
- CVE-2019-2923
- CVE-2019-2922
- CVE-2019-2911
- CVE-2019-2910
- CVE-2019-2805
- CVE-2019-2791
- CVE-2019-2778
- CVE-2019-2758
- CVE-2019-2740
- CVE-2019-2739
- CVE-2019-2730
- CVE-2019-2628
- CVE-2018-3064
- CVE-2018-3058
- CVE-2018-2813

- CVE-2018-2786
- CVE-2017-3653
- CVE-2017-3465
- CVE-2017-3464
- CVE-2017-3455
- CVE-2017-3244
- CVE-2016-5612
- CVE-2016-5436

Availability improvements:

- Fixed an issue that could cause a database restart or failover due to execution of a `kill session` command. If you encounter this issue, contact AWS support to enable this fix on your instance.
- Fixed an issue that causes a database restart during execution of a complex query involving multi-table joins and aggregation that use intermediate tables internally.
- Fixed an issue that causes database restarts due to an interrupted `DROP TABLE` on multiple tables.
- Fixed an issue that causes a database failover during database recovery.
- Fixed a database restart caused by incorrect reporting of threads_running when audit and slow query logs are enabled.
- Fixed an issue where a `kill query` command might get stuck during execution.
- Fixed a race condition in the lock manager that resulted in a database restart or failover during transaction rollback.
- Fixed an issue that triggered database restart or failover when multiple connections are trying to update the same table with a Full-Text Search index.
- Fixed an issue that can cause a deadlatch when purging an index resulting in a failover or restart.

General improvements:

- Fixed issues that could cause queries on read replicas to use data from an uncommitted transaction. This issue is limited to the transactions that are started immediately after a database restart.

- Fixed an issue encountered during `INPLACE ALTER TABLE` for a table with triggers defined and when the DDL did not contain a `RENAME` clause.
- Fixed an issue that caused cloning to take longer on some database clusters with high writeload.
- Fixed an issue encountered during an upgrade when a partitioned table has embedded spaces in the name.
- Fixed an issue where the read replica might transiently see partial results of a recently committed transaction on the writer.
- Fixed an issue where queries on a read replica against an FTS table may produce stale results. This will only occur when the FTS query on the read replica closely follows a query on `INFORMATION_SCHEMA.INNODB_SYS_TABLES` for the same FTS table on the writer.
- Fixed an issue that resulted in a slow restore of Aurora 1.x database cluster containing FTS (Full-Text Search) indexes to an Aurora 2.x database cluster.
- Extended maximum allowable length to 2000 for `server_audit_incl_users` and `server_audit_excl_users` global parameters.
- Fixed an issue where Aurora 1.x to Aurora 2.x restore might take an extended time to complete.
- Fixed an issue where a `lambda_async` invocation through stored procedure doesn't work with Unicode.
- Fixed an issue encountered when a spatial index does not properly handle an off-record geometry column.
- Fixed an issue that might cause a query to fail on a reader DB instance with `InternalFailureException` error with message"Operation terminated (internal error)".

Integration of MySQL bug fixes

- Bug #23070734, Bug #80060: Concurrent TRUNCATE TABLEs cause stalls
- Bug #23103937: PS_TRUNCATE_ALL_TABLES() DOES NOT WORK IN SUPER_READ_ONLY MODE
- Bug#22551677: When taking the server offline, a race condition within the Performance Schema could lead to a server exit.
- Bug #27082268: Invalid FTS sync synchronization.
- BUG #12589870: Fixed an issues which causes a restart with multi-query statement when query cache is enabled.
- Bug #26402045: Certain cases of subquery materialization could cause a server exit. These queries now produce an error suggesting that materialization be disabled.

- Bug #18898433: Queries with many left joins were slow if join buffering was used (for example, using the block nested loop algorithm).
- Bug #25222337: A NULL virtual column field name in a virtual index caused a server exit during a field name comparison that occurs while populating virtual columns affected by a foreign key constraint. (https://github.com/mysql/mysql-server/commit/273d5c9d7072c63b6c47dbef6963d7dc491d5131)
- Bug #25053286: Executing a stored procedure containing a query that accessed a view could allocate memory that was not freed until the session ended. (https://github.com/mysql/mysql-server/commit/d7b37d4d141a95f577916448650c429f0d6e193d)
- Bug #25586773: Executing a stored procedure containing a statement that created a table from the contents of certain SELECT (https://dev.mysql.com/doc/refman/5.7/en/select.html) statements could result in a memory leak. (https://github.com/mysql/mysql-server/commit/88301e5adab65f6750f66af284be410c4369d0c1)
- Bug #26666274: INFINITE LOOP IN PERFORMANCE SCHEMA BUFFER CONTAINER.
- Bug #23550835, Bug #23298025, Bug #81464: A SELECT Performance Schema tables when an internal buffer was full could cause a server exit.

Comparison with Aurora MySQL version 1

The following Amazon Aurora MySQL features are supported in Aurora MySQL Version 1 (compatible with MySQL 5.6), but these features are currently not supported in Aurora MySQL Version 2 (compatible with MySQL 5.7).

- Asynchronous key prefetch (AKP). For more information, see Optimizing Aurora indexed join queries with asynchronous key prefetch in the *Amazon Aurora User Guide*.
- Hash joins. For more information, see Optimizing large Aurora MySQL join queries with hash joins in the *Amazon Aurora User Guide*.
- Native functions for synchronously invoking AWS Lambda functions. For more information, see Invoking a Lambda function with an Aurora MySQL native function in the *Amazon Aurora User Guide*.
- Scan batching. For more information, see Aurora MySQL database engine updates 2017-12-11 (version 1.16) (Deprecated).
- Migrating data from MySQL using an Amazon S3 bucket. For more information, see Migrating data from MySQL by using an Amazon S3 bucket in the *Amazon Aurora User Guide*.

MySQL 5.7 compatibility

Aurora MySQL 2.04.9 is wire-compatible with MySQL 5.7 and includes features such as JSON support, spatial indexes, and generated columns. Aurora MySQL uses a native implementation of spatial indexing using z-order curves to deliver >20x better write performance and >10x better read performance than MySQL 5.7 for spatial datasets.

Aurora MySQL 2.04.9 does not currently support the following MySQL 5.7 features:

- Group replication plugin
- Increased page size
- InnoDB buffer pool loading at startup
- InnoDB full-text parser plugin
- Multisource replication
- Online buffer pool resizing
- Password validation plugin
- Query rewrite plugins
- Replication filtering
- The `CREATE TABLESPACE` SQL statement

Aurora MySQL database engine updates 2019-11-20 (version 2.04.8) (Deprecated)

Version: 2.04.8

Aurora MySQL 2.04.8 is generally available. Aurora MySQL 2.x versions are compatible with MySQL 5.7 and Aurora MySQL 1.x versions are compatible with MySQL 5.6.

Currently supported Aurora MySQL releases are 1.19.5, 1.19.6, 1.22.*, 1.23.*, 2.04.*, 2.07.*, 2.08.*, 2.09.*, 2.10.*, 3.01.* and 3.02.*.

You can restore a snapshot of any 2.* Aurora MySQL release into Aurora MySQL 2.04.8. You also have the option to upgrade existing Aurora MySQL 2.* database clusters to Aurora MySQL 2.04.8.

To create a cluster with an older version of Aurora MySQL, please specify the engine version through the AWS Management Console, the AWS CLI, or the RDS API.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

> **Note**
>
> This version is currently not available in the following AWS Regions: AWS GovCloud (US-East) [us-gov-east-1], AWS GovCloud (US-West) [us-gov-west-1], Asia Pacific (Hong Kong) [ap-east-1], and Middle East (Bahrain) [me-south-1]. There will be a separate announcement once it is made available.

> **Note**
>
> For information on how to upgrade your Aurora MySQL database cluster, see Upgrading the minor version or patch level of an Aurora MySQL DB cluster in the *Amazon Aurora User Guide*.

Improvements

New features:

- **Read replica improvements:**
  - Reduced network traffic from the writer instance by efficiently transmitting data to reader instances within the Aurora DB cluster. This improvement is enabled by default, because it helps prevent replicas from falling behind and restarting. The parameter for this feature is `aurora_enable_repl_bin_log_filtering`.
  - Reduced network traffic from the writer to reader instances within the Aurora DB cluster using compression. This improvement is enabled by default for 8xlarge and 16xlarge instance classes only, because these instances can tolerate additional CPU overhead for compression. The parameter for this feature is `aurora_enable_replica_log_compression`.

High-priority fixes:

- Improved memory management in the Aurora writer instance that prevents restart of writer due to out of memory conditions during heavy workload in presence of reader instances within the Aurora DB cluster.
- Fix for a non-deterministic condition in the scheduler that results in engine restart while accessing the performance schema object concurrently.

Comparison with Aurora MySQL version 1

The following Amazon Aurora MySQL features are supported in Aurora MySQL Version 1 (compatible with MySQL 5.6), but these features are currently not supported in Aurora MySQL Version 2 (compatible with MySQL 5.7).

- Asynchronous key prefetch (AKP). For more information, see Optimizing Aurora indexed join queries with asynchronous key prefetch in the *Amazon Aurora User Guide*.
- Hash joins. For more information, see Optimizing large Aurora MySQL join queries with hash joins in the *Amazon Aurora User Guide*.
- Native functions for synchronously invoking AWS Lambda functions. For more information, see Invoking a Lambda function with an Aurora MySQL native function in the *Amazon Aurora User Guide*.
- Scan batching. For more information, see Aurora MySQL database engine updates 2017-12-11 (version 1.16) (Deprecated).
- Migrating data from MySQL using an Amazon S3 bucket. For more information, see Migrating data from MySQL by using an Amazon S3 bucket in the *Amazon Aurora User Guide*.

MySQL 5.7 compatibility

Aurora MySQL 2.04.8 is wire-compatible with MySQL 5.7 and includes features such as JSON support, spatial indexes, and generated columns. Aurora MySQL uses a native implementation of spatial indexing using z-order curves to deliver >20x better write performance and >10x better read performance than MySQL 5.7 for spatial datasets.

Aurora MySQL 2.04.8 does not currently support the following MySQL 5.7 features:

- Group replication plugin
- Increased page size
- InnoDB buffer pool loading at startup

- InnoDB full-text parser plugin
- Multisource replication
- Online buffer pool resizing
- Password validation plugin
- Query rewrite plugins
- Replication filtering
- The CREATE TABLESPACE SQL statement

Aurora MySQL database engine updates 2019-11-14 (version 2.04.7) (Deprecated)

Version: 2.04.7

Aurora MySQL 2.04.7 is generally available. Aurora MySQL 2.x versions are compatible with MySQL 5.7 and Aurora MySQL 1.x versions are compatible with MySQL 5.6.

Currently supported Aurora MySQL releases are 1.19.5, 1.19.6, 1.22.*, 1.23.*, 2.04.*, 2.07.*, 2.08.*, 2.09.*, 2.10.*, 3.01.* and 3.02.*.

You can restore a snapshot from a currently supported Aurora MySQL release into Aurora MySQL 2.04.7. You also have the option to upgrade existing Aurora MySQL 2.* database clusters to Aurora MySQL 2.04.7. You can't upgrade an existing Aurora MySQL 1.* cluster directly to 2.04.7; however, you can restore its snapshot to Aurora MySQL 2.04.7.

To create a cluster with an older version of Aurora MySQL, please specify the engine version through the AWS Management Console, the AWS CLI, or the RDS API.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

> **Note**
>
> This version is currently not available in the following AWS Regions: AWS GovCloud (US-East) [us-gov-east-1], AWS GovCloud (US-West) [us-gov-west-1], Asia Pacific (Hong Kong) [ap-east-1], and Middle East (Bahrain) [me-south-1]. There will be a separate announcement once it is made available.

> **Note**
>
> For information on how to upgrade your Aurora MySQL database cluster, see Upgrading the minor version or patch level of an Aurora MySQL DB cluster in the *Amazon Aurora User Guide*.

Improvements

High-priority fixes:

Connection Handling

- Database availability has been improved to better service a surge in client connections while executing one or more DDLs. It is handled by temporarily creating additional threads when needed. You are advised to upgrade if the database becomes unresponsive following a surge in connections while processing DDL.
- Fixed an issue that resulted in an incorrect value for the `Threads_running` global status variable.

Engine Restart

- Fixed an issue of prolonged unavailability while restarting the engine. This addresses an issue in the buffer pool initialization. This issue occurs rarely but can potentially impact any supported release.

General stability fixes:

- Made improvements where queries accessing uncached data could be slower than usual. Customers experiencing unexplained elevated read latencies while accessing uncached data are encouraged to upgrade as they may be experiencing this issue.

Comparison with Aurora MySQL version 1

The following Amazon Aurora MySQL features are supported in Aurora MySQL Version 1 (compatible with MySQL 5.6), but these features are currently not supported in Aurora MySQL Version 2 (compatible with MySQL 5.7).

- Asynchronous key prefetch (AKP). For more information, see Optimizing Aurora indexed join queries with asynchronous key prefetch in the *Amazon Aurora User Guide*.
- Hash joins. For more information, see Optimizing large Aurora MySQL join queries with hash joins in the *Amazon Aurora User Guide*.
- Native functions for synchronously invoking AWS Lambda functions. For more information, see Invoking a Lambda function with an Aurora MySQL native function in the *Amazon Aurora User Guide*.
- Scan batching. For more information, see Aurora MySQL database engine updates 2017-12-11 (version 1.16) (Deprecated).
- Migrating data from MySQL using an Amazon S3 bucket. For more information, see Migrating data from MySQL by using an Amazon S3 bucket in the *Amazon Aurora User Guide*.

MySQL 5.7 compatibility

Aurora MySQL 2.04.7 is wire-compatible with MySQL 5.7 and includes features such as JSON support, spatial indexes, and generated columns. Aurora MySQL uses a native implementation of spatial indexing using z-order curves to deliver >20x better write performance and >10x better read performance than MySQL 5.7 for spatial datasets.

Aurora MySQL 2.04.7 does not currently support the following MySQL 5.7 features:

- Group replication plugin
- Increased page size
- InnoDB buffer pool loading at startup
- InnoDB full-text parser plugin
- Multisource replication
- Online buffer pool resizing
- Password validation plugin
- Query rewrite plugins
- Replication filtering
- The `CREATE TABLESPACE` SQL statement

Aurora MySQL database engine updates 2019-09-19 (version 2.04.6) (Deprecated)

Version: 2.04.6

Aurora MySQL 2.04.6 is generally available. Aurora MySQL 2.x versions are compatible with MySQL 5.7 and Aurora MySQL 1.x versions are compatible with MySQL 5.6.

Currently supported Aurora MySQL releases are 1.19.5, 1.19.6, 1.22.*, 1.23.*, 2.04.*, 2.07.*, 2.08.*, 2.09.*, 2.10.*, 3.01.* and 3.02.*.

You have the option to upgrade existing Aurora MySQL 2.* database clusters to Aurora MySQL 2.04.6. We do not allow in-place upgrade of Aurora MySQL 1.* clusters. This restriction will be lifted in a later Aurora MySQL 2.* release. You can restore snapshots of Aurora MySQL 1.14.*, 1.15.*, 1.16.*, 1.17.*, 1.18.*, 1.19.*, 2.01.*, 2.02.*, 2.03.* and 2.04.* into Aurora MySQL 2.04.6.

To use an older version of Aurora MySQL, you can create new database clusters by specifying the engine version through the AWS Management Console, the AWS CLI, or the Amazon RDS API.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Note

This version is currently not available in the following AWS Regions: Europe (London) [eu-west-2], AWS GovCloud (US-East) [us-gov-east-1], AWS GovCloud (US-West) [us-gov-west-1], China (Ningxia) [cn-northwest-1], and Asia Pacific (Hong Kong) [ap-east-1]. There will be a separate announcement once it is made available.

Note

For information on how to upgrade your Aurora MySQL database cluster, see Upgrading the minor version or patch level of an Aurora MySQL DB cluster in the *Amazon Aurora User Guide*.

Improvements

- Fixed an issue where the events in current binlog file on the master were not replicated on the worker if the value of the parameter `sync_binlog` was not set to 1.
- The default value of the parameter `aurora_binlog_replication_max_yield_seconds` has been changed to zero to prevent an increase in replication lag in favor of foreground query performance on the binlog master.

Integration of MySQL bug fixes

- Bug#23054591: PURGE BINARY LOGS TO is reading the whole binlog file and causing MySql to Stall

Comparison with Aurora MySQL version 1

The following Amazon Aurora MySQL features are supported in Aurora MySQL Version 1 (compatible with MySQL 5.6), but these features are currently not supported in Aurora MySQL Version 2 (compatible with MySQL 5.7).

- Asynchronous key prefetch (AKP). For more information, see Optimizing Aurora indexed join queries with asynchronous key prefetch in the *Amazon Aurora User Guide*.
- Hash joins. For more information, see Optimizing large Aurora MySQL join queries with hash joins in the *Amazon Aurora User Guide*.
- Native functions for synchronously invoking AWS Lambda functions. For more information, see Invoking a Lambda function with an Aurora MySQL native function in the *Amazon Aurora User Guide*.
- Scan batching. For more information, see Aurora MySQL database engine updates 2017-12-11 (version 1.16) (Deprecated).
- Migrating data from MySQL using an Amazon S3 bucket. For more information, see Migrating data from MySQL by using an Amazon S3 bucket in the *Amazon Aurora User Guide*.

MySQL 5.7 compatibility

Aurora MySQL 2.04.6 is wire-compatible with MySQL 5.7 and includes features such as JSON support, spatial indexes, and generated columns. Aurora MySQL uses a native implementation of

spatial indexing using z-order curves to deliver >20x better write performance and >10x better read performance than MySQL 5.7 for spatial datasets.

Aurora MySQL 2.04.6 does not currently support the following MySQL 5.7 features:

- Group replication plugin
- Increased page size
- InnoDB buffer pool loading at startup
- InnoDB full-text parser plugin
- Multisource replication
- Online buffer pool resizing
- Password validation plugin
- Query rewrite plugins
- Replication filtering
- The `CREATE TABLESPACE` SQL statement

Aurora MySQL database engine updates 2019-07-08 (version 2.04.5) (Deprecated)

Version: 2.04.5

Aurora MySQL 2.04.5 is generally available. Aurora MySQL 2.x versions are compatible with MySQL 5.7 and Aurora MySQL 1.x versions are compatible with MySQL 5.6.

Currently supported Aurora MySQL releases are 1.19.5, 1.19.6, 1.22.*, 1.23.*, 2.04.*, 2.07.*, 2.08.*, 2.09.*, 2.10.*, 3.01.* and 3.02.*.

You have the option to upgrade existing Aurora MySQL 2.* database clusters to Aurora MySQL 2.04.5. We do not allow in-place upgrade of Aurora MySQL 1.* clusters. This restriction will be lifted in a later Aurora MySQL 2.* release. You can restore snapshots of Aurora MySQL 1.14.*, 1.15.*, 1.16.*, 1.17.*, 1.18.*, 1.19.*, 2.01.*, 2.02.*, 2.03.* and 2.04.* into Aurora MySQL 2.04.5.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

> **Note**
>
> For information on how to upgrade your Aurora MySQL database cluster, see Upgrading the minor version or patch level of an Aurora MySQL DB cluster in the *Amazon Aurora User Guide*.

Improvements

Security fixes:

- CVE-2016-3518

General fixes:

- Fixed a race condition during storage volume growth that caused the database to restart.
- Fixed an internal communication failure during volume open that caused the database to restart.
- Added DDL recovery support for `ALTER TABLE ALGORITHM=INPLACE` on partitioned tables.
- Fixed an issue with DDL recovery of `ALTER TABLE ALGORITHM=COPY` that caused the database to restart.
- Improved Aurora Replica stability under heavy delete workload on the writer.
- Fixed a database restart caused by a deadlatch between the thread performing full-text search index sync and the thread performing eviction of full-text search table from dictionary cache.
- Fixed a stability issue on the binlog worker during DDL replication while the connection to the binlog master is unstable.
- Fixed an out-of-memory issue in the full-text search code that caused the database to restart.
- Fixed an issue on the Aurora Writer that caused it to restart when the entire 64 tebibyte (TiB) volume is used.
- Fixed a race condition in the Performance Schema feature that caused the database to restart.
- Fixed an issue that caused aborted connections when handling an error in network protocol management.

Comparison with Aurora MySQL version 1

The following Amazon Aurora MySQL features are supported in Aurora MySQL Version 1 (compatible with MySQL 5.6), but these features are currently not supported in Aurora MySQL Version 2 (compatible with MySQL 5.7).

- Asynchronous key prefetch (AKP). For more information, see Optimizing Aurora indexed join queries with asynchronous key prefetch in the *Amazon Aurora User Guide*.
- Hash joins. For more information, see Optimizing large Aurora MySQL join queries with hash joins in the *Amazon Aurora User Guide*.
- Native functions for synchronously invoking AWS Lambda functions. For more information, see Invoking a Lambda function with an Aurora MySQL native function in the *Amazon Aurora User Guide*.
- Scan batching. For more information, see Aurora MySQL database engine updates 2017-12-11 (version 1.16) (Deprecated).
- Migrating data from MySQL using an Amazon S3 bucket. For more information, see Migrating data from MySQL by using an Amazon S3 bucket in the *Amazon Aurora User Guide*.

MySQL 5.7 compatibility

Aurora MySQL 2.04.5 is wire-compatible with MySQL 5.7 and includes features such as JSON support, spatial indexes, and generated columns. Aurora MySQL uses a native implementation of spatial indexing using z-order curves to deliver >20x better write performance and >10x better read performance than MySQL 5.7 for spatial datasets.

Aurora MySQL 2.04.5 does not currently support the following MySQL 5.7 features:

- Group replication plugin
- Increased page size
- InnoDB buffer pool loading at startup
- InnoDB full-text parser plugin
- Multisource replication
- Online buffer pool resizing
- Password validation plugin
- Query rewrite plugins

- Replication filtering
- The CREATE TABLESPACE SQL statement

Aurora MySQL database engine updates 2019-05-29 (version 2.04.4) (Deprecated)

Version: 2.04.4

Aurora MySQL 2.04.4 is generally available. Aurora MySQL 2.x versions are compatible with MySQL 5.7 and Aurora MySQL 1.x versions are compatible with MySQL 5.6.

Currently supported Aurora MySQL releases are 1.19.5, 1.19.6, 1.22.*, 1.23.*, 2.04.*, 2.07.*, 2.08.*, 2.09.*, 2.10.*, 3.01.* and 3.02.*.

When creating a new Aurora MySQL DB cluster (including restoring a snapshot), you have the option of choosing compatibility with either MySQL 5.7 or MySQL 5.6. We do not allow in-place upgrade of Aurora MySQL 1.* clusters or restore of Aurora MySQL 1.* clusters from an Amazon S3 backup into Aurora MySQL 2.04.4. We plan to remove these restrictions in a later Aurora MySQL 2.* release.

You can restore snapshots of Aurora MySQL 1.14.*, 1.15.*, 1.16.*, 1.17.*, 1.18.*, 1.19.*, 2.01.*, 2.02.*, 2.03.*, and 2.04.* into Aurora MySQL 2.04.4.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

> **Note**
>
> This version is currently not available in the AWS GovCloud (US-West) [us-gov-west-1], Europe (Stockholm) [eu-north-1], China (Ningxia) [cn-northwest-1], and Asia Pacific (Hong Kong) [ap-east-1] AWS Regions. There will be a separate announcement once it is made available.

> **Note**
>
> For information on how to upgrade your Aurora MySQL database cluster, see Upgrading the minor version or patch level of an Aurora MySQL DB cluster in the *Amazon Aurora User Guide*.

Improvements

- Fixed an issue that could cause failures when loading data into Aurora from S3.
- Fixed an issue that could cause failures when upload data from Aurora to S3.
- Fixed an issue that caused aborted connections when handling an error in network protocol management.
- Fixed an issue that could cause a crash when dealing with partitioned tables.
- Fixed an issue with the Performance Insights feature being unavailable in some regions.

Comparison with Aurora MySQL version 1

The following Amazon Aurora MySQL features are supported in Aurora MySQL Version 1 (compatible with MySQL 5.6), but these features are currently not supported in Aurora MySQL Version 2 (compatible with MySQL 5.7).

- Asynchronous key prefetch (AKP). For more information, see Optimizing Aurora indexed join queries with asynchronous key prefetch in the *Amazon Aurora User Guide*.
- Hash joins. For more information, see Optimizing large Aurora MySQL join queries with hash joins in the *Amazon Aurora User Guide*.
- Native functions for synchronously invoking AWS Lambda functions. For more information, see Invoking a Lambda function with an Aurora MySQL native function in the *Amazon Aurora User Guide*.
- Scan batching. For more information, see Aurora MySQL database engine updates 2017-12-11 (version 1.16) (Deprecated).
- Migrating data from MySQL using an Amazon S3 bucket. For more information, see Migrating data from MySQL by using an Amazon S3 bucket in the *Amazon Aurora User Guide*.

MySQL 5.7 compatibility

Aurora MySQL 2.04.4 is wire-compatible with MySQL 5.7 and includes features such as JSON support, spatial indexes, and generated columns. Aurora MySQL uses a native implementation of spatial indexing using z-order curves to deliver >20x better write performance and >10x better read performance than MySQL 5.7 for spatial datasets.

Aurora MySQL 2.04.4 does not currently support the following MySQL 5.7 features:

- Group replication plugin
- Increased page size
- InnoDB buffer pool loading at startup
- InnoDB full-text parser plugin
- Multisource replication
- Online buffer pool resizing
- Password validation plugin
- Query rewrite plugins
- Replication filtering
- The `CREATE TABLESPACE` SQL statement

Aurora MySQL database engine updates 2019-05-09 (version 2.04.3) (Deprecated)

Version: 2.04.3

Aurora MySQL 2.04.3 is generally available. Aurora MySQL 2.x versions are compatible with MySQL 5.7 and Aurora MySQL 1.x versions are compatible with MySQL 5.6.

Currently supported Aurora MySQL releases are 1.19.5, 1.19.6, 1.22.*, 1.23.*, 2.04.*, 2.07.*, 2.08.*, 2.09.*, 2.10.*, 3.01.* and 3.02.*.

When creating a new Aurora MySQL DB cluster (including restoring a snapshot), you have the option of choosing compatibility with either MySQL 5.7 or MySQL 5.6. We do not allow in-place upgrade of Aurora MySQL 1.* clusters or restore of Aurora MySQL 1.* clusters from an Amazon S3 backup into Aurora MySQL 2.04.3. We plan to remove these restrictions in a later Aurora MySQL 2.* release.

You can restore snapshots of Aurora MySQL 1.14.*, 1.15.*, 1.16.*, 1.17.*, 1.18.*, 1.19.*, 2.01.*, 2.02.*, 2.03.*, and 2.04.* into Aurora MySQL 2.04.3.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Note

This version is currently not available in the AWS GovCloud (US-West) [us-gov-west-1] and China (Ningxia) [cn-northwest-1] AWS Regions. There will be a separate announcement once it is made available.

Note

For information on how to upgrade your Aurora MySQL database cluster, see Upgrading the minor version or patch level of an Aurora MySQL DB cluster in the *Amazon Aurora User Guide*.

Improvements

- Fixed a bug in binlog replication that can cause an issue on Aurora instances configured as binlog worker.
- Fixed an out-of-memory condition when handling large stored routines.
- Fixed an error in handling certain kinds of `ALTER TABLE` commands.
- Fixed an issue with aborted connections because of an error in network protocol management.

Comparison with Aurora MySQL version 1

The following Amazon Aurora MySQL features are supported in Aurora MySQL Version 1 (compatible with MySQL 5.6), but these features are currently not supported in Aurora MySQL Version 2 (compatible with MySQL 5.7).

- Asynchronous key prefetch (AKP). For more information, see Optimizing Aurora indexed join queries with asynchronous key prefetch in the *Amazon Aurora User Guide*.

- Hash joins. For more information, see Optimizing large Aurora MySQL join queries with hash joins in the *Amazon Aurora User Guide*.
- Native functions for synchronously invoking AWS Lambda functions. For more information, see Invoking a Lambda function with an Aurora MySQL native function in the *Amazon Aurora User Guide*.
- Scan batching. For more information, see Aurora MySQL database engine updates 2017-12-11 (version 1.16) (Deprecated).
- Migrating data from MySQL using an Amazon S3 bucket. For more information, see Migrating data from MySQL by using an Amazon S3 bucket in the *Amazon Aurora User Guide*.

MySQL 5.7 compatibility

Aurora MySQL 2.04.3 is wire-compatible with MySQL 5.7 and includes features such as JSON support, spatial indexes, and generated columns. Aurora MySQL uses a native implementation of spatial indexing using z-order curves to deliver >20x better write performance and >10x better read performance than MySQL 5.7 for spatial datasets.

Aurora MySQL 2.04.3 does not currently support the following MySQL 5.7 features:

- Group replication plugin
- Increased page size
- InnoDB buffer pool loading at startup
- InnoDB full-text parser plugin
- Multisource replication
- Online buffer pool resizing
- Password validation plugin
- Query rewrite plugins
- Replication filtering
- The `CREATE TABLESPACE` SQL statement

Aurora MySQL database engine updates 2019-05-02 (version 2.04.2) (Deprecated)

Version: 2.04.2

Aurora MySQL 2.04.2 is generally available. Aurora MySQL 2.x versions are compatible with MySQL 5.7 and Aurora MySQL 1.x versions are compatible with MySQL 5.6.

Currently supported Aurora MySQL releases are 1.19.5, 1.19.6, 1.22.*, 1.23.*, 2.04.*, 2.07.*, 2.08.*, 2.09.*, 2.10.*, 3.01.* and 3.02.*.

When creating a new Aurora MySQL DB cluster (including restoring a snapshot), you have the option of choosing compatibility with either MySQL 5.7 or MySQL 5.6. We do not allow in-place upgrade of Aurora MySQL 1.* clusters or restore of Aurora MySQL 1.* clusters from an Amazon S3 backup into Aurora MySQL 2.04.2. We plan to remove these restrictions in a later Aurora MySQL 2.* release.

You can restore snapshots of Aurora MySQL 1.14.*, 1.15.*, 1.16.*, 1.17.*, 1.18.*, 1.19.*, 2.01.*, 2.02.*, 2.03.*, 2.04.0, and 2.04.1 into Aurora MySQL 2.04.2.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Note

This version is currently not available in the AWS GovCloud (US-West) [us-gov-west-1] and China (Ningxia) [cn-northwest-1] AWS Regions. There will be a separate announcement once it is made available.

Note

For information on how to upgrade your Aurora MySQL database cluster, see Upgrading the minor version or patch level of an Aurora MySQL DB cluster in the *Amazon Aurora User Guide*.

Improvements

- Added support for SSL binlog replication using custom certificates. For information on using SSL binlog replication in Aurora MySQL, see mysql_rds_import_binlog_ssl_material.
- Fixed a deadlatch on the Aurora primary instance that occurs when a table with a Full Text Search index is being optimized.

- Fixed an issue on the Aurora Replicas where performance of certain queries using `SELECT(*)` could be impacted on tables that have secondary indexes.
- Fixed a condition that resulted in Error 1032 being posted.
- Improved the stability of Aurora Replicas by fixing multiple deadlatches.

Integration of MySQL bug fixes

- Bug #24829050 - INDEX_MERGE_INTERSECTION OPTIMIZATION CAUSES WRONG QUERY RESULTS

Comparison with Aurora MySQL version 1

The following Amazon Aurora MySQL features are supported in Aurora MySQL Version 1 (compatible with MySQL 5.6), but these features are currently not supported in Aurora MySQL Version 2 (compatible with MySQL 5.7).

- Asynchronous key prefetch (AKP). For more information, see Optimizing Aurora indexed join queries with asynchronous key prefetch in the *Amazon Aurora User Guide*.
- Hash joins. For more information, see Optimizing large Aurora MySQL join queries with hash joins in the *Amazon Aurora User Guide*.
- Native functions for synchronously invoking AWS Lambda functions. For more information, see Invoking a Lambda function with an Aurora MySQL native function in the *Amazon Aurora User Guide*.
- Scan batching. For more information, see Aurora MySQL database engine updates 2017-12-11 (version 1.16) (Deprecated).
- Migrating data from MySQL using an Amazon S3 bucket. For more information, see Migrating data from MySQL by using an Amazon S3 bucket in the *Amazon Aurora User Guide*.

MySQL 5.7 compatibility

Aurora MySQL 2.04.2 is wire-compatible with MySQL 5.7 and includes features such as JSON support, spatial indexes, and generated columns. Aurora MySQL uses a native implementation of spatial indexing using z-order curves to deliver >20x better write performance and >10x better read performance than MySQL 5.7 for spatial datasets.

Aurora MySQL 2.04.2 does not currently support the following MySQL 5.7 features:

- Group replication plugin
- Increased page size
- InnoDB buffer pool loading at startup
- InnoDB full-text parser plugin
- Multisource replication
- Online buffer pool resizing
- Password validation plugin
- Query rewrite plugins
- Replication filtering
- The `CREATE TABLESPACE` SQL statement

Aurora MySQL database engine updates 2019-03-25 (version 2.04.1) (Deprecated)

Version: 2.04.1

Aurora MySQL 2.04.1 is generally available. Aurora MySQL 2.x versions are compatible with MySQL 5.7 and Aurora MySQL 1.x versions are compatible with MySQL 5.6.

Currently supported Aurora MySQL releases are 1.19.5, 1.19.6, 1.22.*, 1.23.*, 2.04.*, 2.07.*, 2.08.*, 2.09.*, 2.10.*, 3.01.* and 3.02.*.

When creating a new Aurora MySQL DB cluster (including restoring a snapshot), you have the option of choosing compatibility with either MySQL 5.7 or MySQL 5.6. We do not allow in-place upgrade of Aurora MySQL 1.* clusters or restore of Aurora MySQL 1.* clusters from an Amazon S3 backup into Aurora MySQL 2.04.1. We plan to remove these restrictions in a later Aurora MySQL 2.* release.

You can restore snapshots of Aurora MySQL 1.14.*, 1.15.*, 1.16.*, 1.17.*, 1.18.*, 1.19.*, 2.01.*, 2.02.*, 2.03.*, 2.04.0 into Aurora MySQL 2.04.1.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Note

This version is currently not available in the AWS GovCloud (US-West) [us-gov-west-1] region. There will be a separate announcement once it is made available.

Note

The procedure to upgrade your DB cluster has changed. For more information, see Upgrading the minor version or patch level of an Aurora MySQL DB cluster in the *Amazon Aurora User Guide*.

Improvements

- Fixed an issue where an Aurora MySQL 5.6 snapshot for versions lower than 1.16 could not be restored to the latest Aurora MySQL 5.7 cluster.

Comparison with Aurora MySQL version 1

The following Amazon Aurora MySQL features are supported in Aurora MySQL Version 1 (compatible with MySQL 5.6), but these features are currently not supported in Aurora MySQL Version 2 (compatible with MySQL 5.7).

- Asynchronous key prefetch (AKP). For more information, see Optimizing Aurora indexed join queries with asynchronous key prefetch in the *Amazon Aurora User Guide*.
- Hash joins. For more information, see Optimizing large Aurora MySQL join queries with hash joins in the *Amazon Aurora User Guide*.
- Native functions for synchronously invoking AWS Lambda functions. For more information, see Invoking a Lambda function with an Aurora MySQL native function in the *Amazon Aurora User Guide*.
- Scan batching. For more information, see Aurora MySQL database engine updates 2017-12-11 (version 1.16) (Deprecated).
- Migrating data from MySQL using an Amazon S3 bucket. For more information, see Migrating data from MySQL by using an Amazon S3 bucket in the *Amazon Aurora User Guide*.

MySQL 5.7 compatibility

This Aurora MySQL version is wire-compatible with MySQL 5.7 and includes features such as JSON support, spatial indexes, and generated columns. Aurora MySQL uses a native implementation of spatial indexing using z-order curves to deliver >20x better write performance and >10x better read performance than MySQL 5.7 for spatial datasets.

This Aurora MySQL version does not currently support the following MySQL 5.7 features:

- Group replication plugin
- Increased page size
- InnoDB buffer pool loading at startup
- InnoDB full-text parser plugin
- Multisource replication
- Online buffer pool resizing
- Password validation plugin
- Query rewrite plugins
- Replication filtering
- The `CREATE TABLESPACE` SQL statement

Aurora MySQL database engine updates 2019-03-25 (version 2.04.0) (Deprecated)

Version: 2.04

Aurora MySQL 2.04 is generally available. Aurora MySQL 2.x versions are compatible with MySQL 5.7 and Aurora MySQL 1.x versions are compatible with MySQL 5.6.

Currently supported Aurora MySQL releases are 1.19.5, 1.19.6, 1.22.*, 1.23.*, 2.04.*, 2.07.*, 2.08.*, 2.09.*, 2.10.*, 3.01.* and 3.02.*.

When creating a new Aurora MySQL DB cluster (including restoring a snapshot), you have the option of choosing compatibility with either MySQL 5.7 or MySQL 5.6. We do not allow in-place upgrade of Aurora MySQL 1.* clusters or restore of Aurora MySQL 1.* clusters from an Amazon S3

backup into Aurora MySQL 2.04.0. We plan to remove these restrictions in a later Aurora MySQL 2.* release.

You can restore snapshots of Aurora MySQL 1.19.*, 2.01.*, 2.02.*, and 2.03.* into Aurora MySQL 2.04.0. You cannot restore snapshots of Aurora MySQL 1.14.* or lower, 1.15.*, 1.16.*, 1.17.*, 1.18.* into Aurora MySQL 2.04.0. This restriction is removed in Aurora MySQL 2.04.1.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Note

This version is currently not available in the AWS GovCloud (US-West) [us-gov-west-1] region. There will be a separate announcement once it is made available.

Note

The procedure to upgrade your DB cluster has changed. For more information, see Upgrading the minor version or patch level of an Aurora MySQL DB cluster in the *Amazon Aurora User Guide*.

Improvements

- Supports GTID-based replication. For information about using GTID-based replication with Aurora MySQL, Using GTID-based replication for Aurora MySQL in the *Amazon Aurora User Guide*.
- Fixed an issue where an Aurora Replica incorrectly throws a`Running in read-only mode` error when a statement deleting or updating rows in a temporary table contains an InnoDB subquery.

Integration of MySQL bug fixes

- Bug #26225783: MYSQL CRASH ON CREATE TABLE (REPRODUCEABLE) -> INNODB: ALONG SEMAPHORE WAIT.

Comparison with Aurora MySQL version 1

The following Amazon Aurora MySQL features are supported in Aurora MySQL Version 1 (compatible with MySQL 5.6), but these features are currently not supported in Aurora MySQL Version 2 (compatible with MySQL 5.7).

- Asynchronous key prefetch (AKP). For more information, see Optimizing Aurora indexed join queries with asynchronous key prefetch in the *Amazon Aurora User Guide*.
- Hash joins. For more information, see Optimizing large Aurora MySQL join queries with hash joins in the *Amazon Aurora User Guide*.
- Native functions for synchronously invoking AWS Lambda functions. For more information, see Invoking a Lambda function with an Aurora MySQL native function in the *Amazon Aurora User Guide*.
- Scan batching. For more information, see Aurora MySQL database engine updates 2017-12-11 (version 1.16) (Deprecated).
- Migrating data from MySQL using an Amazon S3 bucket. For more information, see Migrating data from MySQL by using an Amazon S3 bucket in the *Amazon Aurora User Guide*.

MySQL 5.7 compatibility

This Aurora MySQL version is wire-compatible with MySQL 5.7 and includes features such as JSON support, spatial indexes, and generated columns. Aurora MySQL uses a native implementation of spatial indexing using z-order curves to deliver >20x better write performance and >10x better read performance than MySQL 5.7 for spatial datasets.

This Aurora MySQL version does not currently support the following MySQL 5.7 features:

- Group replication plugin
- Increased page size
- InnoDB buffer pool loading at startup
- InnoDB full-text parser plugin
- Multisource replication
- Online buffer pool resizing
- Password validation plugin
- Query rewrite plugins

- Replication filtering
- The CREATE TABLESPACE SQL statement

Aurora MySQL database engine updates 2019-02-07 (version 2.03.4) (Deprecated)

Version: 2.03.4

Aurora MySQL 2.03.4 is generally available. Aurora MySQL 2.x versions are compatible with MySQL 5.7 and Aurora MySQL 1.x versions are compatible with MySQL 5.6.

When creating a new Aurora MySQL DB cluster (including restoring a snapshot), you can choose compatibility with either MySQL 5.7 or MySQL 5.6.

We don't allow in-place upgrade of Aurora MySQL 1.* clusters into Aurora MySQL 2.03.4 or restore to Aurora MySQL 2.03.4 from an Amazon S3 backup. We plan to remove these restrictions in a later Aurora MySQL 2.* release.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Note

This version is currently not available in the AWS GovCloud (US-West) [us-gov-west-1] and China (Beijing) [cn-north-1] regions. There will be a separate announcement once it is made available.

Note

The procedure to upgrade your DB cluster has changed. For more information, see Upgrading the minor version or patch level of an Aurora MySQL DB cluster in the *Amazon Aurora User Guide*.

Improvements

- Support for UTF8MB4 Unicode 9.0 accent-sensitive and case-insensitive collation, `utf8mb4_0900_as_ci`.

Comparison with Aurora MySQL version 1

The following Amazon Aurora MySQL features are supported in Aurora MySQL version 1 (compatible with MySQL 5.6), but these features are currently not supported in Aurora MySQL version 2 (compatible with MySQL 5.7).

- Asynchronous key prefetch (AKP). For more information, see Optimizing Aurora indexed join queries with asynchronous key prefetch in the *Amazon Aurora User Guide*.
- Hash joins. For more information, see Optimizing large Aurora MySQL join queries with hash joins in the *Amazon Aurora User Guide*.
- Native functions for synchronously invoking AWS Lambda functions. For more information, see Invoking a Lambda function with an Aurora MySQL native function in the *Amazon Aurora User Guide*.
- Scan batching. For more information, see Aurora MySQL database engine updates 2017-12-11 (version 1.16) (Deprecated).
- Migrating data from MySQL using an Amazon S3 bucket. For more information, see Migrating data from MySQL by using an Amazon S3 bucket in the *Amazon Aurora User Guide*.

MySQL 5.7 compatibility

Aurora MySQL 2.03.4 is wire-compatible with MySQL 5.7 and includes features such as JSON support, spatial indexes, and generated columns. Aurora MySQL uses a native implementation of spatial indexing using z-order curves to deliver >20x better write performance and >10x better read performance than MySQL 5.7 for spatial datasets.

Aurora MySQL 2.03.4 does not currently support the following MySQL 5.7 features:

- Global transaction identifiers (GTIDs). Aurora MySQL supports GTIDs in version 2.04 and higher.
- Group replication plugin
- Increased page size

- InnoDB buffer pool loading at startup
- InnoDB full-text parser plugin
- Multisource replication
- Online buffer pool resizing
- Password validation plugin
- Query rewrite plugins
- Replication filtering
- The CREATE TABLESPACE SQL statement
- X Protocol

Aurora MySQL database engine updates 2019-01-18 (version 2.03.3) (Deprecated)

Version: 2.03.3

Aurora MySQL 2.03.3 is generally available. Aurora MySQL 2.x versions are compatible with MySQL 5.7 and Aurora MySQL 1.x versions are compatible with MySQL 5.6.

When creating a new Aurora MySQL DB cluster (including restoring a snapshot), you can choose compatibility with either MySQL 5.7 or MySQL 5.6.

We don't allow in-place upgrade of Aurora MySQL 1.* clusters into Aurora MySQL 2.03.3 or restore to Aurora MySQL 2.03.3 from an Amazon S3 backup. We plan to remove these restrictions in a later Aurora MySQL 2.* release.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

> **Note**
>
> This version is currently not available in the AWS GovCloud (US-West) [us-gov-west-1] and China (Beijing) [cn-north-1] regions. There will be a separate announcement once it is made available.

> **Note**
>
> The procedure to upgrade your DB cluster has changed. For more information, see Upgrading the minor version or patch level of an Aurora MySQL DB cluster in the *Amazon Aurora User Guide*.

Improvements

CVE fixes

- CVE-2016-5436

Critical fixes:

- Fixed an issue where an Aurora Replica might become dead-latched when running a backward scan on an index.
- Fixed an issue where an Aurora Replica might restart when the Aurora primary instance runs in-place DDL operations on partitioned tables.
- Fixed an issue where an Aurora Replica might restart during query cache invalidation after a DDL operation on the Aurora primary instance.
- Fixed an issue where an Aurora Replica might restart during a `SELECT` query on a table while the Aurora primary instance runs truncation on that table.
- Fixed a wrong result issue with MyISAM temporary tables where only indexed columns are accessed.
- Fixed an issue in slow logs that generated incorrect large values for `query_time` and `lock_time` periodically after approximately 40,000 queries.
- Fixed an issue where a schema named "tmp" could cause migration from RDS for MySQL to Aurora MySQL to become stuck.
- Fixed an issue where the audit log might have missing events during log rotation.
- Fixed an issue where the Aurora primary instance restored from an Aurora 5.6 snapshot might restart when the Fast DDL feature in the lab mode is enabled.
- Fixed an issue where the CPU usage is 100% caused by the dictionary stats thread.
- Fixed an issue where an Aurora Replica might restart when running a `CHECK TABLE` statement.

Integration of MySQL bug fixes

- Bug #25361251: INCORRECT BEHAVIOR WITH INSERT ON DUPLICATE KEY IN SP
- Bug #26734162: INCORRECT BEHAVIOR WITH INSERT OF BLOB + ON DUPLICATE KEY UPDATE
- Bug #27460607: INCORRECT BEHAVIOR OF IODKU WHEN INSERT SELECT's SOURCE TABLE IS EMPTY
- A query using a `DISTINCT` or `GROUP BY` clause could return incorrect results. (MySQL 5.7 Bug #79591, Bug #22343910)
- A `DELETE` from joined tables using a derived table in the `WHERE` clause fails with error 1093 (Bug #23074801).
- GCOLS: INCORRECT BEHAVIOR WITH CHARSET CHANGES (Bug #25287633).

Comparison with Aurora MySQL version 1

The following Amazon Aurora MySQL features are supported in Aurora MySQL version 1 (compatible with MySQL 5.6), but these features are currently not supported in Aurora MySQL version 2 (compatible with MySQL 5.7).

- Asynchronous key prefetch (AKP). For more information, see Optimizing Aurora indexed join queries with asynchronous key prefetch in the *Amazon Aurora User Guide*.
- Hash joins. For more information, see Optimizing large Aurora MySQL join queries with hash joins in the *Amazon Aurora User Guide*.
- Native functions for synchronously invoking AWS Lambda functions. For more information, see Invoking a Lambda function with an Aurora MySQL native function in the *Amazon Aurora User Guide*.
- Scan batching. For more information, see Aurora MySQL database engine updates 2017-12-11 (version 1.16) (Deprecated).
- Migrating data from MySQL using an Amazon S3 bucket. For more information, see Migrating data from MySQL by using an Amazon S3 bucket in the *Amazon Aurora User Guide*.

MySQL 5.7 compatibility

Aurora MySQL 2.03.3 is wire-compatible with MySQL 5.7 and includes features such as JSON support, spatial indexes, and generated columns. Aurora MySQL uses a native implementation of

spatial indexing using z-order curves to deliver >20x better write performance and >10x better read performance than MySQL 5.7 for spatial datasets.

Aurora MySQL 2.03.3 does not currently support the following MySQL 5.7 features:

- Global transaction identifiers (GTIDs). Aurora MySQL supports GTIDs in version 2.04 and higher.
- Group replication plugin
- Increased page size
- InnoDB buffer pool loading at startup
- InnoDB full-text parser plugin
- Multisource replication
- Online buffer pool resizing
- Password validation plugin
- Query rewrite plugins
- Replication filtering
- The `CREATE TABLESPACE` SQL statement
- X Protocol

Aurora MySQL database engine updates 2019-01-09 (version 2.03.2) (Deprecated)

Version: 2.03.2

Aurora MySQL 2.03.2 is generally available. Aurora MySQL 2.x versions are compatible with MySQL 5.7 and Aurora MySQL 1.x versions are compatible with MySQL 5.6.

When creating a new Aurora MySQL DB cluster (including restoring a snapshot), you can choose compatibility with either MySQL 5.7 or MySQL 5.6.

We don't allow in-place upgrade of Aurora MySQL 1.* clusters into Aurora MySQL 2.03.2 or restore to Aurora MySQL 2.03.2 from an Amazon S3 backup. We plan to remove these restrictions in a later Aurora MySQL 2.* release.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Note

This version is currently not available in the AWS GovCloud (US-West) [us-gov-west-1] and China (Beijing) [cn-north-1] regions. There will be a separate announcement once it is made available.

Note

The procedure to upgrade your DB cluster has changed. For more information, see Upgrading the minor version or patch level of an Aurora MySQL DB cluster in the *Amazon Aurora User Guide*.

Improvements

- **Aurora Version Selector** – Starting with Aurora MySQL 2.03.2, you can choose from among multiple versions of MySQL 5.7-compatible Aurora on the AWS Management Console. For more information, see Checking or specifying Aurora MySQL engine versions through AWS in the *Amazon Aurora User Guide*.

Critical fixes:

- CVE-2016-3495

Comparison with Aurora MySQL version 1

The following Amazon Aurora MySQL features are supported in Aurora MySQL version 1 (compatible with MySQL 5.6), but these features are currently not supported in Aurora MySQL version 2 (compatible with MySQL 5.7).

- Asynchronous key prefetch (AKP). For more information, see Optimizing Aurora indexed join queries with asynchronous key prefetch in the *Amazon Aurora User Guide*.
- Hash joins. For more information, see Optimizing large Aurora MySQL join queries with hash joins in the *Amazon Aurora User Guide*.

- Native functions for synchronously invoking AWS Lambda functions. For more information, see Invoking a Lambda function with an Aurora MySQL native function in the *Amazon Aurora User Guide*.
- Scan batching. For more information, see Aurora MySQL database engine updates 2017-12-11 (version 1.16) (Deprecated).
- Migrating data from MySQL using an Amazon S3 bucket. For more information, see Migrating data from MySQL by using an Amazon S3 bucket in the *Amazon Aurora User Guide*.

MySQL 5.7 compatibility

Aurora MySQL 2.03.2 is wire-compatible with MySQL 5.7 and includes features such as JSON support, spatial indexes, and generated columns. Aurora MySQL uses a native implementation of spatial indexing using z-order curves to deliver >20x better write performance and >10x better read performance than MySQL 5.7 for spatial datasets.

Aurora MySQL 2.03.2 does not currently support the following MySQL 5.7 features:

- Global transaction identifiers (GTIDs). Aurora MySQL supports GTIDs in version 2.04 and higher.
- Group replication plugin
- Increased page size
- InnoDB buffer pool loading at startup
- InnoDB full-text parser plugin
- Multisource replication
- Online buffer pool resizing
- Password validation plugin
- Query rewrite plugins
- Replication filtering
- The `CREATE TABLESPACE` SQL statement
- X Protocol

Aurora MySQL database engine updates 2018-10-24 (version 2.03.1) (Deprecated)

Version: 2.03.1

Aurora MySQL 2.03.1 is generally available. Aurora MySQL 2.x versions are compatible with MySQL 5.7 and Aurora MySQL 1.x versions are compatible with MySQL 5.6.

When creating a new Aurora MySQL DB cluster, you can choose compatibility with either MySQL 5.7 or MySQL 5.6. When restoring a MySQL 5.6-compatible snapshot, you can choose compatibility with either MySQL 5.7 or MySQL 5.6.

You can restore snapshots of Aurora MySQL 1.14.*, 1.15.*, 1.16.*, 1.17.*, 1.18.*, 2.01.*, 2.02.*, and 2.03 into Aurora MySQL 2.03.1.

We don't allow in-place upgrade of Aurora MySQL 1.* clusters into Aurora MySQL 2.03.1 or restore to Aurora MySQL 2.03.1 from an Amazon S3 backup. We plan to remove these restrictions in a later Aurora MySQL 2.* release.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

> **Note**
>
> This version is currently not available in the AWS GovCloud (US-West) [us-gov-west-1] and China (Beijing) [cn-north-1] regions. There will be a separate announcement once it is made available.

Improvements

- Fix an issue where the Aurora Writer might restart when running transaction deadlock detection.

Comparison with Aurora MySQL version 1

The following Amazon Aurora MySQL features are supported in Aurora MySQL version 1 (compatible with MySQL 5.6), but these features are currently not supported in Aurora MySQL version 2 (compatible with MySQL 5.7).

- Asynchronous key prefetch (AKP). For more information, see Optimizing Aurora indexed join queries with asynchronous key prefetch in the *Amazon Aurora User Guide*.
- Hash joins. For more information, see Optimizing large Aurora MySQL join queries with hash joins in the *Amazon Aurora User Guide*.

- Native functions for synchronously invoking AWS Lambda functions. For more information, see Invoking a Lambda function with an Aurora MySQL native function in the *Amazon Aurora User Guide*.
- Scan batching. For more information, see Aurora MySQL database engine updates 2017-12-11 (version 1.16) (Deprecated).
- Migrating data from MySQL using an Amazon S3 bucket. For more information, see Migrating data from MySQL by using an Amazon S3 bucket in the *Amazon Aurora User Guide*.

MySQL 5.7 compatibility

Aurora MySQL 2.03.1 is wire-compatible with MySQL 5.7 and includes features such as JSON support, spatial indexes, and generated columns. Aurora MySQL uses a native implementation of spatial indexing using z-order curves to deliver >20x better write performance and >10x better read performance than MySQL 5.7 for spatial datasets.

Aurora MySQL 2.03.1 does not currently support the following MySQL 5.7 features:

- Global transaction identifiers (GTIDs). Aurora MySQL supports GTIDs in version 2.04 and higher.
- Group replication plugin
- Increased page size
- InnoDB buffer pool loading at startup
- InnoDB full-text parser plugin
- Multisource replication
- Online buffer pool resizing
- Password validation plugin
- Query rewrite plugins
- Replication filtering
- The `CREATE TABLESPACE` SQL statement
- X Protocol

Aurora MySQL database engine updates 2018-10-11 (version 2.03) (Deprecated)

Version: 2.03

Aurora MySQL 2.03 is generally available. Aurora MySQL 2.x versions are compatible with MySQL 5.7 and Aurora MySQL 1.x versions are compatible with MySQL 5.6.

When creating a new Aurora MySQL DB cluster, you can choose compatibility with either MySQL 5.7 or MySQL 5.6. When restoring a MySQL 5.6-compatible snapshot, you can choose compatibility with either MySQL 5.7 or MySQL 5.6.

You can restore snapshots of Aurora MySQL 1.14.*, 1.15.*, 1.16.*, 1.17.*, 1.18.*, 2.01.*, and 2.02.* into Aurora MySQL 2.03.

We don't allow in-place upgrade of Aurora MySQL 1.* clusters into Aurora MySQL 2.03 or restore to Aurora MySQL 2.03 from an Amazon S3 backup. We plan to remove these restrictions in a later Aurora MySQL 2.* release.

> **Note**
>
> This version is currently not available in the AWS GovCloud (US-West) [us-gov-west-1] and China (Beijing) [cn-north-1] regions. There will be a separate announcement once it is made available.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Improvements

- Performance schema is available.
- Fixed an issue where zombie sessions with killed state might consume more CPU.
- Fixed a dead latch issue when a read-only transaction is acquiring a lock on a record on the Aurora Writer.
- Fixed an issue where the Aurora Replica without customer workload might have high CPU utilization.
- Multiple fixes on issues that might cause the Aurora Replica or the Aurora writer to restart.
- Added capability to skip diagnostic logging when the disk throughput limit is reached.
- Fixed a memory leak issue when binlog is enabled on the Aurora Writer.

Integration of MySQL community edition bug fixes

- REVERSE SCAN ON A PARTITIONED TABLE DOES ICP - ORDER BY DESC (Bug #24929748).
- JSON_OBJECT CREATES INVALID JSON CODE (Bug#26867509).
- INSERTING LARGE JSON DATA TAKES AN INORDINATE AMOUNT OF TIME (Bug #22843444).
- PARTITIONED TABLES USE MORE MEMORY IN 5.7 THAN 5.6 (Bug #25080442).

Comparison with Aurora MySQL version 1

The following Amazon Aurora MySQL features are supported in Aurora MySQL version 1 (compatible with MySQL 5.6), but these features are currently not supported in Aurora MySQL version 2 (compatible with MySQL 5.7).

- Asynchronous key prefetch (AKP). For more information, see Optimizing Aurora indexed join queries with asynchronous key prefetch in the *Amazon Aurora User Guide*.
- Hash joins. For more information, see Optimizing large Aurora MySQL join queries with hash joins in the *Amazon Aurora User Guide*.
- Native functions for synchronously invoking AWS Lambda functions. For more information, see Invoking a Lambda function with an Aurora MySQL native function in the *Amazon Aurora User Guide*.
- Scan batching. For more information, see Aurora MySQL database engine updates 2017-12-11 (version 1.16) (Deprecated).
- Migrating data from MySQL using an Amazon S3 bucket. For more information, see Migrating data from MySQL by using an Amazon S3 bucket in the *Amazon Aurora User Guide*.

MySQL 5.7 compatibility

Aurora MySQL 2.03 is wire-compatible with MySQL 5.7 and includes features such as JSON support, spatial indexes, and generated columns. Aurora MySQL uses a native implementation of spatial indexing using z-order curves to deliver >20x better write performance and >10x better read performance than MySQL 5.7 for spatial datasets.

Aurora MySQL 2.03 does not currently support the following MySQL 5.7 features:

- Global transaction identifiers (GTIDs). Aurora MySQL supports GTIDs in version 2.04 and higher.

- Group replication plugin
- Increased page size
- InnoDB buffer pool loading at startup
- InnoDB full-text parser plugin
- Multisource replication
- Online buffer pool resizing
- Password validation plugin
- Query rewrite plugins
- Replication filtering
- The CREATE TABLESPACE SQL statement
- X Protocol

Aurora MySQL database engine updates 2018-10-08 (version 2.02.5) (Deprecated)

Version: 2.02.5

Aurora MySQL 2.02.5 is generally available. Aurora MySQL 2.x versions are compatible with MySQL 5.7 and Aurora MySQL 1.x versions are compatible with MySQL 5.6.

When creating a new Aurora MySQL DB cluster, you can choose compatibility with either MySQL 5.7 or MySQL 5.6. When restoring a MySQL 5.6-compatible snapshot, you can choose compatibility with either MySQL 5.7 or MySQL 5.6.

You can restore snapshots of Aurora MySQL 1.14.*, 1.15.*, 1.16.*, 1.17.*, 1.18.*, 2.01.*, and 2.02.* into Aurora MySQL 2.02.5. You can also perform an in-place upgrade from Aurora MySQL 2.01.* or 2.02.* to Aurora MySQL 2.02.5.

We don't allow in-place upgrade of Aurora MySQL 1.* clusters into Aurora MySQL 2.02.5 or restore to Aurora MySQL 2.02.5 from an Amazon S3 backup. We plan to remove these restrictions in a later Aurora MySQL 2.* release.

The performance schema is disabled for this release of Aurora MySQL 5.7. Upgrade to Aurora 2.03 for performance schema support.

> **Note**
>
> This version is currently not available in the AWS GovCloud (US-West) [us-gov-west-1] and China (Beijing) [cn-north-1] regions. There will be a separate announcement once it is made available.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Improvements

- Fix an issue where an Aurora Replica might restart when it is doing a reverse scan on a table.

Comparison with Aurora MySQL version 1

The following Amazon Aurora MySQL features are supported in Aurora MySQL version 1 (compatible with MySQL 5.6), but these features are currently not supported in Aurora MySQL version 2 (compatible with MySQL 5.7).

- Asynchronous key prefetch (AKP). For more information, see Optimizing Aurora indexed join queries with asynchronous key prefetch in the *Amazon Aurora User Guide*.
- Hash joins. For more information, see Optimizing large Aurora MySQL join queries with hash joins in the *Amazon Aurora User Guide*.
- Native functions for synchronously invoking AWS Lambda functions. For more information, see Invoking a Lambda function with an Aurora MySQL native function in the *Amazon Aurora User Guide*.
- Scan batching. For more information, see Aurora MySQL database engine updates 2017-12-11 (version 1.16) (Deprecated).
- Migrating data from MySQL using an Amazon S3 bucket. For more information, see Migrating data from MySQL by using an Amazon S3 bucket in the *Amazon Aurora User Guide*.

MySQL 5.7 compatibility

Aurora MySQL 2.02.5 is wire-compatible with MySQL 5.7 and includes features such as JSON support, spatial indexes, and generated columns. Aurora MySQL uses a native implementation of spatial indexing using z-order curves to deliver >20x better write performance and >10x better read performance than MySQL 5.7 for spatial datasets.

Aurora MySQL 2.02.5 does not currently support the following MySQL 5.7 features:

- Global transaction identifiers (GTIDs). Aurora MySQL supports GTIDs in version 2.04 and higher.
- Group replication plugin
- Increased page size
- InnoDB buffer pool loading at startup
- InnoDB full-text parser plugin
- Multisource replication
- Online buffer pool resizing
- Password validation plugin
- Query rewrite plugins
- Replication filtering
- The `CREATE TABLESPACE` SQL statement
- X Protocol

Aurora MySQL database engine updates 2018-09-21 (version 2.02.4) (Deprecated)

Version: 2.02.4

Aurora MySQL 2.02.4 is generally available. Aurora MySQL 2.x versions are compatible with MySQL 5.7 and Aurora MySQL 1.x versions are compatible with MySQL 5.6.

When creating a new Aurora MySQL DB cluster, you can choose compatibility with either MySQL 5.7 or MySQL 5.6. When restoring a MySQL 5.6-compatible snapshot, you can choose compatibility with either MySQL 5.7 or MySQL 5.6.

You can restore snapshots of Aurora MySQL 1.14.*, 1.15.*, 1.16.*, 1.17.*, 1.18.*, 2.01.*, and 2.02.* into Aurora MySQL 2.02.4. You can also perform an in-place upgrade from Aurora MySQL 2.01.* or 2.02.* to Aurora MySQL 2.02.4.

We don't allow in-place upgrade of Aurora MySQL 1.* clusters into Aurora MySQL 2.02.4 or restore to Aurora MySQL 2.02.4 from an Amazon S3 backup. We plan to remove these restrictions in a later Aurora MySQL 2.* release.

The performance schema is disabled for this release of Aurora MySQL 5.7. Upgrade to Aurora 2.03 for performance schema support.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Improvements

- Fixed a stability issue related to Full Text Search indexes on tables restored from an Aurora MySQL 5.6 snapshot.

Integration of MySQL community edition bug fixes

- BUG#13651665 INNODB MAY BE UNABLE TO LOAD TABLE DEFINITION AFTER RENAME
- BUG#21371070 INNODB: CANNOT ALLOCATE 0 BYTES.
- BUG#21378944 FTS ASSERT ENC.SRC_ILIST_PTR != NULL, FTS_OPTIMIZE_WORD(), OPTIMIZE TABLE
- BUG#21508537 ASSERTION FAILURE UT_A(!VICTIM_TRX->READ_ONLY)
- BUG#21983865 UNEXPECTED DEADLOCK WITH INNODB_AUTOINC_LOCK_MODE=0
- BUG#22679185 INVALID INNODB FTS DOC ID DURING INSERT
- BUG#22899305 GCOLS: ASSERTION: !(COL->PRTYPE & 256).
- BUG#22956469 MEMORY LEAK INTRODUCED IN 5.7.8 IN MEMORY/INNODB/OS0FILE
- BUG#22996488 CRASH IN FTS_SYNC_INDEX WHEN DOING DDL IN A LOOP
- BUG#23014521 GCOL:INNODB: ASSERTION: !IS_V
- BUG#23021168 REPLICATION STOPS AFTER TRX IS ROLLED BACK ASYNC
- BUG#23052231 ASSERTION: ADD_AUTOINC < DICT_TABLE_GET_N_USER_COLS

- BUG#23149683 ROTATE INNODB MASTER KEY WITH KEYRING_OKV_CONF_DIR MISSING: SIGSEGV; SIGNAL 11
- BUG#23762382 INSERT VALUES QUERY WITH JOIN IN A SELECT CAUSES INCORRECT BEHAVIOR
- BUG#25209512 CURRENT_TIMESTAMP PRODUCES ZEROS IN TRIGGER
- BUG#26626277 BUG IN "INSERT... ON DUPLICATE KEY UPDATE" QUERY
- BUG#26734162 INCORRECT BEHAVIOR WITH INSERT OF BLOB + ON DUPLICATE KEY UPDATE
- BUG#27460607 INCORRECT WHEN INSERT SELECT's SOURCE TABLE IS EMPTY

Comparison with Aurora MySQL version 1

The following Amazon Aurora MySQL features are supported in Aurora MySQL version 1 (compatible with MySQL 5.6), but these features are currently not supported in Aurora MySQL version 2 (compatible with MySQL 5.7).

- Asynchronous key prefetch (AKP). For more information, see Optimizing Aurora indexed join queries with asynchronous key prefetch in the *Amazon Aurora User Guide*.
- Hash joins. For more information, see Optimizing large Aurora MySQL join queries with hash joins in the *Amazon Aurora User Guide*.
- Native functions for synchronously invoking AWS Lambda functions. For more information, see Invoking a Lambda function with an Aurora MySQL native function in the *Amazon Aurora User Guide*.
- Scan batching. For more information, see Aurora MySQL database engine updates 2017-12-11 (version 1.16) (Deprecated).
- Migrating data from MySQL using an Amazon S3 bucket. For more information, see Migrating data from MySQL by using an Amazon S3 bucket in the *Amazon Aurora User Guide*.

MySQL 5.7 compatibility

Aurora MySQL 2.02.4 is wire-compatible with MySQL 5.7 and includes features such as JSON support, spatial indexes, and generated columns. Aurora MySQL uses a native implementation of spatial indexing using z-order curves to deliver >20x better write performance and >10x better read performance than MySQL 5.7 for spatial datasets.

Aurora MySQL 2.02.4 does not currently support the following MySQL 5.7 features:

- Global transaction identifiers (GTIDs). Aurora MySQL supports GTIDs in version 2.04 and higher.
- Group replication plugin
- Increased page size
- InnoDB buffer pool loading at startup
- InnoDB full-text parser plugin
- Multisource replication
- Online buffer pool resizing
- Password validation plugin
- Query rewrite plugins
- Replication filtering
- The `CREATE TABLESPACE` SQL statement
- X Protocol

Aurora MySQL database engine updates 2018-08-23 (version 2.02.3) (Deprecated)

Version: 2.02.3

Aurora MySQL 2.02.3 is generally available. Aurora MySQL 2.x versions are compatible with MySQL 5.7 and Aurora MySQL 1.x versions are compatible with MySQL 5.6.

When creating a new Aurora MySQL DB cluster, you can choose compatibility with either MySQL 5.7 or MySQL 5.6. When restoring a MySQL 5.6-compatible snapshot, you can choose compatibility with either MySQL 5.7 or MySQL 5.6.

You can restore snapshots of Aurora MySQL 1.14.*, 1.15.*, 1.16.*, 1.17.*, 2.01.*, and 2.02.* into Aurora MySQL 2.02.3. You can also perform an in-place upgrade from Aurora MySQL 2.01.* or 2.02.* to Aurora MySQL 2.02.3.

We don't allow in-place upgrade of Aurora MySQL 1.* clusters into Aurora MySQL 2.02.3 or restore to Aurora MySQL 2.02.3 from an Amazon S3 backup. We plan to remove these restrictions in a later Aurora MySQL 2.* release.

The performance schema is disabled for this release of Aurora MySQL 5.7. Upgrade to Aurora 2.03 for performance schema support.

> **Note**
>
> This version is currently not available in the AWS GovCloud (US-West) [us-gov-west-1] and China (Beijing) [cn-north-1] regions. There will be a separate announcement once it is made available.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Comparison with Aurora MySQL version 1

The following Amazon Aurora MySQL features are supported in Aurora MySQL version 1 (compatible with MySQL 5.6), but these features are currently not supported in Aurora MySQL version 2 (compatible with MySQL 5.7).

- Asynchronous key prefetch (AKP). For more information, see Optimizing Aurora indexed join queries with asynchronous key prefetch in the *Amazon Aurora User Guide*.
- Hash joins. For more information, see Optimizing large Aurora MySQL join queries with hash joins in the *Amazon Aurora User Guide*.
- Native functions for synchronously invoking AWS Lambda functions. For more information, see Invoking a Lambda function with an Aurora MySQL native function in the *Amazon Aurora User Guide*.
- Scan batching. For more information, see Aurora MySQL database engine updates 2017-12-11 (version 1.16) (Deprecated).
- Migrating data from MySQL using an Amazon S3 bucket. For more information, see Migrating data from MySQL by using an Amazon S3 bucket in the *Amazon Aurora User Guide*.

Currently, Aurora MySQL 2.01 does not support features added in Aurora MySQL version 1.16 and later. For information about Aurora MySQL version 1.16, see Aurora MySQL database engine updates 2017-12-11 (version 1.16) (Deprecated).

MySQL 5.7 compatibility

Aurora MySQL 2.02.3 is wire-compatible with MySQL 5.7 and includes features such as JSON support, spatial indexes, and generated columns. Aurora MySQL uses a native implementation of spatial indexing using z-order curves to deliver >20x better write performance and >10x better read performance than MySQL 5.7 for spatial datasets.

Aurora MySQL 2.02.3 does not currently support the following MySQL 5.7 features:

- Global transaction identifiers (GTIDs). Aurora MySQL supports GTIDs in version 2.04 and higher.
- Group replication plugin
- Increased page size
- InnoDB buffer pool loading at startup
- InnoDB full-text parser plugin
- Multisource replication
- Online buffer pool resizing
- Password validation plugin
- Query rewrite plugins
- Replication filtering
- The `CREATE TABLESPACE` SQL statement
- X Protocol

CLI differences between Aurora MySQL 2.x and Aurora MySQL 1.x

- The engine name for Aurora MySQL 2.x is `aurora-mysql`; the engine name for Aurora MySQL 1.x continues to be `aurora`.
- The default parameter group for Aurora MySQL 2.x is `default.aurora-mysql5.7`; the default parameter group for Aurora MySQL 1.x continues to be `default.aurora5.6`.
- The DB cluster parameter group family name for Aurora MySQL 2.x is `aurora-mysql5.7`; the DB cluster parameter group family name for Aurora MySQL 1.x continues to be `aurora5.6`.

Refer to the Aurora documentation for the full set of CLI commands and differences between Aurora MySQL 2.x and Aurora MySQL 1.x.

Improvements

- Fixed an issue where an Aurora Replica can restart when using optimistic cursor restores while reading records.
- Updated the default value of the parameter `innodb_stats_persistent_sample_pages` to 128 to improve index statistics.
- Fixed an issue where an Aurora Replica might restart when it accesses a small table that is being concurrently modified on the Aurora primary instance.
- Fixed `ANALYZE TABLE` to stop flushing the table definition cache.
- Fixed an issue where the Aurora primary instance or an Aurora Replica might restart when converting a point query for geospatial to a search range.

Aurora MySQL database engine updates 2018-06-04 (version 2.02.2) (Deprecated)

Version: 2.02.2

Aurora MySQL 2.02.2 is generally available. Aurora MySQL 2.x versions are compatible with MySQL 5.7 and Aurora MySQL 1.x versions are compatible with MySQL 5.6.

When creating a new Aurora MySQL DB cluster, you can choose compatibility with either MySQL 5.7 or MySQL 5.6. When restoring a MySQL 5.6-compatible snapshot, you can choose compatibility with either MySQL 5.7 or MySQL 5.6.

You can restore snapshots of Aurora MySQL 1.14*, 1.15*, 1.16*, 1.17*, 2.01*, and 2.02 into Aurora MySQL 2.02.2. You can also perform an in-place upgrade from Aurora MySQL 2.01* or 2.02 to Aurora MySQL 2.02.2.

We don't allow in-place upgrade of Aurora MySQL 1.* clusters into Aurora MySQL 2.02.2 or restore to Aurora MySQL 2.02.2 from an Amazon S3 backup. We plan to remove these restrictions in a later Aurora MySQL 2.* release.

The performance schema is disabled for this release of Aurora MySQL 5.7. Upgrade to Aurora 2.03 for performance schema support.

> **Note**
>
> This version is currently not available in the AWS GovCloud (US-West) [us-gov-west-1] and China (Beijing) [cn-north-1] regions. There will be a separate announcement once it is made available.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Improvements

Critical fixes:

- CVE-2016-3486

Comparison with Aurora MySQL 5.6

The following Amazon Aurora MySQL features are supported in Aurora MySQL 5.6, but these features are currently not supported in Aurora MySQL 5.7.

- Asynchronous key prefetch (AKP). For more information, see Optimizing Aurora indexed join queries with asynchronous key prefetch in the *Amazon Aurora User Guide*.
- Hash joins. For more information, see Optimizing large Aurora MySQL join queries with hash joins in the *Amazon Aurora User Guide*.
- Native functions for synchronously invoking AWS Lambda functions. For more information, see Invoking a Lambda function with an Aurora MySQL native function in the *Amazon Aurora User Guide*.
- Scan batching. For more information, see Aurora MySQL database engine updates 2017-12-11 (version 1.16) (Deprecated).
- Migrating data from MySQL using an Amazon S3 bucket. For more information, see Migrating data from MySQL by using an Amazon S3 bucket in the *Amazon Aurora User Guide*.

Currently, Aurora MySQL 2.01 does not support features added in Aurora MySQL version 1.16 and later. For information about Aurora MySQL version 1.16, see Aurora MySQL database engine updates 2017-12-11 (version 1.16) (Deprecated).

MySQL 5.7 compatibility

Aurora MySQL 2.02.2 is wire-compatible with MySQL 5.7 and includes features such as JSON support, spatial indexes, and generated columns. Aurora MySQL uses a native implementation of spatial indexing using z-order curves to deliver >20x better write performance and >10x better read performance than MySQL 5.7 for spatial datasets.

Aurora MySQL 2.02.2 does not currently support the following MySQL 5.7 features:

- Global transaction identifiers (GTIDs). Aurora MySQL supports GTIDs in version 2.04 and higher.
- Group replication plugin
- Increased page size
- InnoDB buffer pool loading at startup
- InnoDB full-text parser plugin
- Multisource replication
- Online buffer pool resizing
- Password validation plugin
- Query rewrite plugins
- Replication filtering
- The `CREATE TABLESPACE` SQL statement
- X Protocol

CLI differences between Aurora MySQL 2.x and Aurora MySQL 1.x

- The engine name for Aurora MySQL 2.x is `aurora-mysql`; the engine name for Aurora MySQL 1.x continues to be `aurora`.
- The default parameter group for Aurora MySQL 2.x is `default.aurora-mysql5.7`; the default parameter group for Aurora MySQL 1.x continues to be `default.aurora5.6`.
- The DB cluster parameter group family name for Aurora MySQL 2.x is `aurora-mysql5.7`; the DB cluster parameter group family name for Aurora MySQL 1.x continues to be `aurora5.6`.

Refer to the Aurora documentation for the full set of CLI commands and differences between Aurora MySQL 2.x and Aurora MySQL 1.x.

Improvements

- Fixed an issue where an Aurora Writer can occasionally restart when tracking Aurora Replica progress.
- Fixed an issue where an Aurora Replica restarts or throws an error when a partitioned table is accessed after running index create or drop statements on the table on the Aurora Writer.
- Fixed an issue where a table on an Aurora Replica is inaccessible while it is applying the changes caused by running ALTER table ADD/DROP column statements on the Aurora Writer.

Aurora MySQL database engine updates 2018-05-03 (version 2.02) (Deprecated)

Version: 2.02

Aurora MySQL 2.02 is generally available. Aurora MySQL 2.x versions are compatible with MySQL 5.7 and Aurora MySQL 1.x versions are compatible with MySQL 5.6.

When creating a new Aurora MySQL DB cluster, you can choose compatibility with either MySQL 5.7 or MySQL 5.6. When restoring a MySQL 5.6-compatible snapshot, you can choose compatibility with either MySQL 5.7 or MySQL 5.6.

You can restore snapshots of Aurora MySQL 1.14*, 1.15*, 1.16*, 1.17* and 2.01* into Aurora MySQL 2.02. You can also perform an in-place upgrade from Aurora MySQL 2.01* to Aurora MySQL 2.02.

We don't allow in-place upgrade of Aurora MySQL 1.x clusters into Aurora MySQL 2.02 or restore to Aurora MySQL 2.02 from an Amazon S3 backup. We plan to remove these restrictions in a later Aurora MySQL 2.x release.

The performance schema is disabled for this release of Aurora MySQL 5.7. Upgrade to Aurora 2.03 for performance schema support.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Comparison with Aurora MySQL 5.6

The following Amazon Aurora MySQL features are supported in Aurora MySQL 5.6, but these features are currently not supported in Aurora MySQL 5.7.

- Asynchronous key prefetch (AKP). For more information, see Optimizing Aurora indexed join queries with asynchronous key prefetch in the *Amazon Aurora User Guide*.
- Hash joins. For more information, see Optimizing large Aurora MySQL join queries with hash joins in the *Amazon Aurora User Guide*.
- Native functions for synchronously invoking AWS Lambda functions. For more information, see Invoking a Lambda function with an Aurora MySQL native function in the *Amazon Aurora User Guide*.
- Scan batching. For more information, see Aurora MySQL database engine updates 2017-12-11 (version 1.16) (Deprecated).
- Migrating data from MySQL using an Amazon S3 bucket. For more information, see Migrating data from MySQL by using an Amazon S3 bucket in the *Amazon Aurora User Guide*.

Currently, Aurora MySQL 2.01 does not support features added in Aurora MySQL version 1.16 and later. For information about Aurora MySQL version 1.16, see Aurora MySQL database engine updates 2017-12-11 (version 1.16) (Deprecated).

MySQL 5.7 compatibility

Aurora MySQL 2.02 is wire-compatible with MySQL 5.7 and includes features such as JSON support, spatial indexes, and generated columns. Aurora MySQL uses a native implementation of spatial indexing using z-order curves to deliver >20x better write performance and >10x better read performance than MySQL 5.7 for spatial datasets.

Aurora MySQL 2.02 does not currently support the following MySQL 5.7 features:

- Global transaction identifiers (GTIDs). Aurora MySQL supports GTIDs in version 2.04 and higher.
- Group replication plugin
- Increased page size
- InnoDB buffer pool loading at startup
- InnoDB full-text parser plugin
- Multisource replication
- Online buffer pool resizing
- Password validation plugin
- Query rewrite plugins

- Replication filtering
- The CREATE TABLESPACE SQL statement
- X Protocol

CLI differences between Aurora MySQL 2.x and Aurora MySQL 1.x

- The engine name for Aurora MySQL 2.x is `aurora-mysql`; the engine name for Aurora MySQL 1.x continues to be `aurora`.
- The default parameter group for Aurora MySQL 2.x is `default.aurora-mysql5.7`; the default parameter group for Aurora MySQL 1.x continues to be `default.aurora5.6`.
- The DB cluster parameter group family name for Aurora MySQL 2.x is `aurora-mysql5.7`; the DB cluster parameter group family name for Aurora MySQL 1.x continues to be `aurora5.6`.

Refer to the Aurora documentation for the full set of CLI commands and differences between Aurora MySQL 2.x and Aurora MySQL 1.x.

Improvements

- Fixed an issue where an Aurora Writer restarts when running INSERT statements and exploiting the Fast Insert optimization.
- Fixed an issue where an Aurora Replica restarts when running ALTER DATABASE statements on the Aurora Replica.
- Fixed an issue where an Aurora Replica restarts when running queries on tables that have just been dropped on the Aurora Writer.
- Fixed an issue where an Aurora Replica restarts when setting `innodb_adaptive_hash_index` to `OFF` on the Aurora Replica.
- Fixed an issue where an Aurora Replica restarts when running TRUNCATE TABLE queries on the Aurora Writer.
- Fixed an issue where the Aurora Writer freezes in certain circumstances when running INSERT statements. On a multi-node cluster, this can result in a failover.
- Fixed a memory leak associated with setting session variables.
- Fixed an issue where the Aurora Writer freezes in certain circumstances associated with purging undo for tables with generated columns.
- Fixed an issue where the Aurora Writer can sometimes restart when binary logging is enabled.

Integration of MySQL bug fixes

- Left join returns incorrect results on the outer side (Bug #22833364).

Aurora MySQL database engine updates 2018-03-13 (version 2.01.1) (Deprecated)

Version: 2.01.1

Aurora MySQL 2.01.1 is generally available. Aurora MySQL 2.x versions are compatible with MySQL 5.7 and Aurora MySQL 1.x versions are compatible with MySQL 5.6.

When creating a new Aurora MySQL DB cluster, you can choose compatibility with either MySQL 5.7 or MySQL 5.6. When restoring a MySQL 5.6-compatible snapshot, you can choose compatibility with either MySQL 5.7 or MySQL 5.6.

You can restore snapshots of Aurora MySQL 1.14*, 1.15*, 1.16*, and 1.17* into Aurora MySQL 2.01.1.

We don't allow in-place upgrade of Aurora MySQL 1.x clusters into Aurora MySQL 2.01.1 or restore to Aurora MySQL 2.01.1 from an Amazon S3 backup. We plan to remove these restrictions in a later Aurora MySQL 2.x release.

The performance schema is disabled for this release of Aurora MySQL 5.7. Upgrade to Aurora 2.03 for performance schema support.

Comparison with Aurora MySQL 5.6

The following Amazon Aurora MySQL features are supported in Aurora MySQL 5.6, but these features are currently not supported in Aurora MySQL 5.7.

- Asynchronous key prefetch (AKP). For more information, see Optimizing Aurora indexed join queries with asynchronous key prefetch in the *Amazon Aurora User Guide*.
- Hash joins. For more information, see Optimizing large Aurora MySQL join queries with hash joins in the *Amazon Aurora User Guide*.
- Native functions for synchronously invoking AWS Lambda functions. For more information, see Invoking a Lambda function with an Aurora MySQL native function in the *Amazon Aurora User Guide*.

- Scan batching. For more information, see Aurora MySQL database engine updates 2017-12-11 (version 1.16) (Deprecated).
- Migrating data from MySQL using an Amazon S3 bucket. For more information, see Migrating data from MySQL by using an Amazon S3 bucket in the *Amazon Aurora User Guide*.

Currently, Aurora MySQL 2.01.1 does not support features added in Aurora MySQL version 1.16 and later. For information about Aurora MySQL version 1.16, see Aurora MySQL database engine updates 2017-12-11 (version 1.16) (Deprecated).

MySQL 5.7 compatibility

Aurora MySQL 2.01.1 is wire-compatible with MySQL 5.7 and includes features such as JSON support, spatial indexes, and generated columns. Aurora MySQL uses a native implementation of spatial indexing using z-order curves to deliver >20x better write performance and >10x better read performance than MySQL 5.7 for spatial datasets.

Aurora MySQL 2.01.1 does not currently support the following MySQL 5.7 features:

- Global transaction identifiers (GTIDs). Aurora MySQL supports GTIDs in version 2.04 and higher.
- Group replication plugin
- Increased page size
- InnoDB buffer pool loading at startup
- InnoDB full-text parser plugin
- Multisource replication
- Online buffer pool resizing
- Password validation plugin
- Query rewrite plugins
- Replication filtering
- The `CREATE TABLESPACE` SQL statement
- X Protocol

CLI differences between Aurora MySQL 2.x and Aurora MySQL 1.x

- The engine name for Aurora MySQL 2.x is `aurora-mysql`; the engine name for Aurora MySQL 1.x continues to be `aurora`.

- The default parameter group for Aurora MySQL 2.x is `default.aurora-mysql5.7`; the default parameter group for Aurora MySQL 1.x continues to be `default.aurora5.6`.
- The DB cluster parameter group family name for Aurora MySQL 2.x is `aurora-mysql5.7`; the DB cluster parameter group family name for Aurora MySQL 1.x continues to be `aurora5.6`.

Refer to the Aurora documentation for the full set of CLI commands and differences between Aurora MySQL 2.x and Aurora MySQL 1.x.

Improvements

- Fixed an issue with snapshot restore where Aurora-specific database privileges were created incorrectly when a MySQL 5.6-compatible snapshot was restored with MySQL 5.7 compatibility.
- Added support for 1.17 snapshot restores.

Aurora MySQL database engine updates 2018-02-06 (version 2.01) (Deprecated)

Version: 2.01

Aurora MySQL 2.01 is generally available. Going forward, Aurora MySQL 2.x versions will be compatible with MySQL 5.7 and Aurora MySQL 1.x versions will be compatible with MySQL 5.6.

When creating a new Aurora MySQL DB cluster, including those restored from snapshots, you can choose compatibility with either MySQL 5.7 or MySQL 5.6.

You can restore snapshots of Aurora MySQL 1.14*, 1.15*, and 1.16* into Aurora MySQL 2.01.

We don't allow in-place upgrade of Aurora MySQL 1.x clusters into Aurora MySQL 2.01 or restore to Aurora MySQL 2.01 from an Amazon S3 backup. We plan to remove these restrictions in a later Aurora MySQL 2.x release.

The performance schema is disabled for this release of Aurora MySQL 5.7. Upgrade to Aurora 2.03 for performance schema support.

Comparison with Aurora MySQL 5.6

The following Amazon Aurora MySQL features are supported in Aurora MySQL 5.6, but these features are currently not supported in Aurora MySQL 5.7.

- Asynchronous key prefetch (AKP). For more information, see Optimizing Aurora indexed join queries with asynchronous key prefetch in the *Amazon Aurora User Guide*.
- Hash joins. For more information, see Optimizing large Aurora MySQL join queries with hash joins in the *Amazon Aurora User Guide*.
- Native functions for synchronously invoking AWS Lambda functions. For more information, see Invoking a Lambda function with an Aurora MySQL native function in the *Amazon Aurora User Guide*.
- Scan batching. For more information, see Aurora MySQL database engine updates 2017-12-11 (version 1.16) (Deprecated).
- Migrating data from MySQL using an Amazon S3 bucket. For more information, see Migrating data from MySQL by using an Amazon S3 bucket in the *Amazon Aurora User Guide*.

Currently, Aurora MySQL 2.01 does not support features added in Aurora MySQL version 1.16 and later. For information about Aurora MySQL version 1.16, see Aurora MySQL database engine updates 2017-12-11 (version 1.16) (Deprecated).

MySQL 5.7 compatibility

Aurora MySQL 2.01 is wire-compatible with MySQL 5.7 and includes features such as JSON support, spatial indexes, and generated columns. Aurora MySQL uses a native implementation of spatial indexing using z-order curves to deliver >20x better write performance and >10x better read performance than MySQL 5.7 for spatial datasets.

Aurora MySQL 2.01 does not currently support the following MySQL 5.7 features:

- Global transaction identifiers (GTIDs). Aurora MySQL supports GTIDs in version 2.04 and higher.
- Group replication plugin
- Increased page size
- InnoDB buffer pool loading at startup
- InnoDB full-text parser plugin
- Multisource replication
- Online buffer pool resizing
- Password validation plugin
- Query rewrite plugins
- Replication filtering

- The CREATE TABLESPACE SQL statement
- X Protocol

CLI differences between Aurora MySQL 2.x and Aurora MySQL 1.x

- The engine name for Aurora MySQL 2.x is `aurora-mysql`; the engine name for Aurora MySQL 1.x continues to be `aurora`.
- The default parameter group for Aurora MySQL 2.x is `default.aurora-mysql5.7`; the default parameter group for Aurora MySQL 1.x continues to be `default.aurora5.6`.
- The DB cluster parameter group family name for Aurora MySQL 2.x is `aurora-mysql5.7`; the DB cluster parameter group family name for Aurora MySQL 1.x continues to be `aurora5.6`.

Refer to the Aurora documentation for the full set of CLI commands and differences between Aurora MySQL 2.x and Aurora MySQL 1.x.

Database engine updates for Amazon Aurora MySQL version 1 (Deprecated)

The following are Amazon Aurora version 1 database engine updates:

- Aurora MySQL database engine updates 2021-09-30 (version 1.23.4) (Deprecated)
- Aurora MySQL database engine updates 2021-06-28 (version 1.23.3) (Deprecated)
- Aurora MySQL database engine updates 2021-03-18 (version 1.23.2) (Deprecated)
- Aurora MySQL database engine updates 2020-11-24 (version 1.23.1) (Deprecated)
- Aurora MySQL database engine updates 2020-09-02 (version 1.23.0) (Deprecated)
- Aurora MySQL database engine updates 2021-06-03 (version 1.22.5) (Deprecated)
- Aurora MySQL database engine updates 2021-03-04 (version 1.22.4) (Deprecated)
- Aurora MySQL database engine updates 2020-11-09 (version 1.22.3) (Deprecated)
- Aurora MySQL database engine updates 2020-03-05 (version 1.22.2) (Deprecated)
- Aurora MySQL database engine updates 2019-12-23 (version 1.22.1) (Deprecated)
- Aurora MySQL database engine updates 2019-11-25 (version 1.22.0) (Deprecated)
- Aurora MySQL database engine updates 2019-11-25 (version 1.21.0) (Deprecated)
- Aurora MySQL database engine updates 2020-03-05 (version 1.20.1) (Deprecated)
- Aurora MySQL database engine updates 2019-11-11 (version 1.20.0) (Deprecated)
- Aurora MySQL database engine updates 2020-03-05 (version 1.19.6) (Deprecated)
- Aurora MySQL database engine updates 2019-09-19 (version 1.19.5) (Deprecated)
- Aurora MySQL database engine updates 2019-06-05 (version 1.19.2) (Deprecated)
- Aurora MySQL database engine updates 2019-05-09 (version 1.19.1) (Deprecated)
- Aurora MySQL database engine updates 2019-02-07 (version 1.19.0) (Deprecated)
- Aurora MySQL database engine updates 2018-09-20 (version 1.18.0) (Deprecated)
- Aurora MySQL database engine updates 2020-03-05 (version 1.17.9) (Deprecated)
- Aurora MySQL database engine updates 2019-01-17 (version 1.17.8) (Deprecated)
- Aurora MySQL database engine updates 2018-10-08 (version 1.17.7) (Deprecated)
- Aurora MySQL database engine updates 2018-09-06 (version 1.17.6) (Deprecated)
- Aurora MySQL database engine updates 2018-08-14 (version 1.17.5) (Deprecated)

Aurora MySQL database engine updates 2021-09-30 (version 1.23.4) (Deprecated)

Version: 1.23.4

Aurora MySQL 1.23.4 is generally available. Aurora MySQL 2.* versions are compatible with MySQL 5.7 and Aurora MySQL 1.* versions are compatible with MySQL 5.6.

This engine version is scheduled to be deprecated on February 28, 2023. For more information, see Preparing for Amazon Aurora MySQL-Compatible Edition version 1 end of life.

Currently supported Aurora MySQL releases are 1.19.5, 1.19.6, 1.22.*, 1.23.*, 2.04.*, 2.07.*, 2.08.*, 2.09.*, 2.10.*, 3.01.* and 3.02.*.

To create a cluster with an older version of Aurora MySQL, specify the engine version through the RDS Console, the AWS CLI, or the Amazon RDS API.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Improvements

General improvements:

- Fixed an issue that can cause high CPU consumption on the reader instances due to excessive logging of informational messages in internal diagnostic log files.

High-priority fixes:

- CVE-2021-2307
- CVE-2021-2226
- CVE-2021-2160
- CVE-2021-2154
- CVE-2021-2060
- CVE-2021-2032
- CVE-2021-2001

Aurora MySQL database engine updates 2021-06-28 (version 1.23.3) (Deprecated)

Version: 1.23.3

Aurora MySQL 1.23.3 is generally available. Aurora MySQL 1.* versions are compatible with MySQL 5.6 and Aurora MySQL 2.* versions are compatible with MySQL 5.7.

This engine version is scheduled to be deprecated on February 28, 2023. For more information, see Preparing for Amazon Aurora MySQL-Compatible Edition version 1 end of life.

Currently supported Aurora MySQL releases are 1.19.5, 1.19.6, 1.22.*, 1.23.*, 2.04.*, 2.07.*, 2.08.*, 2.09.*, 2.10.*, 3.01.* and 3.02.*.

To create a cluster with an older version of Aurora MySQL, specify the engine version through the RDS Console, the AWS CLI, or the Amazon RDS API.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Improvements

General stability and availability enhancements.

Security fixes:

- CVE-2021-23841
- CVE-2021-3449
- CVE-2020-28196

Aurora MySQL database engine updates 2021-03-18 (version 1.23.2) (Deprecated)

Version: 1.23.2

Aurora MySQL 1.23.2 is generally available. Aurora MySQL 1.* versions are compatible with MySQL 5.6 and Aurora MySQL 2.* versions are compatible with MySQL 5.7.

This engine version is scheduled to be deprecated on February 28, 2023. For more information, see Preparing for Amazon Aurora MySQL-Compatible Edition version 1 end of life.

Currently supported Aurora MySQL releases are 1.19.5, 1.19.6, 1.22.*, 1.23.*, 2.04.*, 2.07.*, 2.08.*, 2.09.*, 2.10.*, 3.01.* and 3.02.*.

To create a cluster with an older version of Aurora MySQL, specify the engine version through the RDS Console, the AWS CLI, or the Amazon RDS API.

> **Note**
>
> This version is currently not available in the following regions: AWS GovCloud (US-East) [us-gov-east-1], AWS GovCloud (US-West) [us-gov-west-1]. There will be a separate announcement once it is made available.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Improvements

High priority fixes:

- CVE-2020-14867
- CVE-2020-14812
- CVE-2020-14769
- CVE-2020-14765
- CVE-2020-14793
- CVE-2020-14672
- CVE-2020-1971
- CVE-2018-3143

Availability improvements:

- Fixed an issue in the dynamic cluster storage resizing feature that could cause reader DB instances to restart.
- Fixed a failover issue due to a race condition in `RESET QUERY CACHE` statement.
- Fixed a crash in a nested stored procedure call with query cache.

- Fixed an issue to prevent repeated restart of `mysqld` when recovering from an incomplete truncation of partitioned or sub-partitioned tables.
- Fixed an issue that could cause migration from on-prem or RDS for MySQL to Aurora MySQL to not succeed.
- Fixed a rare race condition where the database can restart during the scaling of the storage volume.
- Fixed an issue in the lock manager where a race condition can cause a lock to be shared by two transactions, causing the database to restart.
- Fixed an issue related to transaction lock memory management with long-running write transactions resulting in a database restart.
- Fixed a race condition in the lock manager that resulted in a database restart or failover during transaction rollback.
- Fixed an issue during upgrade from 5.6 to 5.7 when the table had Fast Online DDL enabled in lab mode in 5.6.
- Fixed multiple issues where the engine might restart during zero-downtime patching while checking for a quiesced point in database activity for patching.
- Fixed multiple issues related to repeated restarts due to interrupted DDL operations, such as `DROP TRIGGER`, `ALTER TABLE`, and specifically `ALTER TABLE` that modifies the type of partitioning or number of partitions in a table.
- Updated the default value of `table_open_cache` on 16XL and 24XL instances to avoid repeated restarts and high CPU utilization on large instances classes (R4/R5-16XL, R5-12XL, R5-24XL). This impacted 1.21.x and 1.22.x releases.
- Fixed an issue that caused a binlog replica to stop with an `HA_ERR_KEY_NOT_FOUND` error.

Integration of MySQL community edition bug fixes

- *Replication*: While a `SHOW BINLOG EVENTS` statement was executing, any parallel transaction was blocked. The fix ensures that the `SHOW BINLOG EVENTS` process now only acquires a lock for the duration of calculating the file's end position, therefore parallel transactions are not blocked for long durations. (Bug #76618, Bug #20928790)

Aurora MySQL database engine updates 2020-11-24 (version 1.23.1) (Deprecated)

Version: 1.23.1

Aurora MySQL 1.23.1 is generally available. Aurora MySQL 1.* versions are compatible with MySQL 5.6 and Aurora MySQL 2.* versions are compatible with MySQL 5.7.

This engine version is scheduled to be deprecated on February 28, 2023. For more information, see Preparing for Amazon Aurora MySQL-Compatible Edition version 1 end of life.

Currently supported Aurora MySQL releases are 1.19.5, 1.19.6, 1.22.*, 1.23.*, 2.04.*, 2.07.*, 2.08.*, 2.09.*, 2.10.*, 3.01.* and 3.02.*.

To create a cluster with an older version of Aurora MySQL, specify the engine version through the RDS Console, the AWS CLI, or the Amazon RDS API.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Improvements

Security fixes:

Fixes and other enhancements to fine-tune handling in a managed environment. Additional CVE fixes below:

- CVE-2020-14559
- CVE-2020-14539

Incompatible changes:

This version introduces a permission change that affects the behavior of the `mysqldump` command. Users must have the `PROCESS` privilege to access the `INFORMATION_SCHEMA.FILES` table. To run the `mysqldump` command without any changes, grant the `PROCESS` privilege to the database user that the `mysqldump` command connects to. You can also run the `mysqldump` command with the `--no-tablespaces` option. With that option, the `mysqldump` output doesn't include any `CREATE LOGFILE GROUP` or `CREATE TABLESPACE` statements. In that case, the `mysqldump` command doesn't access the `INFORMATION_SCHEMA.FILES` table, and you don't need to grant the `PROCESS` permission.

Availability improvements:

- Fixed an issue that causes an Aurora reader instance in a global database secondary cluster running 1.23.0 to restart repeatedly.
- Fixed an issue where a global database secondary Region's replicas might restart when upgraded to release 1.23.0 while the primary Region writer was on an older release version.
- Fixed a memory leak in dynamic resizing feature, introduced in Aurora MySQL 1.23.0.
- Fixed an issue that might cause server restart during execution of a query using the parallel query feature.
- Fixed an issue that might cause a client session to hang when the database engine encounters an error while reading from or writing to the network.

Aurora MySQL database engine updates 2020-09-02 (version 1.23.0) (Deprecated)

Version: 1.23.0

Aurora MySQL 1.23.0 is generally available. Aurora MySQL 1.* versions are compatible with MySQL 5.6 and Aurora MySQL 2.* versions are compatible with MySQL 5.7.

This engine version is scheduled to be deprecated on February 28, 2023. For more information, see Preparing for Amazon Aurora MySQL-Compatible Edition version 1 end of life.

Currently supported Aurora MySQL releases are 1.19.5, 1.19.6, 1.22.*, 1.23.*, 2.04.*, 2.07.*, 2.08.*, 2.09.*, 2.10.*, 3.01.* and 3.02.*.

You can restore the snapshot of an Aurora MySQL 1.* database into Aurora MySQL 1.23.0.

> **Important**
>
> The improvements to Aurora storage in this version limit the available upgrade paths from Aurora MySQL 1.23 to Aurora MySQL 2.*. When you upgrade an Aurora MySQL 1.23 cluster to 2.*, you must upgrade to Aurora MySQL 2.09.0 or later.

To create a cluster with an older version of Aurora MySQL, please specify the engine version through the RDS Console, the AWS CLI, or the Amazon RDS API.

> **Note**
>
> This version is currently not available in the following regions: AWS GovCloud (US-East) [us-gov-east-1], AWS GovCloud (US-West) [us-gov-west-1]. There will be a separate announcement once it is made available.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Improvements

New features:

- You can now turn parallel query on or off for an existing cluster by changing the value of the DB cluster parameter `aurora_parallel_query`. You don't need to use the `parallelquery` setting for the `--engine-mode` parameter when creating the cluster.

  Parallel query is now expanded to be available in all regions where Aurora MySQL is available.

  There are a number of other functionality enhancements and changes to the procedures for upgrading and enabling parallel query in an Aurora cluster. For more information, see Working with parallel query for Amazon Aurora MySQL in the *Amazon Aurora User Guide*.
- With this release, you can create Amazon Aurora MySQL database instances with up to 128 tebibytes (TiB) of storage. The new storage limit is an increase from the prior 64 TiB. The 128 TiB storage size supports larger databases. This capability is not supported on small instances sizes (db.t2 or db.t3). A single tablespace cannot grow beyond 64 TiB due to InnoDB limitations with 16 KB page size.

  Aurora alerts you when the cluster volume size is near 128 TiB, so that you can take action prior to hitting the size limit. The alerts appear in the mysql log and RDS Events in the AWS Management Console.
- Improved binary log (binlog) processing to reduce crash recovery time and commit time latency when very large transactions are involved.
- Aurora dynamically resizes your cluster storage space. With dynamic resizing, the storage space for your Aurora DB cluster automatically decreases when you remove data from the DB cluster. For more information, see Storage scaling in the *Amazon Aurora User Guide*.

> **Note**
>
> The dynamic resizing feature is being deployed in phases to the AWS Regions where Aurora is available. Depending on the Region where your cluster is, this feature might not be available yet. For more information, see the What's New announcement.

High priority fixes:

- CVE-2019-2911
- CVE-2019-2537
- CVE-2018-2787
- CVE-2018-2784
- CVE-2018-2645
- CVE-2018-2640

Availability improvements:

- Fixed an issue in the lock manager where a race condition can cause a lock to be shared by two transactions, causing the database to restart.
- Fixed an issue related to transaction lock memory management with long-running write transactions resulting in a database restart.
- Fixed a race condition in the lock manager that resulted in a database restart or failover during transaction rollback.
- Fixed an issue during upgrade from 5.6 to 5.7 when the `innodb_file_format` changed on a table that had Fast DDL enabled.
- Fixed multiple issues where the engine might restart during zero-downtime patching while checking for a quiesced point in database activity for patching.
- Fixed an issue related to DDL recovery that impacts restart of the DB instance while recovering an interrupted `DROP TRIGGER` operation.
- Fixed a bug that might cause unavailability of the database if a crash occurs during the execution of certain partitioning operations. Specifically, an interrupted `ALTER TABLE` operation that modifies the type of partitioning or the number of partitions in a table.

- Fix default value of `table_open_cache` on 16XL and 24XL instances which could cause repeated failovers and high CPU utilization on large instances classes (R4/R5-16XL, R5-12XL, R5-24XL). This impacted 1.21.x and 1.22.x.

Global databases:

- Populate missing data in the MySQL `INFORMATION_SCHEMA.REPLICA_HOST_STATUS` view on primary and secondary AWS Regions in an Aurora global database.
- Fixed unexpected query failures that could occur in a Global DB secondary Region due to garbage collection of UNDO records in the primary Region, after temporary network connectivity issues between the primary and secondary Regions.

Parallel query:

- Fixed an issue where parallel query might cause a long-running query to return an empty result.
- Fixed an issue where a query on a small table on the Aurora read replica might take more than one second.
- Fixed an issue that might cause a restart when a parallel query and a DML statement are executing concurrently under a heavy workload.

General improvements:

- Fixed an issue where queries using spatial index might return partial results if spatial index was created on tables with already existing large spatial values.
- Increased maximum allowable length for audit system variables `server_audit_incl_users` and `server_audit_excl_users` from 1024 bytes to 2000 bytes.
- Fixed an issue where a binlog replica connected to an Aurora MySQL binlog primary might show incomplete data when the Aurora MySQL binlog primary loads data from S3 under `statement binlog_format`.
- Comply with community behavior to map `mixed` binlog_format to `row` instead of `statement` for loading data.
- Fixed an issue causing binlog replication to stop working when the user closes the connection and the session is using temporary tables.
- Improved response time of a query involving MyISAM temporary tables.
- Fix permission issue when binlog worker runs a native lambda function.

- Fixed an issue on Aurora read replicas when trying to query or rotate the slow log or general log.
- Fixed an issue that broke logical replication when the `binlog_checksum` parameter is set to different values on the master and the replica.
- Fixed an issue where the read replica might transiently see partial results of a recently committed transaction on the writer.
- Include transaction info of the rolled-back transaction in `show engine innodb status` when a deadlock is resolved.

Integration of MySQL community edition bug fixes

- Binlog events with `ALTER TABLE ADD COLUMN ALGORITHM=QUICK` will be rewritten as `ALGORITHM=DEFAULT` to be compatible with the community edition.
- BUG #22350047: IF CLIENT KILLED AFTER ROLLBACK TO SAVEPOINT PREVIOUS STMTS COMMITTED
- Bug #29915479: RUNNING COM_REGISTER_SLAVE WITHOUT COM_BINLOG_DUMP CAN RESULTS IN SERVER EXIT
- Bug #30441969: BUG #29723340: MYSQL SERVER CRASH AFTER SQL QUERY WITH DATA ?AST
- Bug #30628268: OUT OF MEMORY CRASH
- Bug #27081349: UNEXPECTED BEHAVIOUR WHEN DELETE WITH SPATIAL FUNCTION
- Bug #27230859: UNEXPECTED BEHAVIOUR WHILE HANDLING INVALID POLYGON"
- Bug #27081349: UNEXPECTED BEHAVIOUR WHEN DELETE WITH SPATIAL"
- Bug #26935001: ALTER TABLE AUTO_INCREMENT TRIES TO READ INDEX FROM DISCARDED TABLESPACE
- Bug #29770705: SERVER CRASHED WHILE EXECUTING SELECT WITH SPECIFIC WHERE CLAUSE
- Bug #27659490: SELECT USING DYNAMIC RANGE AND INDEX MERGE USE TOO MUCH MEMORY(OOM)
- Bug #24786290: REPLICATION BREAKS AFTER BUG #74145 HAPPENS IN MASTER
- Bug #27703912: EXCESSIVE MEMORY USAGE WITH MANY PREPARE
- Bug #20527363: TRUNCATE TEMPORARY TABLE CRASH: !DICT_TF2_FLAG_IS_SET(TABLE, DICT_TF2_TEMPORARY)
- Bug#23103937 PS_TRUNCATE_ALL_TABLES() DOES NOT WORK IN SUPER_READ_ONLY MODE

- Bug #25053286: USE VIEW WITH CONDITION IN PROCEDURE CAUSES INCORRECT BEHAVIOR (fixed in 5.6.36)
- Bug #25586773: INCORRECT BEHAVIOR FOR CREATE TABLE SELECT IN A LOOP IN SP (fixed in 5.6.39)
- Bug #27407480: AUTOMATIC_SP_PRIVILEGES REQUIRES NEED THE INSERT PRIVILEGES FOR MYSQL.USER TABLE
- Bug #26997096: `relay_log_space` value is not updated in a synchronized manner so that its value is sometimes much higher than the actual disk space used by relay logs.
- Bug#15831300 SLAVE_TYPE_CONVERSIONS=ALL_NON_LOSSY NOT WORKING AS EXPECTED
- SSL Bug backport Bug #17087862, Bug #20551271
- Bug #16894092: PERFORMANCE REGRESSION IN 5.6.6+ FOR INSERT INTO ... SELECT ... FROM (fixed in 5.6.15).
- Port a bug fix related to `SLAVE_TYPE_CONVERSIONS`.

Aurora MySQL database engine updates 2021-06-03 (version 1.22.5) (Deprecated)

Version: 1.22.5

Aurora MySQL 1.22.5 is generally available. Aurora MySQL 1.* versions are compatible with MySQL 5.6 and Aurora MySQL 2.* versions are compatible with MySQL 5.7.

This engine version is scheduled to be deprecated on February 28, 2023. For more information, see Preparing for Amazon Aurora MySQL-Compatible Edition version 1 end of life.

Currently supported Aurora MySQL releases are 1.19.5, 1.19.6, 1.22.*, 1.23.*, 2.04.*, 2.07.*, 2.08.*, 2.09.*, 2.10.*, 3.01.* and 3.02.*.

To create a cluster with an older version of Aurora MySQL, specify the engine version through the RDS Console, the AWS CLI, or the Amazon RDS API.

> **Note**
>
> This version is designated as a long-term support (LTS) release. For more information, see Aurora MySQL long-term support (LTS) releases in the *Amazon Aurora User Guide*.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Improvements

Availability improvements:

- Resolved an issue that could cause the database to stall, and subsequently restart or fail over due to a concurrency conflict between internal cleanup threads.
- Resolved an issue that could cause the cluster to become unavailable if the database restarted while holding XA transactions in prepared state, and then restarted again before those transactions were committed or rolled back. Prior to this fix, you can address the issue by restoring the cluster to a point in time before the first restart.
- Resolved an issue that could cause the InnoDB purge to become blocked if the database restarts while processing a DDL statement. As a result, the InnoDB history list length would grow and the cluster storage volume would keep growing until it fills up, making the database unavailable. Prior to this fix, you can mitigate the issue by restarting the database again to unblock purge.

Aurora MySQL database engine updates 2021-03-04 (version 1.22.4) (Deprecated)

Version: 1.22.4

Aurora MySQL 1.22.4 is generally available. Aurora MySQL 1.* versions are compatible with MySQL 5.6 and Aurora MySQL 2.* versions are compatible with MySQL 5.7.

This engine version is scheduled to be deprecated on February 28, 2023. For more information, see Preparing for Amazon Aurora MySQL-Compatible Edition version 1 end of life.

Currently supported Aurora MySQL releases are 1.19.5, 1.19.6, 1.22.*, 1.23.*, 2.04.*, 2.07.*, 2.08.*, 2.09.*, 2.10.*, 3.01.* and 3.02.*.

To create a cluster with an older version of Aurora MySQL, specify the engine version through the RDS Console, the AWS CLI, or the Amazon RDS API.

> **Note**
>
> This version is designated as a long-term support (LTS) release. For more information, see Aurora MySQL long-term support (LTS) releases in the *Amazon Aurora User Guide*.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Improvements

Security fixes:

Fixes and other enhancements to fine-tune handling in a managed environment. Additional CVE fixes below:

- CVE-2020-14867
- CVE-2020-14812
- CVE-2020-14793
- CVE-2020-14769
- CVE-2020-14765
- CVE-2020-14672
- CVE-2020-1971

Availability improvements:

- Fixed an issue that could trigger a database restart or failover during a `kill session` command. If you encounter this issue, contact AWS support to enable this fix on your instance.
- Improved binary logging to reduce crash recovery time and commit latency when large transactions are involved.
- Fixed an issue that caused a binlog replica to stop with an `HA_ERR_KEY_NOT_FOUND` error.

Aurora MySQL database engine updates 2020-11-09 (version 1.22.3) (Deprecated)

Version: 1.22.3

Aurora MySQL 1.22.3 is generally available. Aurora MySQL 1.* versions are compatible with MySQL 5.6 and Aurora MySQL 2.* versions are compatible with MySQL 5.7.

This engine version is scheduled to be deprecated on February 28, 2023. For more information, see Preparing for Amazon Aurora MySQL-Compatible Edition version 1 end of life.

Currently supported Aurora MySQL releases are 1.19.5, 1.19.6, 1.22.*, 1.23.*, 2.04.*, 2.07.*, 2.08.*, 2.09.*, 2.10.*, 3.01.* and 3.02.*.

To create a cluster with an older version of Aurora MySQL, specify the engine version through the RDS Console, the AWS CLI, or the Amazon RDS API.

> **Note**
>
> This version is designated as a long-term support (LTS) release. For more information, see Aurora MySQL long-term support (LTS) releases in the *Amazon Aurora User Guide*.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Improvements

Security fixes:

Fixes and other enhancements to fine-tune handling in a managed environment. Additional CVE fixes below:

- CVE-2020-14559
- CVE-2020-14539
- CVE-2020-2579
- CVE-2020-2812
- CVE-2020-2780
- CVE-2020-2763

Incompatible changes:

This version introduces a permission change that affects the behavior of the `mysqldump` command. Users must have the `PROCESS` privilege to access the `INFORMATION_SCHEMA.FILES` table. To run the `mysqldump` command without any changes, grant the `PROCESS` privilege to the database user that the `mysqldump` command connects to. You can also run the `mysqldump` command with the `--no-tablespaces` option. With that option, the `mysqldump` output doesn't include any `CREATE LOGFILE GROUP` or `CREATE TABLESPACE` statements. In that case, the `mysqldump` command doesn't access the `INFORMATION_SCHEMA.FILES` table, and you don't need to grant the `PROCESS` permission.

Availability improvements:

- Fixed issues that might cause server restarts during recovery of a DDL statement that was not committed.
- Fixed race conditions in the lock manager that can cause a server restart.
- Fixed an issue that might cause the monitoring agent to restart the server during recovery of a large transaction

General improvements:

- Changed the behavior to map `MIXED` `binlog_format` to `ROW` instead of `STATEMENT` when executing `LOAD DATA FROM INFILE | S3`.
- Fixed an issue where a binlog replica connected to an Aurora MySQL binlog primary might show incomplete data when the primary executed `LOAD DATA FROM S3` and `binlog_format` is set to `STATEMENT`.

Integration of MySQL community edition bug fixes

- Bug #26654685: A corrupt index ID encountered during a foreign key check raised an assertion
- Bug #15831300: By default, when promoting integers from a smaller type on the master to a larger type on the slave (for example, from a SMALLINT column on the master to a BIGINT column on the slave), the promoted values are treated as though they are signed. Now in such cases it is possible to modify or override this behavior using one or both of `ALL_SIGNED`, `ALL_UNSIGNED` in the set of values specified for the slave_type_conversions server system variable. For more information, see Row-based replication: attribute promotion and demotion, as well as the description of the variable.

- Bug #17449901: With `foreign_key_checks=0`, InnoDB permitted an index required by a foreign key constraint to be dropped, placing the table into an inconsistent and causing the foreign key check that occurs at table load to fail. InnoDB now prevents dropping an index required by a foreign key constraint, even with foreign_key_checks=0. The foreign key constraint must be removed before dropping the foreign key index.
- BUG #20768847: An ALTER TABLE ... DROP INDEX operation on a table with foreign key dependencies raised an assertion.

Aurora MySQL database engine updates 2020-03-05 (version 1.22.2) (Deprecated)

Version: 1.22.2

Aurora MySQL 1.22.2 is generally available. Aurora MySQL 1.* versions are compatible with MySQL 5.6 and Aurora MySQL 2.* versions are compatible with MySQL 5.7.

This engine version is scheduled to be deprecated on February 28, 2023. For more information, see Preparing for Amazon Aurora MySQL-Compatible Edition version 1 end of life.

Currently supported Aurora MySQL releases are 1.19.5, 1.19.6, 1.22.*, 1.23.*, 2.04.*, 2.07.*, 2.08.*, 2.09.*, 2.10.*, 3.01.* and 3.02.*.

To create a cluster with an older version of Aurora MySQL, please specify the engine version through the RDS Console, the AWS CLI, or the Amazon RDS API.

> **Note**
>
> This version is currently not available in the following regions: AWS GovCloud (US-East) [us-gov-east-1], AWS GovCloud (US-West) [us-gov-west-1]. There will be a separate announcement once it is made available.
> This version is designated as a long-term support (LTS) release. For more information, see Aurora MySQL long-term support (LTS) releases in the *Amazon Aurora User Guide*.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Improvements

High priority fixes:

- Fixed an issue of intermittent connection failures after certificate rotation.
- Fixed an issue that caused cloning to take longer on some database clusters with high write loads.
- Fixed an issue that broke logical replication when the `binlog_checksum` parameter is set to different values on the master and the replica.
- Fixed an issue where slow log and general log may not properly rotate on read replicas.
- Fixed an issue with ANSI Read Committed Isolation Level behavior.

Aurora MySQL database engine updates 2019-12-23 (version 1.22.1) (Deprecated)

Version: 1.22.1

Aurora MySQL 1.22.1 is generally available. Aurora MySQL 1.* versions are compatible with MySQL 5.6 and Aurora MySQL 2.* versions are compatible with MySQL 5.7.

This engine version is scheduled to be deprecated on February 28, 2023. For more information, see Preparing for Amazon Aurora MySQL-Compatible Edition version 1 end of life.

Currently supported Aurora MySQL releases are 1.19.5, 1.19.6, 1.22.*, 1.23.*, 2.04.*, 2.07.*, 2.08.*, 2.09.*, 2.10.*, 3.01.* and 3.02.*.

To create a cluster with an older version of Aurora MySQL, please specify the engine version through the AWS Management Console, the AWS CLI or the RDS API. You have the option to upgrade existing Aurora MySQL 1.* database clusters to Aurora MySQL 1.22.1.

> **Note**
>
> This version is currently not available in the following AWS Regions: AWS GovCloud (US-East) [us-gov-east-1], AWS GovCloud (US-West) [us-gov-west-1], China (Ningxia) [cn-northwest-1], Asia Pacific (Hong Kong) [ap-east-1], and Middle East (Bahrain) [me-south-1. There will be a separate announcement once it is made available.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

> **Note**
>
> The procedure to upgrade your DB cluster has changed. For more information, see Upgrading the minor version or patch level of an Aurora MySQL DB cluster in the *Amazon Aurora User Guide*.

Improvements

Critical fixes:

- Fixed issues that prevented engine recovery involving table locks and temporary tables.
- Improved the stability of binary log when temporary tables are used.

High priority fixes:

- Fixed a slow memory leak in Aurora specific database tracing and logging sub-system that lowers the freeable memory.

Aurora MySQL database engine updates 2019-11-25 (version 1.22.0) (Deprecated)

Version: 1.22.0

Aurora MySQL 1.22.0 is generally available. Aurora MySQL 1.* versions are compatible with MySQL 5.6 and Aurora MySQL 2.* versions are compatible with MySQL 5.7.

This engine version is scheduled to be deprecated on February 28, 2023. For more information, see Preparing for Amazon Aurora MySQL-Compatible Edition version 1 end of life.

Currently supported Aurora MySQL releases are 1.19.5, 1.19.6, 1.22.*, 1.23.*, 2.04.*, 2.07.*, 2.08.*, 2.09.*, 2.10.*, 3.01.* and 3.02.*.

To create a cluster with an older version of Aurora MySQL, please specify the engine version through the AWS Management Console, the AWS CLI or the RDS API. You have the option to upgrade existing Aurora MySQL 1.* database clusters to Aurora MySQL 1.22.0.

> **Note**
>
> This version is currently not available in the following AWS Regions: AWS GovCloud (US-East) [us-gov-east-1], AWS GovCloud (US-West) [us-gov-west-1], China (Ningxia) [cn-northwest-1], Asia Pacific (Hong Kong) [ap-east-1], Middle East (Bahrain) [me-south-1], and South America (São Paulo) [sa-east-1]. There will be a separate announcement once it is made available.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

> **Note**
>
> The procedure to upgrade your DB cluster has changed. For more information, see Upgrading the minor version or patch level of an Aurora MySQL DB cluster in the *Amazon Aurora User Guide*.

Improvements

New features:

- Aurora MySQL clusters now support the instance types r5.8xlarge, r5.16xlarge and r5.24xlarge.
- Binlog has new enhancements for improved commit time latency when very large transactions are involved.
- Aurora MySQL now has a mechanism to minimize the time window during which events of a large transaction are written to binlog on commit. This effectively prevents lengthy offline recovery incurred when database crashes occur during that time window. This feature also fixes the issue where a large transaction blocks small transactions on binlog commit. This feature is off by default and can be enabled by the service team if needed for your workload. When enabled, it will be triggered when a transaction size is > 500MB.

- Added support for the ANSI `READ COMMITTED` isolation level on the read replicas. This isolation level enables long-running queries on the read replica to execute without impacting the high throughput of writes on the writer node. For more information, see Aurora MySQL isolation levels.
- Global Databases now allow adding secondary read-only replica regions for database clusters deployed in these AWS Regions: regions: US East (N. Virginia) [us-east-1], US East (Ohio) [us-east-2], US West (N. California) [us-west-1], US West (Oregon) [us-west-2], Europe (Ireland) [eu-west-1], Europe (London) [eu-west-2], Europe (Paris) [eu-west-3], Asia Pacific (Tokyo) [ap-northeast-1], Asia Pacific (Seoul) [ap-northeast-2], Asia Pacific (Singapore) [ap-southeast-1], Asia Pacific (Sydney) [ap-southeast-2], Canada (Central) [ca-central-1], Europe (Frankfurt) [eu-central-1], and Asia Pacific (Mumbai) [ap-south-1].
- The hot row contention feature is now generally available and does not require the Aurora lab mode setting to be ON. This feature substantially improves throughput for workloads with many transactions contending for rows on the same page.
- This version has updated timezone files to support the latest Brazil timezone update for new clusters.

Critical fixes:

- CVE-2019-2922
- CVE-2019-2923
- CVE-2019-2924
- CVE-2019-2910

High priority fixes:

- CVE-2019-2805
- CVE-2019-2730
- CVE-2019-2740
- CVE-2018-3064
- CVE-2018-3058
- CVE-2017-3653
- CVE-2017-3464
- CVE-2017-3244

- CVE-2016-5612
- CVE-2016-5439
- CVE-2016-0606
- CVE-2015-4904
- CVE-2015-4879
- CVE-2015-4864
- CVE-2015-4830
- CVE-2015-4826
- CVE-2015-2620
- CVE-2015-0382
- CVE-2015-0381
- CVE-2014-6555
- CVE-2014-4258
- CVE-2014-4260
- CVE-2014-2444
- CVE-2014-2436
- CVE-2013-5881
- CVE-2014-0393
- CVE-2013-5908
- CVE-2013-5807
- CVE-2013-3806
- CVE-2013-3811
- CVE-2013-3804
- CVE-2013-3807
- CVE-2013-2378
- CVE-2013-2375
- CVE-2013-1523
- CVE-2013-2381
- CVE-2012-5615
- CVE-2014-6489

- Fixed an issue in the DDL recovery component that resulted in prolonged database downtime. Clusters that become unavailable after executing `TRUNCATE TABLE` query on a table with an `AUTO_INCREMENT` column should be updated.
- Fixed an issue in the DDL recovery component that resulted in prolonged database downtime. Clusters that become unavailable after executing `DROP TABLE` query on multiple tables in parallel should be updated.

General stability fixes:

- Fixed an issue that caused read replicas to restart during a long-running transaction. Customers who encounter replica restarts that coincide with an accelerated drop in freeable memory should consider upgrading to this version.
- Fixed an issue that incorrectly reported `ERROR 1836` when a nested query is executed against a temporary table on the read replica.
- Fixed a parallel query abort error on an Aurora reader instance while a heavy write workload is running on the Aurora writer instance.
- Fixed an issue that causes a database configured as a Binlog Master to restart while a heavy write workload is running.
- Fixed an issue of prolonged unavailability while restarting the engine. This addresses an issue in the buffer pool initialization. This issue occurs rarely but can potentially impact any supported release.
- Fixed an issue that generated inconsistent data in the `information_schema.replica_host_status` table.
- Fixed a race condition between the parallel query and the standard execution paths that caused the Reader nodes to restart intermittently.
- Improved stability of the database when the number of number of client connections exceeds the `max_connections` parameter value.
- Improved stability of the reader instances by blocking unsupported DDL and `LOAD FROM S3` queries.

Integration of MySQL community edition bug fixes

- Bug#16346241 - SERVER CRASH IN ITEM_PARAM::QUERY_VAL_STR
- Bug#17733850 - NAME_CONST() CRASH IN ITEM_NAME_CONST::ITEM_NAME_CONST()

- Bug #20989615 - INNODB AUTO_INCREMENT PRODUCES SAME VALUE TWICE
- Bug #20181776 - ACCESS CONTROL DOESN'T MATCH MOST SPECIFIC HOST WHEN IT CONTAINS WILDCARD
- Bug #27326796 - MYSQL CRASH WITH INNODB ASSERTION FAILURE IN FILE PARS0PARS.CC
- Bug #20590013 - IF YOU HAVE A FULLTEXT INDEX AND DROP IT YOU CAN NO LONGER PERFORM ONLINE DDL

Aurora MySQL database engine updates 2019-11-25 (version 1.21.0) (Deprecated)

Version: 1.21.0

Aurora MySQL 1.21.0 is generally available. Aurora MySQL 1.* versions are compatible with MySQL 5.6 and Aurora MySQL 2.* versions are compatible with MySQL 5.7.

Currently supported Aurora MySQL releases are 1.14.*, 1.15.*, 1.16.*, 1.17.*, 1.18.*, 1.19.*, 1.20.*, 1.21.*, 1.22.*, 2.01.*, 2.02.*, 2.03.*, 2.04.*, 2.05.*, 2.06.* and 2.07.*. To create a cluster with an older version of Aurora MySQL, please specify the engine version through the AWS Management Console, the AWS CLI or the RDS API. You have the option to upgrade existing Aurora MySQL 1.* database clusters to Aurora MySQL 1.21.0.

> **Note**
>
> This version is currently not available in the following AWS Regions: AWS GovCloud (US-East) [us-gov-east-1], AWS GovCloud (US-West) [us-gov-west-1], China (Ningxia) [cn-northwest-1], Asia Pacific (Hong Kong) [ap-east-1], Europe (Stockholm) [eu-north-1], and Middle East (Bahrain) [me-south-1]. There will be a separate announcement once it is made available.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

> **Note**
>
> The procedure to upgrade your DB cluster has changed. For more information, see Upgrading the minor version or patch level of an Aurora MySQL DB cluster in the *Amazon Aurora User Guide*.

Improvements

Critical fixes:

- CVE-2018-0734
- CVE-2019-2534
- CVE-2018-2612
- CVE-2017-3599
- CVE-2018-2562
- CVE-2017-3329
- CVE-2018-2696
- CVE-2015-4737

High priority fixes:

- Customers with database size close to 64 tebibytes (TiB) are strongly advised to upgrade to this version to avoid downtime due to stability bugs affecting volumes close to the Aurora storage limit.

General stability fixes:

- Fixed a parallel query abort error on Aurora reader instances while a heavy write workload is running on the Aurora writer instance.
- Fixed an issue on Aurora reader instances that reduced free memory during long-running transactions while there is a heavy transaction commit traffic on the writer instance.
- The value of the parameter `aurora_disable_hash_join` is now persisted after database restart or host replacement.

- Fixed an issue related to the Full Text Search cache that caused the Aurora instance to run out of memory. Customers using Full Text Search should upgrade.
- Improved stability of the database when the hash join feature is enabled and the instance is low on memory. Customers using hash join should upgrade.
- Fixed an issue in the query cache where the "Too many connections" error could cause a reboot.
- Fixed the free memory calculation on T2 instances to include swap memory space to prevent unnecessary reboots.

Integration of MySQL community edition bug fixes

- Bug #19929406: HANDLE_FATAL_SIGNAL (SIG=11) IN __MEMMOVE_SSSE3_BACK FROM STRING::COPY
- Bug #17059925: For UNION statements, the rows-examined value was calculated incorrectly. This was manifested as too-large values for the `ROWS_EXAMINED` column of Performance Schema statement tables (such as events_statements_current).
- Bug #11827369: Some queries with `SELECT ... FROM DUAL` nested subqueries raised an assertion.
- Bug #16311231: Incorrect results were returned if a query contained a subquery in an `IN` clause that contained an XOR operation in the `WHERE` clause.

Aurora MySQL database engine updates 2020-03-05 (version 1.20.1) (Deprecated)

Version: 1.20.1

Aurora MySQL 1.20.1 is generally available. Aurora MySQL 1.* versions are compatible with MySQL 5.6 and Aurora MySQL 2.* versions are compatible with MySQL 5.7.

Currently supported Aurora MySQL releases are 1.14.*, 1.15.*, 1.16.*, 1.17.*, 1.18.*, 1.19.*, 1.20.*, 1.21.*, 1.22.*, 2.01.*, 2.02.*, 2.03.*, 2.04.*, 2.05.*, 2.06.* and 2.07.*. You can restore the snapshot of an Aurora MySQL 1.* database into Aurora MySQL 1.20.1.

To create a cluster with an older version of Aurora MySQL, please specify the engine version through the RDS Console, the AWS CLI, or the Amazon RDS API.

> **Note**
>
> This version is currently not available in the following regions: AWS GovCloud (US-East) [us-gov-east-1], AWS GovCloud (US-West) [us-gov-west-1]. There will be a separate announcement once it is made available.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Improvements

High priority fixes:

- Fixed an issue of intermittent connection failures after certificate rotation.
- Fixed an issue related to connection close concurrency that would result in a failover under heavy workload.

General stability fixes:

- Fixed a crash during execution of a complex query involving multi-table joins and aggregation that uses intermediate tables internally.

Aurora MySQL database engine updates 2019-11-11 (version 1.20.0) (Deprecated)

Version: 1.20.0

Aurora MySQL 1.20.0 is generally available. Aurora MySQL 1.* versions are compatible with MySQL 5.6 and Aurora MySQL 2.* versions are compatible with MySQL 5.7.

Currently supported Aurora MySQL releases are 1.14.*, 1.15.*, 1.16.*, 1.17.*, 1.18.*, 1.19.*, 1.20.*, 2.01.*, 2.02.*, 2.03.* and 2.04.*. To create a cluster with an older version of Aurora MySQL, please specify the engine version through the AWS Management Console, the AWS CLI or the RDS API. You have the option to upgrade existing Aurora MySQL 1.* database clusters, up to 1.19.5, to Aurora MySQL 1.20.0.

> **Note**
>
> This version is currently not available in the following AWS Regions: AWS GovCloud (US-East) [us-gov-east-1], AWS GovCloud (US-West) [us-gov-west-1], China (Ningxia) [cn-northwest-1], Asia Pacific (Hong Kong) [ap-east-1], Europe (Stockholm) [eu-north-1], and Middle East (Bahrain) [me-south-1]. There will be a separate announcement once it is made available.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

> **Note**
>
> The procedure to upgrade your DB cluster has changed. For more information, see Upgrading the minor version or patch level of an Aurora MySQL DB cluster in the *Amazon Aurora User Guide*.

Improvements

Critical fixes:

- CVE-2018-0734
- CVE-2019-2534
- CVE-2018-2612
- CVE-2017-3599
- CVE-2018-2562
- CVE-2017-3329
- CVE-2018-2696
- CVE-2015-4737

High priority fixes:

- Customers with database size close to 64 tebibytes (TiB) are strongly advised to upgrade to this version to avoid downtime due to stability bugs affecting volumes close to the Aurora storage limit.

General stability fixes:

- Fixed a parallel query abort error on Aurora reader instances while a heavy write workload is running on the Aurora writer instance.
- Fixed an issue on Aurora reader instances that reduced free memory during long-running transactions while there is a heavy transaction commit traffic on the writer instance.
- The value of the parameter `aurora_disable_hash_join` is now persisted after database restart or host replacement.
- Fixed an issue related to the Full Text Search cache that caused the Aurora instance to run out of memory. Customers using Full Text Search should upgrade.
- Improved stability of the database when the hash join feature is enabled and the instance is low on memory. Customers using hash join should upgrade.
- Fixed an issue in the query cache where the "Too many connections" error could cause a reboot.
- Fixed the free memory calculation on T2 instances to include swap memory space to prevent unnecessary reboots.

Integration of MySQL community edition bug fixes

- Bug #19929406: HANDLE_FATAL_SIGNAL (SIG=11) IN __MEMMOVE_SSSE3_BACK FROM STRING::COPY
- Bug #17059925: For UNION statements, the rows-examined value was calculated incorrectly. This was manifested as too-large values for the `ROWS_EXAMINED` column of Performance Schema statement tables (such as events_statements_current).
- Bug #11827369: Some queries with `SELECT ... FROM DUAL` nested subqueries raised an assertion.
- Bug #16311231: Incorrect results were returned if a query contained a subquery in an `IN` clause that contained an XOR operation in the `WHERE` clause.

Aurora MySQL database engine updates 2020-03-05 (version 1.19.6) (Deprecated)

Version: 1.19.6

Aurora MySQL 1.19.6 is generally available. Aurora MySQL 1.* versions are compatible with MySQL 5.6 and Aurora MySQL 2.* versions are compatible with MySQL 5.7.

This engine version is scheduled to be deprecated on February 28, 2023. For more information, see Preparing for Amazon Aurora MySQL-Compatible Edition version 1 end of life.

Currently supported Aurora MySQL releases are 1.19.5, 1.19.6, 1.22.*, 1.23.*, 2.04.*, 2.07.*, 2.08.*, 2.09.*, 2.10.*, 3.01.* and 3.02.*.

You can restore the snapshot of an Aurora MySQL 1.* database into Aurora MySQL 1.19.6.

To create a cluster with an older version of Aurora MySQL, please specify the engine version through the RDS Console, the AWS CLI, or the Amazon RDS API.

> **Note**
>
> This version is currently not available in the following regions: AWS GovCloud (US-East) [us-gov-east-1], AWS GovCloud (US-West) [us-gov-west-1]. There will be a separate announcement once it is made available.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Improvements

High priority fixes:

- Fixed an issue of intermittent connection failures after certificate rotation.

Aurora MySQL database engine updates 2019-09-19 (version 1.19.5) (Deprecated)

Version: 1.19.5

Aurora MySQL 1.19.5 is generally available. Aurora MySQL 1.* versions are compatible with MySQL 5.6 and Aurora MySQL 2.* versions are compatible with MySQL 5.7.

This engine version is scheduled to be deprecated on February 28, 2023. For more information, see Preparing for Amazon Aurora MySQL-Compatible Edition version 1 end of life.

Currently supported Aurora MySQL releases are 1.19.5, 1.19.6, 1.22.*, 1.23.*, 2.04.*, 2.07.*, 2.08.*, 2.09.*, 2.10.*, 3.01.* and 3.02.*.

You have the option to upgrade existing database clusters to Aurora MySQL 1.19.5. You can restore snapshots of Aurora MySQL 1.14.*, 1.15.*, 1.16.*, 1.17.*, 1.18.*, 1.19.1, and 1.19.2 into Aurora MySQL 1.19.5.

To use an older version of Aurora MySQL, you can create new database clusters by specifying the engine version through the AWS Management Console, the AWS CLI, or the RDS API.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Note

This version is currently not available in the following AWS Regions: Europe (London) [eu-west-2], AWS GovCloud (US-East) [us-gov-east-1], AWS GovCloud (US-West) [us-gov-west-1], China (Ningxia) [cn-northwest-1], and Asia Pacific (Hong Kong) [ap-east-1]. There will be a separate announcement once it is made available.

Note

The procedure to upgrade your DB cluster has changed. For more information, see Upgrading the minor version or patch level of an Aurora MySQL DB cluster in the *Amazon Aurora User Guide*.

Improvements

- Fixed an issue on Aurora reader instances that reduced free memory during long-running transactions while there is a heavy transaction commit traffic on the writer instance.
- Fixed a parallel query abort error on Aurora reader instances while a heavy write workload is running on the Aurora writer instance.
- The value of the parameter `aurora_disable_hash_join` is now persisted after database restart or host replacement.
- Fixed an issue related to the Full Text Search cache that caused the Aurora instance to run out of memory.
- Improved stability of the database when the volume size is close to the 64 tebibyte (TiB) volume limit by reserving 160 GB of space for the recovery workflow to complete without a failover.
- Improved stability of the database when the hash join feature is enabled and the instance is low on memory.
- Fixed the free memory calculation to include swap memory space on T2 instances that caused them to reboot prematurely.
- Fixed an issue in the query cache where the "Too many connections" error could cause a reboot.

Integration of MySQL community edition bug fixes

- CVE-2018-2696
- CVE-2015-4737
- Bug #19929406: HANDLE_FATAL_SIGNAL (SIG=11) IN __MEMMOVE_SSSE3_BACK FROM STRING::COPY
- Bug #17059925: For UNION statements, the rows-examined value was calculated incorrectly. This was manifested as too-large values for the `ROWS_EXAMINED` column of Performance Schema statement tables (such as events_statements_current).
- Bug #11827369: Some queries with `SELECT ... FROM DUAL` nested subqueries raised an assertion.
- Bug #16311231: Incorrect results were returned if a query contained a subquery in an `IN` clause that contained an XOR operation in the `WHERE` clause.

Aurora MySQL database engine updates 2019-06-05 (version 1.19.2) (Deprecated)

Version: 1.19.2

Aurora MySQL 1.19.2 is generally available. All new Aurora MySQL database clusters with MySQL 5.6 compatibility, including those restored from snapshots, can be created with 1.17.8, 1.19.0, 1.19.1, or 1.19.2. You have the option, but are not required, to upgrade existing database clusters to Aurora MySQL 1.19.2. To use an older version, you can create new database clusters in Aurora MySQL 1.14.4, Aurora MySQL 1.15.1, Aurora MySQL 1.16, Aurora MySQL 1.17.8, or Aurora MySQL 1.18. You can do so using the AWS CLI or the Amazon RDS API and specifying the engine version.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Note

This version is currently not available in the AWS GovCloud (US-West) [us-gov-west-1], Europe (Stockholm) [eu-north-1], China (Ningxia) [cn-northwest-1], and Asia Pacific (Hong Kong) [ap-east-1] AWS Regions. There will be a separate announcement once it is made available.

Note

The procedure to upgrade your DB cluster has changed. For more information, see Upgrading the minor version or patch level of an Aurora MySQL DB cluster in the *Amazon Aurora User Guide*.

Improvements

- Fixed an issue that could cause failures when loading data into Aurora from Amazon S3.
- Fixed an issue that could cause failures when uploading data from Aurora to Amazon S3.
- Fixed an issue that created zombie sessions left in a killed state.

- Fixed an issue that caused aborted connections when handling an error in network protocol management.
- Fixed an issue that could cause a crash when dealing with partitioned tables.
- Fixed an issue related to binlog replication of trigger creation.

Aurora MySQL database engine updates 2019-05-09 (version 1.19.1) (Deprecated)

Version: 1.19.1

Aurora MySQL 1.19.1 is generally available. All new Aurora MySQL database clusters with MySQL 5.6 compatibility, including those restored from snapshots, can be created with 1.17.8, 1.19.0, or 1.19.1. You have the option, but are not required, to upgrade existing database clusters to Aurora MySQL 1.19.1. To use an older version, you can create new database clusters in Aurora MySQL 1.14.4, Aurora MySQL 1.15.1, Aurora MySQL 1.16, Aurora MySQL 1.17.8, or Aurora MySQL 1.18. You can do so using the AWS CLI or the Amazon RDS API and specifying the engine version.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Note

This version is currently not available in the AWS GovCloud (US-West) [us-gov-west-1] and China (Beijing) [cn-north-1] regions. There will be a separate announcement once it is made available.

Note

The procedure to upgrade your DB cluster has changed. For more information, see Upgrading the minor version or patch level of an Aurora MySQL DB cluster in the *Amazon Aurora User Guide*.

Improvements

- Fixed a bug in binlog replication that can cause an issue on Aurora instances configured as binlog worker.
- Fixed an error in handling certain kinds of `ALTER TABLE` commands.
- Fixed an issue with aborted connections because of an error in network protocol management.

Aurora MySQL database engine updates 2019-02-07 (version 1.19.0) (Deprecated)

Version: 1.19.0

Aurora MySQL 1.19.0 is generally available. All new Aurora MySQL database clusters with MySQL 5.6 compatibility, including those restored from snapshots, can be created with 1.17.8 or 1.19.0. You have the option, but are not required, to upgrade existing database clusters to Aurora MySQL 1.19.0. To use an older version, you can create new database clusters in Aurora MySQL 1.14.4, Aurora MySQL 1.15.1, Aurora MySQL 1.16, Aurora MySQL 1.17.8, or Aurora MySQL 1.18.0. You can do so using the AWS CLI or the Amazon RDS API and specifying the engine version.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

> **Note**
>
> This version is currently not available in the AWS GovCloud (US-West) [us-gov-west-1] and China (Beijing) [cn-north-1] regions. There will be a separate announcement once it is made available.

> **Note**
>
> The procedure to upgrade your DB cluster has changed. For more information, see Upgrading the minor version or patch level of an Aurora MySQL DB cluster in the *Amazon Aurora User Guide*.

Features

- **Aurora Version Selector** - Starting with Aurora MySQL 1.19.0, you can choose from among multiple versions of MySQL 5.6 compatible Aurora on the Amazon RDS console. For more information, see Checking or specifying Aurora MySQL engine versions through AWS in the *Amazon Aurora User Guide*.

Improvements

- Fixed a stability issue related to the `CHECK TABLE` query on an Aurora Replica.
- Introduced a new global user variable `aurora_disable_hash_join` to disable Hash Join.
- Fixed a stability issue when generating the output row during multiple table hash join.
- Fixed an issue that returned a wrong result because of a plan change during Hash Join applicability check.
- Zero Downtime Patching is supported with long running transactions. This enhancement will come into effect when upgrading from version 1.19 to a higher one.
- Zero Downtime Patching is now supported when binlog is enabled. This enhancement will come into effect when upgrading from version 1.19 to a higher one.
- Fixed an issue that caused a spike in CPU utilization on the Aurora Replica unrelated to the workload.
- Fixed a race condition in the lock manager that resulted in a database restart.
- Fixed a race condition in the lock manager component to improve stability of Aurora instances.
- Improved stability of the deadlock detector inside the lock manager component.
- `INSERT` operation on a table is prohibited if InnoDB detects that the index is corrupted.
- Fixed a stability issue in Fast DDL.
- Improved Aurora stability by reducing the memory consumption in scan batching for single-row subquery.
- Fixed a stability issue that occurred after a foreign key was dropped while the system variable `foreign_key_checks` is set to 0.
- Fixed an issue in the Out Of Memory Avoidance feature that erroneously overrode changes to the `table_definition_cache` value made by the user.
- Fixed stability issues in the Out Of Memory Avoidance feature.

- Fixed an issue that set `query_time` and `lock_time` in `slow_query_log` to garbage values.
- Fixed a parallel query stability issue triggered by improper handling of string collation internally.
- Fixed a parallel query stability issue triggered by a secondary index search.
- Fixed a parallel query stability issue triggered by a multi-table update.

Integration of MySQL community edition bug fixes

- BUG #32917: DETECT ORPHAN TEMP-POOL FILES, AND HANDLE GRACEFULLY
- BUG #63144 CREATE TABLE IF NOT EXISTS METADATA LOCK IS TOO RESTRICTIVE

Aurora MySQL database engine updates 2018-09-20 (version 1.18.0) (Deprecated)

Version: 1.18.0

Aurora MySQL 1.18.0 is generally available. All new Aurora MySQL parallel query clusters with MySQL 5.6 compatibility, including those restored from snapshots, will be created in Aurora MySQL 1.18.0. You have the option, but are not required, to upgrade existing parallel query clusters to Aurora MySQL 1.18.0. You can create new DB clusters in Aurora MySQL 1.14.4, Aurora MySQL 1.15.1, Aurora MySQL 1.16, or Aurora MySQL 1.17.6. You can do so using the AWS CLI or the Amazon RDS API and specifying the engine version.

With version 1.18.0 of Aurora MySQL, we are using a cluster patching model where all nodes in an Aurora DB cluster are patched at the same time.

> **Important**
>
> Aurora MySQL 1.18.0 only applies to Aurora parallel query clusters. If you upgrade a provisioned 5.6.10a cluster, the resulting version is 1.17.8. If you upgrade a parallel query 5.6.10a cluster, the resulting version is 1.18.0.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Features

- **Parallel Query** is available with this release, for new clusters and restored snapshots. Aurora MySQL parallel query is an optimization that parallelizes some of the I/O and computation involved in processing data-intensive queries. The work that is parallelized includes retrieving rows from storage, extracting column values, and determining which rows match the conditions in the `WHERE` clause and join clauses. This data-intensive work is delegated (in database optimization terms, pushed down) to multiple nodes in the Aurora distributed storage layer. Without parallel query, each query brings all the scanned data to a single node within the Aurora MySQL cluster (the head node) and performs all the query processing there.
  - When the parallel query feature is enabled, the Aurora MySQL engine automatically determines when queries can benefit, without requiring SQL changes such as hints or table attributes.

  For more information, see Working with parallel query for Amazon Aurora MySQL in the *Amazon Aurora User Guide*.
- **OOM Avoidance:** This feature monitors the system memory and tracks memory consumed by various components of the database. Once the system runs low on memory, it performs a list of actions to release memory from various tracked components in an attempt to save the database from running into Out of Memory (OOM) and thereby avoiding a database restart. This best-effort feature is enabled by default for t2 instances and can be enabled on other instance classes via a new instance parameter named `aurora_oom_response`. The instance parameter takes a string of comma separated actions that an instance should take when its memory is low. Valid actions include "print", "tune", "decline", "kill_query" or any combination of these. Any empty string means there should be no actions taken and effectively renders the feature to be disabled. Note that the default actions for the feature is "print, tune". Usage examples:
  - "print" – Only prints the queries taking high amount of memory.
  - "tune" – Tunes the internal table caches to release some memory back to the system.
  - "decline" – Declines new queries once the instance is low on memory.
  - "kill_query" – Kills the queries in descending order of memory consumption until the instance memory surfaces above the low threshold. Data definition language (DDL) statements are not killed.
  - "print, tune" – Performs actions described for both "print" and "tune".
  - "tune, decline, kill_query" – Performs the actions described for "tune", "decline", and "kill_query".

For information about handling out-of-memory conditions and other troubleshooting advice, see Amazon Aurora MySQL out of memory issues in the *Amazon Aurora User Guide*.

Aurora MySQL database engine updates 2020-03-05 (version 1.17.9) (Deprecated)

Version: 1.17.9

Aurora MySQL 1.17.9 is generally available. Aurora MySQL 1.* versions are compatible with MySQL 5.6 and Aurora MySQL 2.* versions are compatible with MySQL 5.7.

Currently supported Aurora MySQL releases are 1.14.*, 1.15.*, 1.16.*, 1.17.*, 1.18.*, 1.19.*, 1.20.*, 1.21.*, 1.22.*, 2.01.*, 2.02.*, 2.03.*, 2.04.*, 2.05.*, 2.06.* and 2.07.*. You can restore the snapshot of an Aurora MySQL 1.* database into Aurora MySQL 1.17.9.

To create a cluster with an older version of Aurora MySQL, please specify the engine version through the RDS Console, the AWS CLI, or the Amazon RDS API.

> **Note**
>
> This version is currently not available in the following regions: AWS GovCloud (US-East) [us-gov-east-1], AWS GovCloud (US-West) [us-gov-west-1]. There will be a separate announcement once it is made available.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Improvements

High priority fixes:

- Fixed an issue of intermittent connection failures after certificate rotation.

Aurora MySQL database engine updates 2019-01-17 (version 1.17.8) (Deprecated)

Version: 1.17.8

Aurora MySQL 1.17.8 is generally available. All new Aurora MySQL database clusters with MySQL 5.6 compatibility, including those restored from snapshots, will be created in Aurora MySQL 1.17.8. You have the option, but are not required, to upgrade existing database clusters to Aurora MySQL 1.17.8. To use an older version, you can create new database clusters in Aurora MySQL 1.14.4, 1.15.1, 1.16, or 1.17.7. You can do so using the AWS CLI or the Amazon RDS API and specifying the engine version.

With version 1.17.8 of Aurora MySQL, we are using a cluster patching model where all nodes in an Aurora DB cluster are patched at the same time.

> **Note**
>
> This version is currently not available in the AWS GovCloud (US-West) [us-gov-west-1] and China (Beijing) [cn-north-1] regions. There will be a separate announcement once it is made available.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Improvements

- Fixed a performance issue that increased the CPU utilization on an Aurora Replica after a restart.
- Fixed a stability issue for SELECT queries that used hash join.

Integration of MySQL community edition bug fixes

- BUG #13418638: CREATE TABLE IF NOT EXISTS METADATA LOCK IS TOO RESTRICTIVE

Aurora MySQL database engine updates 2018-10-08 (version 1.17.7) (Deprecated)

Version: 1.17.7

Aurora MySQL 1.17.7 is generally available. All new Aurora MySQL database clusters with MySQL 5.6 compatibility, including those restored from snapshots, will be created in Aurora MySQL 1.17.7. You have the option, but are not required, to upgrade existing database clusters to Aurora MySQL 1.17.7. To use an older version, you can create new database clusters in Aurora MySQL 1.14.4, 1.15.1, 1.16, or 1.17.6. You can do so using the AWS CLI or the Amazon RDS API and specifying the engine version.

With version 1.17.7 of Aurora MySQL, we are using a cluster patching model where all nodes in an Aurora DB cluster are patched at the same time.

> **Note**
>
> This version is currently not available in the AWS GovCloud (US-West) [us-gov-west-1] and China (Beijing) [cn-north-1] regions. There will be a separate announcement once it is made available.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Improvements

- The InnoDB status variable `innodb_buffer_pool_size` has been made publicly visible for the customers to modify.
- Fixed a stability issue on the Aurora cluster that occurred during failovers.
- Improved cluster availability by fixing a DDL recovery issue that occurred after an unsuccessful `TRUNCATE` operation.
- Fixed a stability issue related to the `mysql.innodb_table_stats` table update, triggered by DDL operations.
- Fixed Aurora Replica stability issues triggered during query cache invalidation after a DDL operation.

- Fixed a stability issue triggered by invalid memory access during periodic dictionary cache eviction in the background.

Integration of MySQL community edition bug fixes

- Bug #16208542: Drop index on a foreign key column leads to missing table.
- Bug #76349: memory leak in add_derived_key().
- Bug #16862316: For partitioned tables, queries could return different results depending on whether Index Merge was used.
- Bug #17588348: Queries using the index_merge optimization (see Index merge optimization) could return invalid results when run against tables that were partitioned by HASH.

Aurora MySQL database engine updates 2018-09-06 (version 1.17.6) (Deprecated)

Version: 1.17.6

Aurora MySQL 1.17.6 is generally available. All new Aurora MySQL database clusters with MySQL 5.6 compatibility, including those restored from snapshots, will be created in Aurora MySQL 1.17.6. You have the option, but are not required, to upgrade existing database clusters to Aurora MySQL 1.17.6. To use an older version, you can create new database clusters in Aurora MySQL 1.14.4, 1.15.1, 1.16, or 1.17.5. You can do so using the AWS CLI or the Amazon RDS API and specifying the engine version.

With version 1.17.6 of Aurora MySQL, we are using a cluster patching model where all nodes in an Aurora DB cluster are patched at the same time.

> **Note**
>
> This version is currently not available in the AWS GovCloud (US-West) [us-gov-west-1] and China (Beijing) [cn-north-1] regions. There will be a separate announcement once it is made available.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Improvements

- Fixed a stability issue on the Aurora Reader for SELECT queries while the Aurora Writer is performing DDL operations on the same table.
- Fixed a stability issue caused by the creation and deletion of DDL logs for temporary tables that use Heap/Memory engine.
- Fixed a stability issue on the binlog worker when DDL statements are being replicated while the connection to the Binlog Master is unstable.
- Fixed a stability issue encountered while writing to the slow query log.
- Fixed an issue with the replica status table that exposed incorrect Aurora Reader lag information.

Integration of MySQL community edition bug fixes

- For an ALTER TABLE statement that renamed or changed the default value of a BINARY column, the alteration was done using a table copy and not in place. (Bug #67141, Bug #14735373, Bug #69580, Bug #17024290)
- An outer join between a regular table and a derived table that is implicitly groups could cause a server exit. (Bug #16177639)

Aurora MySQL database engine updates 2018-08-14 (version 1.17.5) (Deprecated)

Version: 1.17.5

Aurora MySQL 1.17.5 is generally available. All new Aurora MySQL database clusters with MySQL 5.6 compatibility, including those restored from snapshots, will be created in Aurora MySQL 1.17.5. You have the option, but are not required, to upgrade existing database clusters to Aurora MySQL 1.17.5. To use an older version, you can create new database clusters in Aurora MySQL 1.14.4, 1.15.1, 1.16, or 1.17.4. You can do so using the AWS CLI or the Amazon RDS API and specifying the engine version.

With version 1.17.5 of Aurora MySQL, we are using a cluster patching model where all nodes in an Aurora DB cluster are patched at the same time.

> **Note**
>
> This version is currently not available in the AWS GovCloud (US-West) [us-gov-west-1] and China (Beijing) [cn-north-1] regions. There will be a separate announcement once it is made available.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Improvements

- Fixed an issue where an Aurora Writer might experience a restart after an Aurora cluster is patched using the Zero-Downtime Patching feature.

Aurora MySQL database engine updates 2018-08-07 (version 1.17.4) (Deprecated)

Version: 1.17.4

Aurora MySQL 1.17.4 is generally available. All new Aurora MySQL database clusters with MySQL 5.6 compatibility, including those restored from snapshots, will be created in Aurora MySQL 1.17.4. You have the option, but are not required, to upgrade existing database clusters to Aurora MySQL 1.17.4. To use an older version, you can create new database clusters in Aurora MySQL 1.14.4, 1.15.1, 1.16, or 1.17.3. You can do so using the AWS CLI or the Amazon RDS API and specifying the engine version.

With version 1.17.4 of Aurora MySQL, we are using a cluster patching model where all nodes in an Aurora DB cluster are patched at the same time.

> **Note**
>
> This version is currently not available in the AWS GovCloud (US-West) [us-gov-west-1] and China (Beijing) [cn-north-1] regions. There will be a separate announcement once it is made available.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Improvements

- Replication improvements:
  - Reduced network traffic by not transmitting binlog records to cluster replicas. This improvement is enabled by default.
  - Reduced network traffic by compressing replication messages. This improvement is enabled by default for 8xlarge and 16xlarge instance classes. Such large instances can sustain a heavy volume of write traffic that results in substantial network traffic for replication messages.
  - Fixes to the replica query cache.
- Fixed an issue where `ORDER BY LOWER(col_name)` could produce incorrect ordering while using the `utf8_bin` collation.
- Fixed an issue where DDL statements (especially `TRUNCATE TABLE`) could cause problems on Aurora replicas, including instability or missing tables.
- Fixed an issue where sockets are left in a half-open state when storage nodes are restarted.
- The following new DB cluster parameters are available:
  - `aurora_enable_zdr` – Allow connections opened on an Aurora Replica to stay active on replica restart.
  - `aurora_enable_replica_log_compression` – Enable compression of replication payloads to improve network bandwidth utilization between the master and Aurora Replicas.
  - `aurora_enable_repl_bin_log_filtering` – Enable filtering of replication records that are unusable by Aurora Replicas on the master.

Aurora MySQL database engine updates 2018-06-05 (version 1.17.3) (Deprecated)

Version: 1.17.3

Aurora MySQL 1.17.3 is generally available. All new Aurora MySQL database clusters with MySQL 5.6 compatibility, including those restored from snapshots, will be created in Aurora MySQL 1.17.3. You have the option, but are not required, to upgrade existing database clusters to Aurora MySQL 1.17.3. You can create new database clusters in Aurora MySQL 1.14.4, Aurora MySQL 1.15.1, or Aurora MySQL 1.16. You can do so using the AWS CLI or the Amazon RDS API and specifying the engine version.

With version 1.17.3 of Aurora MySQL, we are using a cluster patching model where all nodes in an Aurora DB cluster are patched at the same time.

> **Note**
>
> This version is currently not available in the AWS GovCloud (US-West) [us-gov-west-1] and China (Beijing) [cn-north-1] regions. There will be a separate announcement once it is made available.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Improvements

- Fixed an issue where an Aurora Replica can restart when using optimistic cursor restores while reading records.
- Fixed an issue where an Aurora Writer restarts when trying to kill a MySQL session (kill "`<session id>`") with performance schema enabled.
- Fixed an issue where an Aurora Writer restarts when computing a threshold for garbage collection.
- Fixed an issue where an Aurora Writer can occasionally restart when tracking Aurora Replica progress in log application.

- Fixed an issue with the Query Cache when auto-commit is off and that could potentially cause stale reads.

Aurora MySQL database engine updates 2018-04-27 (version 1.17.2) (Deprecated)

Version: 1.17.2

Aurora MySQL 1.17.2 is generally available. All new Aurora MySQL database clusters with MySQL 5.6 compatibility, including those restored from snapshots, will be created in Aurora MySQL 1.17.2. You have the option, but are not required, to upgrade existing database clusters to Aurora MySQL 1.17.2. You can create new database clusters in Aurora MySQL 1.14.4, Aurora MySQL 1.15.1, or Aurora MySQL 1.16. You can do so using the AWS CLI or the Amazon RDS API and specifying the engine version.

With version 1.17.2 of Aurora MySQL, we are using a cluster patching model where all nodes in an Aurora DB cluster are patched at the same time.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Improvements

- Fixed an issue which was causing restarts during certain DDL partition operations.
- Fixed an issue which was causing support for invocation of AWS Lambda functions via native Aurora MySQL functions to be disabled.
- Fixed an issue with cache invalidation which was causing restarts on Aurora Replicas.
- Fixed an issue in lock manager which was causing restarts.

Aurora MySQL database engine updates 2018-03-23 (version 1.17.1) (Deprecated)

Version: 1.17.1

Aurora MySQL 1.17.1 is generally available. All new database clusters, including those restored from snapshots, will be created in Aurora MySQL 1.17.1. You have the option, but are not required, to upgrade existing database clusters to Aurora MySQL 1.17.1. You can create new DB clusters in Aurora MySQL 1.15.1, Aurora MySQL 1.16, or Aurora MySQL 1.17. You can do so using the AWS CLI or the Amazon RDS API and specifying the engine version.

With version 1.17.1 of Aurora MySQL, we are using a cluster patching model where all nodes in an Aurora DB cluster are patched at the same time. This release fixes some known engine issues as well as regressions.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

> **Note**
>
> There is an issue in the latest version of the Aurora MySQL engine. After upgrading to 1.17.1, the engine version is reported incorrectly as `1.17`. If you upgraded to 1.17.1, you can confirm the upgrade by checking the **Maintenance** column for the DB cluster in the AWS Management Console. If it displays none, then the engine is upgraded to 1.17.1.

Improvements

- Fixed an issue in binary log recovery that resulted in longer recovery times for situations with large binary log index files which can happen if binary logs rotate very often.
- Fixed an issue in the query optimizer that generated an inefficient query plan for partitioned tables.
- Fixed an issue in the query optimizer due to which a range query resulted in a restart of the database engine.

Aurora MySQL database engine updates 2018-03-13 (version 1.17) (Deprecated)

Version: 1.17

Aurora MySQL 1.17 is generally available. Aurora MySQL 1.x versions are only compatible with MySQL 5.6, and not MySQL 5.7. All new 5.6-compatible database clusters, including those restored from snapshots, will be created in Aurora 1.17. You have the option, but are not required, to upgrade existing database clusters to Aurora 1.17. You can create new DB clusters in Aurora 1.14.1, Aurora 1.15.1, or Aurora 1.16. You can do so using the AWS CLI or the Amazon RDS API and specifying the engine version.

With version 1.17 of Aurora, we are using a cluster patching model where all nodes in an Aurora DB cluster are patched at the same time. We support zero-downtime patching, which works on a best-effort basis to preserve client connections through the patching process. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support.

Zero-downtime patching

The zero-downtime patching (ZDP) feature attempts, on a best-effort basis, to preserve client connections through an engine patch. For more information about ZDP, see Using zero-downtime patching in the *Amazon Aurora User Guide*.

New features

- Aurora MySQL now supports lock compression, which optimizes the lock manager's memory usage. Starting in version 1.17, you can use this feature without enabling lab mode.

Improvements

- Fixed an issue predominantly seen on instances with fewer cores where a single core might have 100% CPU utilization even when the database is idle.
- Improved the performance of fetching binary logs from Aurora clusters.
- Fixed an issue where Aurora Replicas attempt to write table statistics to persistent storage, and crash.
- Fixed an issue where query cache did not work as expected on Aurora Replicas.
- Fixed a race condition in lock manager that resulted in an engine restart.
- Fixed an issue where locks taken by read-only, auto-commit transactions resulted in an engine restart.

- Fixed an issue where some queries are not written to the audit logs.
- Fixed an issue with recovery of certain partition maintenance operations on failover.

Integration of MySQL bug fixes

- LAST_INSERT_ID is replicated incorrectly if replication filters are used (Bug #69861)
- Query returns different results depending on whether INDEX_MERGE setting (Bug #16862316)
- Query proc re-execute of stored routine, inefficient query plan (Bug #16346367)
- INNODB FTS : Assert in FTS_CACHE_APPEND_DELETED_DOC_IDS (BUG #18079671)
- Assert RBT_EMPTY(INDEX_CACHE->WORDS) in ALTER TABLE CHANGE COLUMN (BUG #17536995)
- INNODB fulltext search doesn't find records when savepoints are involved (BUG #70333, BUG #17458835)

Aurora MySQL database engine updates 2017-12-11 (version 1.16) (Deprecated)

Version: 1.16

Aurora MySQL 1.16 is generally available. All new database clusters, including those restored from snapshots, will be created in Aurora 1.16. You have the option, but are not required, to upgrade existing database clusters to Aurora 1.16. You can create new DB clusters in Aurora 1.14.1 or Aurora 1.15.1. You can do so using the AWS CLI or the Amazon RDS API and specifying the engine version.

With version 1.16 of Aurora, we are using a cluster patching model where all nodes in an Aurora DB cluster are patched at the same time. We are enabling zero-downtime patching, which works on a best-effort basis to preserve client connections through the patching process. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support.

Zero-downtime patching

The zero-downtime patching (ZDP) feature attempts, on a best-effort basis, to preserve client connections through an engine patch. For more information about ZDP, see Using zero-downtime patching in the *Amazon Aurora User Guide*.

New features

- Aurora MySQL now supports synchronous AWS Lambda invocations via the native function `lambda_sync()`. Also available is native function `lambda_async()`, which can be used as an alternative to the existing stored procedure for asynchronous Lambda invocation. For more information, see Invoking a Lambda function from an Amazon Aurora MySQL DB cluster in the *Amazon Aurora User Guide*.
- Aurora MySQL now supports hash joins to speed up equijoin queries. Aurora's cost-based optimizer can automatically decide when to use hash joins; you can also force their use in a query plan. For more information, see Optimizing large Aurora MySQL join queries with hash joins in the *Amazon Aurora User Guide*.
- Aurora MySQL now supports scan batching to speed up in-memory scan-oriented queries significantly. The feature boosts the performance of table full scans, index full scans, and index range scans by batch processing.

Improvements

- Fixed an issue where read replicas crashed when running queries on tables that have just been dropped on the master.
- Fixed an issue where restarting the writer on a database cluster with a very large number of `FULLTEXT` indexes results in longer than expected recovery.
- Fixed an issue where flushing binary logs causes `LOST_EVENTS` incidents in binlog events.
- Fixed stability issues with the scheduler when performance schema is enabled.
- Fixed an issue where a subquery that uses temporary tables could return partial results.

Integration of MySQL bug fixes

None

Aurora MySQL database engine updates 2017-11-20 (version 1.15.1) (Deprecated)

Version: 1.15.1

Aurora MySQL 1.15.1 is generally available. All new database clusters, including those restored from snapshots, will be created in Aurora 1.15.1. You have the option, but are not required, to upgrade existing DB clusters to Aurora 1.15.1. You can create new DB clusters in Aurora 1.14.1. You can do so using the AWS CLI or the Amazon RDS API and specifying the engine version.

With version 1.15.1 of Aurora, we are using a cluster patching model where all nodes in an Aurora DB cluster are patched at the same time. We are enabling zero-downtime patching, which works on a best-effort basis to preserve client connections through the patching process. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Zero-downtime patching

The zero-downtime patching (ZDP) feature attempts, on a best-effort basis, to preserve client connections through an engine patch. For more information about ZDP, see Using zero-downtime patching in the *Amazon Aurora User Guide*.

Improvements

- Fixed an issue in the adaptive segment selector for a read request that would cause it to choose the same segment twice causing a spike in read latency under certain conditions.
- Fixed an issue that stems from an optimization in Aurora MySQL for the thread scheduler. This problem manifests itself into what are spurious errors while writing to the slow log, while the associated queries themselves perform fine.
- Fixed an issue with stability of read replicas on large (> 5 TB) volumes.
- Fixed an issue where worker thread count increases continuously due to a bogus outstanding connection count.
- Fixed an issue with table locks that caused long semaphore waits during insert workloads.

- Reverted the following MySQL bug fixes included in Aurora MySQL 1.15:
  - MySQL instance stalling "doing SYNC index" (Bug #73816)
  - Assert RBT_EMPTY(INDEX_CACHE->WORDS) in ALTER TABLE CHANGE COLUMN (Bug #17536995)
  - InnoDB Fulltext search doesn't find records when savepoints are involved (Bug #70333)

Integration of MySQL bug fixes

None

Aurora MySQL database engine updates 2017-10-24 (version 1.15) (Deprecated)

Version: 1.15

Aurora MySQL 1.15 is generally available. All new database clusters, including those restored from snapshots, will be created in Aurora 1.15. You have the option, but are not required, to upgrade existing DB clusters to Aurora 1.15. You can create new DB clusters in Aurora 1.14.1. You can do so using the AWS CLI or the Amazon RDS API and specifying the engine version.

With version 1.15 of Aurora, we are using a cluster patching model where all nodes in an Aurora DB cluster are patched at the same time. Updates require a database restart, so you will experience 20 to 30 seconds of downtime, after which you can resume using your DB cluster or clusters. If your DB clusters are currently running Aurora 1.14 or Aurora 1.14.1, the zero-downtime patching feature in Aurora MySQL might allow client connections to your Aurora MySQL primary instance to persist through the upgrade, depending on your workload.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Zero-downtime patching

The zero-downtime patching (ZDP) feature attempts, on a best-effort basis, to preserve client connections through an engine patch. For more information about ZDP, see Using zero-downtime patching in the *Amazon Aurora User Guide*.

New features

- **Asynchronous Key Prefetch** – Asynchronous key prefetch (AKP) is a feature targeted to improve the performance of non-cached index joins, by prefetching keys in memory ahead of when they are needed. The primary use case targeted by AKP is an index join between a small outer and large inner table, where the index is highly selective on the larger table. Also, when the Multi-Range Read (MRR) interface is enabled, AKP will be leveraged for a secondary to primary index lookup. Smaller instances which have memory constraints might in some cases be able to leverage AKP, given the right key cardinality. For more information, see Optimizing Aurora indexed join queries with asynchronous key prefetch in the *Amazon Aurora User Guide*.
- **Fast DDL** – We have extended the feature that was released in Aurora 1.13 to operations that include default values. With this extension, Fast DDL is applicable for operations that add a nullable column at the end of a table, with or without default values. The feature remains under Aurora lab mode. For more information, see Altering tables in Amazon Aurora using fast DDL in the *Amazon Aurora User Guide*.

Improvements

- Fixed a calculation error during optimization of WITHIN/CONTAINS spatial queries which previously resulted in an empty result set.
- Fixed `SHOW VARIABLE` command to show the updated `innodb_buffer_pool_size` parameter value whenever it is changed in the parameter group.
- Improved stability of primary instance during bulk insert into a table altered using Fast DDL when adaptive hash indexing is disabled and the record to be inserted is the first record of a page.
- Improved stability of Aurora when the user attempts to set **server_audit_events** DB cluster parameter value to **`default`**.
- Fixed an issue in which a database character set change for an ALTER TABLE statement that ran on the Aurora primary instance was not being replicated on the Aurora Replicas until they were restarted.
- Improved stability by fixing a race condition on the primary instance which previously allowed it to register an Aurora Replica even if the primary instance had closed its own volume.
- Improved performance of the primary instance during index creation on a large table by changing the locking protocol to enable concurrent data manipulation language (DML) statements during index build.

- Fixed InnoDB metadata inconsistency during ALTER TABLE RENAME query which improved stability. Example: When columns of table t1(c1, c2) are renamed cyclically to t1(c2,c3) within the same ALTER statement.
- Improved stability of Aurora Replicas for the scenario where an Aurora Replica has no active workload and the primary instance is unresponsive.
- Improved availability of Aurora Replicas for a scenario in which the Aurora Replica holds an explicit lock on a table and blocks the replication thread from applying any DDL changes received from the primary instance.
- Improved stability of the primary instance when a foreign key and a column are being added to a table from two separate sessions at the same time and Fast DDL has been enabled.
- Improved stability of the purge thread on the primary instance during a heavy write workload by blocking truncate of undo records until they have been purged.
- Improved stability by fixing the lock release order during commit process of transactions which drop tables.
- Fixed a defect for Aurora Replicas in which the DB instance could not complete startup and complained that port 3306 was already in use.
- Fixed a race condition in which a SELECT query run on certain information_schema tables (innodb_trx, innodb_lock, innodb_lock_waits) increased cluster instability.

Integration of MySQL bug fixes

- CREATE USER accepts plugin and password hash, but ignores the password hash (Bug #78033)
- The partitioning engine adds fields to the read bit set to be able to return entries sorted from a partitioned index. This leads to the join buffer will try to read unneeded fields. Fixed by not adding all partitioning fields to the read_set,but instead only sort on the already set prefix fields in the read_set. Added a DBUG_ASSERT that if doing key_cmp, at least the first field must be read (Bug #16367691).
- MySQL instance stalling "doing SYNC index" (Bug #73816)
- Assert RBT_EMPTY(INDEX_CACHE->WORDS) in ALTER TABLE CHANGE COLUMN (Bug #17536995)
- InnoDB Fulltext search doesn't find records when savepoints are involved (Bug #70333)

Aurora MySQL database engine updates: 2018-03-13 (version 1.14.4) (Deprecated)

Version: 1.14.4

Aurora MySQL 1.14.4 is generally available. You can create new DB clusters in Aurora 1.14.4, using the AWS CLI or the Amazon RDS API and specifying the engine version. You have the option, but are not required, to upgrade existing 1.14.x DB clusters to Aurora 1.14.4.

With version 1.14.4 of Aurora, we are using a cluster-patching model where all nodes in an Aurora DB cluster are patched at the same time. We support zero-downtime patching, which works on a best-effort basis to preserve client connections through the patching process. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Zero-downtime patching

The zero-downtime patching (ZDP) feature attempts, on a best-effort basis, to preserve client connections through an engine patch. For more information about ZDP, see Using zero-downtime patching in the *Amazon Aurora User Guide*.

New features

- Aurora MySQL now supports db.r4 instance classes.

Improvements

- Fixed an issue where `LOST_EVENTS` were generated when writing large binlog events.

Integration of MySQL bug fixes

- Ignorable events don't work and are not tested (Bug #74683)
- NEW->OLD ASSERT FAILURE 'GTID_MODE > 0' (Bug #20436436)

Aurora MySQL database engine updates: 2017-09-22 (version 1.14.1) (Deprecated)

Version: 1.14.1

Aurora MySQL 1.14.1 is generally available. All new database clusters, including those restored from snapshots, will be created in Aurora MySQL 1.14.1. Aurora MySQL 1.14.1 is also a mandatory upgrade for existing Aurora MySQL DB clusters. For more information, see Announcement: Extension to mandatory upgrade schedule for Amazon Aurora on the AWS Developer Forums website.

With version 1.14.1 of Aurora MySQL, we are using a cluster patching model where all nodes in an Aurora MySQL DB cluster are patched at the same time. Updates require a database restart, so you will experience 20 to 30 seconds of downtime, after which you can resume using your DB cluster or clusters. If your DB clusters are currently running version 1.13 or greater, the zero-downtime patching feature in Aurora MySQL might allow client connections to your Aurora MySQL primary instance to persist through the upgrade, depending on your workload.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support.

Improvements

- Fixed race conditions associated with inserts and purge to improve the stability of the Fast DDL feature, which remains in Aurora MySQL lab mode.

Aurora MySQL database engine updates: 2017-08-07 (version 1.14) (Deprecated)

Version: 1.14

Aurora MySQL 1.14 is generally available. All new database clusters, including those restored from snapshots, will be created in Aurora MySQL 1.14. Aurora MySQL 1.14 is also a mandatory upgrade for existing Aurora MySQL DB clusters. We will send a separate announcement with the timeline for deprecating earlier versions of Aurora MySQL.

With version 1.14 of Aurora MySQL, we are using a cluster patching model where all nodes in an Aurora DB cluster are patched at the same time. Updates require a database restart, so you will

experience 20 to 30 seconds of downtime, after which you can resume using your DB cluster or clusters. If your DB clusters are currently running version 1.13, Aurora's zero-downtime patching feature may allow client connections to your Aurora primary instance to persist through the upgrade, depending on your workload.

If you have any questions or concerns, AWS Support is available on the community forums and through AWS Support.

Zero-downtime patching

The zero-downtime patching (ZDP) feature attempts, on a best-effort basis, to preserve client connections through an engine patch. For more information about ZDP, see Using zero-downtime patching in the *Amazon Aurora User Guide*.

Improvements

- Fixed an incorrect "record not found" error when a record is found in the secondary index but not in the primary index.
- Fixed a stability issue that can occur due to a defensive assertion (added in 1.12) that was too strong in the case when an individual write spans over 32 pages. Such a situation can occur, for instance, with large BLOB values.
- Fixed a stability issue because of inconsistencies between the tablespace cache and the dictionary cache.
- Fixed an issue in which an Aurora Replica becomes unresponsive after it has exceeded the maximum number of attempts to connect to the primary instance. An Aurora Replica now restarts if the period of inactivity is more than the heartbeat time period used for health check by the primary instance.
- Fixed a livelock that can occur under very high concurrency when one connection tries to acquire an exclusive meta data lock (MDL) while issuing a command, such as `ALTER TABLE`.
- Fixed a stability issue in an Aurora Read Replica in the presence of logical/parallel read ahead.
- Improved `LOAD FROM S3` in two ways:
  1. Better handling of Amazon S3 timeout errors by using the SDK retry in addition to the existing retry.
  2. Performance optimization when loading very big files or large numbers of files by caching and reusing client state.
- Fixed the following stability issues with Fast DDL for `ALTER TABLE` operations:

1. When the `ALTER TABLE` statement has multiple `ADD COLUMN` commands and the column names are not in ascending order.
2. When the name string of the column to be updated and its corresponding name string, fetched from the internal system table, differs by a null terminating character (/0).
3. Under certain B-tree split operations.
4. When the table has a variable length primary key.

- Fixed a stability issue with Aurora Replicas when it takes too long to make its Full Text Search (FTS) index cache consistent with that of the primary instance. This can happen if a large portion of the newly created FTS index entries on the primary instance have not yet been flushed to disk.
- Fixed a stability issue that can happen during index creation.
- New infrastructure that tracks memory consumption per connection and associated telemetry that will be used for building out Out-Of-Memory (OOM) avoidance strategies.
- Fixed an issue where `ANALYZE TABLE` was incorrectly allowed on Aurora Replicas. This has now been blocked.
- Fixed a stability issue caused by a rare deadlock as a result of a race condition between logical read-ahead and purge.

Integration of MySQL bug fixes

- A full-text search combined with derived tables (subqueries in the `FROM` clause) caused a server exit. Now, if a full-text operation depends on a derived table, the server produces an error indicating that a full-text search cannot be done on a materialized table. (Bug #68751, Bug #16539903)

Aurora MySQL database engine updates: 2017-05-15 (version 1.13) (Deprecated)

Version: 1.13

> **Note**
>
> We enabled a new feature - SELECT INTO OUTFILE S3 - in Aurora MySQL version 1.13 after the initial release, and have updated the release notes to reflect that change.

Aurora MySQL 1.13 is generally available. All new database clusters, including those restored from snapshots, will be created in Aurora MySQL 1.13. You have the option, but are not required, to upgrade existing database clusters to Aurora MySQL 1.13. With version 1.13 of Aurora, we are using a cluster patching model where all nodes in an Aurora DB cluster are patched at the same time. We are enabling zero-downtime patching, which works on a best-effort basis to preserve client connections through the patching process. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Zero-downtime patching

The zero-downtime patching (ZDP) feature attempts, on a best-effort basis, to preserve client connections through an engine patch. For more information about ZDP, see Using zero-downtime patching in the *Amazon Aurora User Guide*.

New features:

- **SELECT INTO OUTFILE S3** – Aurora MySQL now allows you to upload the results of a query to one or more files in an Amazon S3 bucket. For more information, see Saving data from an Amazon Aurora MySQL DB cluster into text files in an Amazon S3 bucket in the *Amazon Aurora User Guide*.

Improvements:

- Implemented truncation of CSV format log files at engine startup to avoid long recovery time. The `general_log_backup`, `general_log`, `slow_log_backup`, and `slow_log` tables now don't survive a database restart.
- Fixed an issue where migration of a database named **`test`** would fail.
- Improved stability in the lock manager's garbage collector by reusing the correct lock segments.
- Improved stability of the lock manager by removing invalid assertions during deadlock detection algorithm.
- Re-enabled asynchronous replication, and fixed an associated issue which caused incorrect replica lag to be reported under no-load or read-only workload. The replication pipeline improvements that were introduced in version 1.10. These improvements were introduced in order to apply log stream updates to the buffer cache of an Aurora Replica. which helps to improve read performance and stability on Aurora Replicas.

- Fixed an issue where autocommit=OFF leads to scheduled events being blocked and long transactions being held open until the server reboots.
- Fixed an issue where general, audit, and slow query logs could not log queries handled by asynchronous commit.
- Improved the performance of the logical read ahead (LRA) feature by up to 2.5 times. This was done by allowing pre-fetches to continue across intermediate pages in a B-tree.
- Added parameter validation for audit variables to trim unnecessary spaces.
- Fixed a regression, introduced in Aurora MySQL version 1.11, in which queries can return incorrect results when using the SQL_CALC_FOUND_ROWS option and invoking the FOUND_ROWS() function.
- Fixed a stability issue when the Metadata Lock list was incorrectly formed.
- Improved stability when sql_mode is set to PAD_CHAR_TO_FULL_LENGTH and the command `SHOW FUNCTION STATUS WHERE Db='string'` is executed.
- Fixed a rare case when instances would not come up after Aurora version upgrade because of a false volume consistency check.
- Fixed the performance issue, introduced in Aurora MySQL version 1.12, where the performance of the Aurora writer was reduced when users have a large number of tables.
- Improved stability issue when the Aurora writer is configured as a binlog worker and the number of connections approaches 16,000.
- Fixed a rare issue where an Aurora Replica could restart when a connection gets blocked waiting for Metadata Lock when running DDL on the Aurora master.

Integration of MySQL bug fixes

- With an empty InnoDB table, it's not possible to decrease the auto_increment value using an ALTER TABLE statement, even when the table is empty. (Bug #69882)
- MATCH() ... AGAINST queries that use a long string as an argument for AGAINST() could result in an error when run on an InnoDB table with a full-text search index. (Bug #17640261)
- Handling of SQL_CALC_FOUND_ROWS in combination with ORDER BY and LIMIT could lead to incorrect results for FOUND_ROWS(). (Bug #68458, Bug # 16383173)
- ALTER TABLE does not allow to change nullability of the column if foreign key exists. (Bug #77591)

Aurora MySQL database engine updates: 2017-04-05 (version 1.12) (Deprecated)

Version: 1.12

Aurora MySQL 1.12 is now the preferred version for the creation of new DB clusters, including restores from snapshots.

This is not a mandatory upgrade for existing clusters. You will have the option to upgrade existing clusters to version 1.12 after we complete the fleet-wide patch to 1.11 (see Aurora 1.11 release notes and corresponding forum announcement). With version 1.12 of Aurora, we are using a cluster patching model where all nodes in an Aurora DB cluster are patched at the same time. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

New features

- **Fast DDL** – Aurora MySQL now allows you to execute an ALTER TABLE *tbl_name* ADD COLUMN *col_name column_definition* operation nearly instantaneously. The operation completes without requiring the table to be copied and without materially impacting other DML statements. Since it does not consume temporary storage for a table copy, it makes DDL statements practical even for large tables on small instance classes. Fast DDL is currently only supported for adding a nullable column, without a default value, at the end of a table. This feature is currently available in Aurora lab mode. For more information, see Altering tables in Amazon Aurora using fast DDL in the *Amazon Aurora User Guide*.
- **Show volume status** – We have added a new monitoring command, SHOW VOLUME STATUS, to display the number of nodes and disks in a volume. For more information, see Displaying volume status for an Aurora MySQL DB cluster in the *Amazon Aurora User Guide*.

Improvements

- Implemented changes to lock compression to further reduce memory allocated per lock object. This improvement is available in lab mode.
- Fixed an issue where the `trx_active_transactions` metric decrements rapidly even when the database is idle.
- Fixed an invalid error message regarding fault injection query syntax when simulating failure in disks and nodes.

- Fixed multiple issues related to race conditions and dead latches in the lock manager.
- Fixed an issue causing a buffer overflow in the query optimizer.
- Fixed a stability issue in Aurora read replicas when the underlying storage nodes experience low available memory.
- Fixed an issue where idle connections persisted past the `wait_timeout` parameter setting.
- Fixed an issue where `query_cache_size` returns an unexpected value after reboot of the instance.
- Fixed a performance issue that is the result of a diagnostic thread probing the network too often in the event that writes are not progressing to storage.

Integration of MySQL bug fixes

- Reloading a table that was evicted while empty caused an AUTO_INCREMENT value to be reset. (Bug #21454472, Bug #77743)
- An index record was not found on rollback due to inconsistencies in the purge_node_t structure. The inconsistency resulted in warnings and error messages such as "error in sec index entry update", "unable to purge a record", and "tried to purge sec index entry not marked for deletion". (Bug #19138298, Bug #70214, Bug #21126772, Bug #21065746)
- Wrong stack size calculation for qsort operation leads to stack overflow. (Bug #73979)
- Record not found in an index upon rollback. (Bug #70214, Bug #72419)
- ALTER TABLE add column TIMESTAMP on update CURRENT_TIMESTAMP inserts ZERO-datas (Bug #17392)

Aurora MySQL database engine updates: 2017-02-23 (version 1.11) (Deprecated)

Version: 1.11

We will patch all Aurora MySQL DB clusters with the latest version over a short period following the release. DB clusters are patched using the legacy procedure with a downtime of about 5-30 seconds.

Patching occurs during the system maintenance window that you have specified for each of your database instances. You can view or change this window using the AWS Management Console. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Alternatively, you can apply the patch immediately in the AWS Management Console by choosing a DB cluster, choosing **Cluster Actions**, and then choosing **Upgrade Now**.

With version 1.11 of Aurora MySQL, we are using a cluster patching model where all nodes in an Aurora DB cluster are patched at the same time.

New features

- **MANIFEST option for LOAD DATA FROM S3** – LOAD DATA FROM S3 was released in version 1.8. The options for this command have been expanded, and you can now specify a list of files to be loaded into an Aurora DB cluster from Amazon S3 by using a manifest file. This makes it easy to load data from specific files in one or more locations, as opposed to loading data from a single file by using the FILE option or loading data from multiple files that have the same location and prefix by using the PREFIX option. The manifest file format is the same as that used by Amazon Redshift. For more information about using LOAD DATA FROM S3 with the MANIFEST option, see Using a manifest to specify data files to load in the *Amazon Aurora User Guide*.
- **Spatial indexing enabled by default** – This feature was released in lab mode in version 1.10, and is now turned on by default. Spatial indexing improves query performance on large datasets for queries that use spatial data. For more information about using spatial indexing, see Amazon Aurora MySQL and spatial data in the *Amazon Aurora User Guide*.
- **Advanced Auditing timing change** – This feature was released in version 1.10.1 to provide a high-performance facility for auditing database activity. In this release, the precision of audit log timestamps has been changed from one second to one microsecond. The more accurate timestamps allow you to better understand when an audit event happened. For more information about audit, see Using Advanced Auditing with an Amazon Aurora MySQL DB cluster in the *Amazon Aurora User Guide*.

Improvements

- Modified the `thread_handling` parameter to prevent you from setting it to anything other than **`multiple-connections-per-thread`**, which is the only supported model with Aurora's thread pool.

- Fixed an issue caused when you set either the `buffer_pool_size` or the `query_cache_size` parameter to be larger than the DB cluster's total memory. In this circumstance, Aurora sets the modified parameter to the default value, so the DB cluster can start up and not crash.
- Fixed an issue in the query cache where a transaction gets stale read results if the table is invalidated in another transaction.
- Fixed an issue where binlog files marked for deletion are removed after a small delay rather than right away.
- Fixed an issue where a database created with the name **`tmp`** is treated as a system database stored on ephemeral storage and not persisted to Aurora distributed storage.
- Modified the behavior of SHOW TABLES to exclude certain internal system tables. This change helps avoid an unnecessary failover caused by mysqldump locking all files listed in SHOW TABLES, which in turn prevents writes on the internal system table, causing the failover.
- Fixed an issue where an Aurora Replica incorrectly restarts when creating a temporary table from a query that invokes a function whose argument is a column of an InnoDB table.
- Fixed an issue related to a metadata lock conflict in an Aurora Replica node that causes the Aurora Replica to fall behind the primary DB cluster and eventually get restarted.
- Fixed a dead latch in the replication pipeline in reader nodes, which causes an Aurora Replica to fall behind and eventually get restarted.
- Fixed an issue where an Aurora Replica lags too much with encrypted volumes larger than 1 terabyte (TB).
- Improved Aurora Replica dead latch detection by using an improved way to read the system clock time.
- Fixed an issue where an Aurora Replica can restart twice instead of once following de-registration by the writer.
- Fixed a slow query performance issue on Aurora Replicas that occurs when transient statistics cause statistics discrepancy on non-unique index columns.
- Fixed an issue where an Aurora Replica can crash when a DDL statement is replicated on the Aurora Replica at the same time that the Aurora Replica is processing a related query.
- Changed the replication pipeline improvements that were introduced in version 1.10 from enabled by default to disabled by default. These improvements were introduced in order to apply log stream updates to the buffer cache of an Aurora Replica, and although this feature helps to improve read performance and stability on Aurora Replicas, it increases replica lag in certain workloads.

- Fixed an issue where the simultaneous occurrence of an ongoing DDL statement and pending Parallel Read Ahead on the same table causes an assertion failure during the commit phase of the DDL transaction.
- Enhanced the general log and slow query log to survive DB cluster restart.
- Fixed an out-of-memory issue for certain long running queries by reducing memory consumption in the ACL module.
- Fixed a restart issue that occurs when a table has non-spatial indexes, there are spatial predicates in the query, the planner chooses to use a non-spatial index, and the planner incorrectly pushes the spatial condition down to the index.
- Fixed an issue where the DB cluster restarts when there is a delete, update, or purge of very large geospatial objects that are stored externally (like LOBs).
- Fixed an issue where fault simulation using ALTER SYSTEM SIMULATE ... FOR INTERVAL isn't working properly.
- Fixed a stability issue caused by an invalid assertion on an incorrect invariant in the lock manager.
- Disabled the following two improvements to InnoDB Full-Text Search that were introduced in version 1.10 because they introduce stability issues for some demanding workloads:
  - Updating the cache only after a read request to an Aurora Replica in order to improve full-text search index cache replication speed.
  - Offloading the cache sync task to a separate thread as soon as the cache size crosses 10% of the total size, in order to avoid MySQL queries stalling for too long during FTS cache sync to disk. (Bugs #22516559, #73816).

Integration of MySQL bug fixes

- Running ALTER table DROP foreign key simultaneously with another DROP operation causes the table to disappear. (Bug #16095573)
- Some INFORMATION_SCHEMA queries that used ORDER BY did not use a filesort optimization as they did previously. (Bug #16423536)
- FOUND_ROWS () returns the wrong count of rows on a table. (Bug #68458)
- The server fails instead of giving an error when too many temp tables are open. (Bug #18948649)

Aurora MySQL database engine updates: 2017-01-12 (version 1.10.1) (Deprecated)

Version: 1.10.1

Version 1.10.1 of Aurora MySQL is an opt-in version and is not used to patch your database instances. It is available for creating new Aurora instances and for upgrading existing instances. You can apply the patch by choosing a cluster in the Amazon RDS console, choosing **Cluster Actions**, and then choosing **Upgrade Now**. Patching requires a database restart with downtime typically lasting 5-30 seconds, after which you can resume using your DB clusters. This patch is using a cluster patching model where all nodes in an Aurora cluster are patched at the same time.

New features

- **Advanced Auditing** – Aurora MySQL provides a high-performance Advanced Auditing feature, which you can use to audit database activity. For more information about enabling and using Advanced Auditing, see Using Advanced Auditing with an Aurora MySQL DB cluster in the *Amazon Aurora User Guide*.

Improvements

- Fixed an issue with spatial indexing when creating a column and adding an index on it in the same statement.
- Fixed an issue where spatial statistics aren't persisted across DB cluster restart.

Aurora MySQL database engine updates: 2016-12-14 (version 1.10) (Deprecated)

Version: 1.10

New features

- **Zero downtime patch** – This feature allows a DB instance to be patched without any downtime. That is, database upgrades are performed without disconnecting client applications, or rebooting the database. This approach increases the availability of your Aurora DB clusters during the maintenance window. Note that temporary data like that in the performance schema is

reset during the upgrade process. This feature applies to service-delivered patches during a maintenance window as well as user-initiated patches.

When a patch is initiated, the service ensures there are no open locks, transactions or temporary tables, and then waits for a suitable window during which the database can be patched and restarted. Application sessions are preserved, although there is a drop in throughput while the patch is in progress (for approximately 5 seconds). If no suitable window can be found, then patching defaults to the standard patching behavior.

Zero downtime patching takes place on a best-effort basis, subject to certain limitations as described following:

- This feature is currently applicable for patching single-node DB clusters or writer instances in multi-node DB clusters.
- SSL connections are not supported in conjunction with this feature. If there are active SSL connections, Amazon Aurora MySQL won't perform a zero downtime patch, and instead will retry periodically to see if the SSL connections have terminated. If they have, zero downtime patching proceeds. If the SSL connections persist after more than a couple seconds, standard patching with downtime proceeds.
- The feature is available in Aurora release 1.10 and beyond. Going forward, we will identify any releases or patches that can't be applied by using zero downtime patching.
- This feature is not applicable if replication based on binary logging is active.

- **Spatial indexing** – Spatial indexing improves query performance on large datasets for queries that use spatial data. For more information about using spatial indexing, see Amazon Aurora MySQL and spatial data in the *Amazon Aurora User Guide*.

  This feature is disabled by default and can be activated by enabling Aurora lab mode. For information, see Amazon Aurora MySQL lab mode in the *Amazon Aurora User Guide*.
- **Replication pipeline improvements** – Aurora MySQL now uses an improved mechanism to apply log stream updates to the buffer cache of an Aurora Replica. This feature improves the read performance and stability on Aurora Replicas when there is a heavy write load on the master as well as a significant read load on the Replica. This feature is enabled by default.
- **Throughput improvement for workloads with cached reads** – Aurora MySQL now uses a latch-free concurrent algorithm to implement read views, which leads to better throughput for read queries served by the buffer cache. As a result of this and other improvements, Amazon Aurora MySQL can achieve throughput of up to 625K reads per second compared to 164K reads per second by MySQL 5.7 for a SysBench SELECT-only workload.

- **Throughput improvement for workloads with hot row contention** – Aurora MySQL uses a new lock release algorithm that improves performance, particularly when there is hot page contention (that is, many transactions contending for the rows on the same page). In tests with the TPC-C benchmark, this can result in up to16x throughput improvement in transactions per minute relative to MySQL 5.7. This feature is disabled by default and can be activated by enabling Aurora lab mode. For information, see Amazon Aurora MySQL lab mode in the *Amazon Aurora User Guide*.

Improvements

- Full-text search index cache replication speed has been improved by updating the cache only after a read request to an Aurora Replica. This approach avoids any reads from disk by the replication thread.
- Fixed an issue where dictionary cache invalidation does not work on an Aurora Replica for tables that have a special character in the database name or table name.
- Fixed a `STUCK IO` issue during data migration for distributed storage nodes when storage heat management is enabled.
- Fixed an issue in the lock manager where an assertion check fails for the transaction lock wait thread when preparing to rollback or commit a transaction.
- Fixed an issue when opening a corrupted dictionary table by correctly updating the reference count to the dictionary table entries.
- Fixed a bug where the DB cluster minimum read point can be held by slow Aurora Replicas.
- Fixed a potential memory leak in the query cache.
- Fixed a bug where an Aurora Replica places a row-level lock on a table when a query is used in an `IF` statement of a stored procedure.

Integration of MySQL bug fixes

- UNION of derived tables returns wrong results with '1=0/false'-clauses. (Bug #69471)
- Server crashes in ITEM_FUNC_GROUP_CONCAT::FIX_FIELDS on 2nd execution of stored procedure. (Bug #20755389)
- Avoid MySQL queries from stalling for too long during FTS cache sync to disk by offloading the cache sync task to a separate thread, as soon as the cache size crosses 10% of the total size. (Bug #22516559, #73816)

Aurora MySQL database engine updates: 2016-11-10 (versions 1.9.0, 1.9.1) (Deprecated)

Version: 1.9.0, 1.9.1

New features

- **Improved index build** – The implementation for building secondary indexes now operates by building the index in a bottom-up fashion, which eliminates unnecessary page splits. This can reduce the time needed to create an index or rebuild a table by up to 75% (based on an `db.r3.8xlarge` DB instance class). This feature was in lab mode in Aurora MySQL version 1.7 and is enabled by default in Aurora version 1.9 and later. For information, see Amazon Aurora MySQL lab mode in the *Amazon Aurora User Guide*.
- **Lock compression (lab mode)** – This implementation significantly reduces the amount of memory that lock manager consumes by up to 66%. Lock manager can acquire more row locks without encountering an out-of-memory exception. This feature is disabled by default and can be activated by enabling Aurora lab mode. For information, see Amazon Aurora MySQL lab mode in the *Amazon Aurora User Guide*.
- **Performance schema** – Aurora MySQL now includes support for performance schema with minimal impact on performance. In our testing using SysBench, enabling performance schema could degrade MySQL performance by up to 60%.

  SysBench testing of an Aurora DB cluster showed an impact on performance that is 4x less than MySQL. Running the `db.r3.8xlarge` DB instance class resulted in 100K SQL writes/sec and over 550K SQL reads/sec, even with performance schema enabled.
- **Hot row contention improvement** – This feature reduces CPU utilization and increases throughput when a small number of hot rows are accessed by a large number of connections. This feature also eliminates `error 188` when there is hot row contention.
- **Improved out-of-memory handling** – When non-essential, locking SQL statements are executed and the reserved memory pool is breached, Aurora forces rollback of those SQL statements. This feature frees memory and prevents engine crashes due to out-of-memory exceptions.
- **Smart read selector** – This implementation improves read latency by choosing the optimal storage segment among different segments for every read, resulting in improved read throughput. SysBench testing has shown up to a 27% performance increase for write workloads .

Improvements

- Fixed an issue where an Aurora Replica encounters a shared lock during engine start up.
- Fixed a potential crash on an Aurora Replica when the read view pointer in the purge system is NULL.

Aurora MySQL database engine updates: 2016-10-26 (version 1.8.1) (Deprecated)

Version: 1.8.1

Improvements

- Fixed an issue where bulk inserts that use triggers that invoke AWS Lambda procedures fail.
- Fixed an issue where catalog migration fails when autocommit is off globally.
- Resolved a connection failure to Aurora when using SSL and improved Diffie-Hellman group to deal with LogJam attacks.

Integration of MySQL bug fixes

- OpenSSL changed the Diffie-Hellman key length parameters due to the LogJam issue. (Bug #18367167)

Aurora MySQL database engine updates: 2016-10-18 (version 1.8) (Deprecated)

Version: 1.8

New features

- **AWS Lambda integration** – You can now asynchronously invoke an AWS Lambda function from an Aurora DB cluster using the `mysql.lambda_async` procedure. For more information, see Invoking a Lambda function from an Amazon Aurora MySQL DB cluster in the *Amazon Aurora User Guide*.

- **Load data from Amazon S3** – You can now load text or XML files from an Amazon S3 bucket into your Aurora DB cluster using the `LOAD DATA FROM S3` or `LOAD XML FROM S3` commands. For more information, see Loading data into an Amazon Aurora MySQL DB cluster from text files in an Amazon S3 bucket in the *Amazon Aurora User Guide*.
- **Catalog migration** – Aurora now persists catalog metadata in the cluster volume to support versioning. This enables seamless catalog migration across versions and restores.
- **Cluster-level maintenance and patching** – Aurora now manages maintenance updates for an entire DB cluster. For more information, see Maintaining an Amazon Aurora DB cluster in the *Amazon Aurora User Guide*.

Improvements

- Fixed an issue where an Aurora Replica crashes when not granting a metadata lock to an inflight DDL table.
- Allowed Aurora Replicas to modify non-InnoDB tables to facilitate rotation of the slow and general log CSV files where `log_output=TABLE`.
- Fixed a lag when updating statistics from the primary instance to an Aurora Replica. Without this fix, the statistics of the Aurora Replica can get out of sync with the statistics of the primary instance and result in a different (and possibly under-performing) query plan on an Aurora Replica.
- Fixed a race condition that ensures that an Aurora Replica does not acquire locks.
- Fixed a rare scenario where an Aurora Replica that registers or de-registers with the primary instance could fail.
- Fixed a race condition that could lead to a deadlock on `db.r3.large` instances when opening or closing a volume.
- Fixed an out-of-memory issue that can occur due to a combination of a large write workload and failures in the Aurora Distributed Storage service.
- Fixed an issue with high CPU consumption because of the purge thread spinning in the presence of a long-running transaction.
- Fixed an issue when running information schema queries to get information about locks under heavy load.
- Fixed an issue with a diagnostics process that could in rare cases cause Aurora writes to storage nodes to stall and restart/fail-over.

- Fixed a condition where a successfully created table might be deleted during crash recovery if the crash occurred while a `CREATE TABLE [if not exists]` statement was being handled.
- Fixed a case where the log rotation procedure is broken when the general log and slow log are not stored on disk using catalog mitigation.
- Fixed a crash when a user creates a temporary table within a user defined function, and then uses the user defined function in the select list of the query.
- Fixed a crash that occurred when replaying GTID events. GTID is not supported by Aurora MySQL.

Integration of MySQL bug fixes:

- When dropping all indexes on a column with multiple indexes, InnoDB failed to block a DROP INDEX operation when a foreign key constraint requires an index. (Bug #16896810)
- Solve add foreign key constraint crash. (Bug #16413976)
- Fixed a crash when fetching a cursor in a stored procedure, and analyzing or flushing the table at the same time. (Bug # 18158639)
- Fixed an auto-increment bug when a user alters a table to change the AUTO_INCREMENT value to less than the maximum auto-increment column value. (Bug # 16310273)

Aurora MySQL database engine updates: 2016-09-20 (version 1.7.1) (Deprecated)

Version: 1.7.1

Improvements

- Fixes an issue where an Aurora Replica crashes if the InnoDB full-text search cache is full.
- Fixes an issue where the database engine crashes if a worker thread in the thread pool waits for itself.
- Fixes an issue where an Aurora Replica crashes if a metadata lock on a table causes a deadlock.
- Fixes an issue where the database engine crashes due to a race condition between two worker threads in the thread pool.
- Fixes an issue where an unnecessary failover occurs under heavy load if the monitoring agent doesn't detect the advancement of write operations to the distributed storage subsystem.

Aurora MySQL database engine updates: 2016-08-30 (version 1.7.0) (Deprecated)

Version: 1.7.0

New features

- **NUMA aware scheduler** – The task scheduler for the Aurora MySQL engine is now Non-Uniform Memory Access (NUMA) aware. This minimizes cross-CPU socket contention resulting in improved performance throughput for the `db.r3.8xlarge` DB instance class.
- **Parallel read-ahead operates asynchronously in the background** – Parallel read-ahead has been revised to improve performance by using a dedicated thread to reduce thread contention.
- **Improved index build (lab mode)** – The implementation for building secondary indexes now operates by building the index in a bottom-up fashion, which eliminates unnecessary page splits. This can reduce the time needed to create an index or rebuild a table. This feature is disabled by default and can be activated by enabling Aurora lab mode. For information, see Amazon Aurora MySQL lab mode in the *Amazon Aurora User Guide*.

Improvements

- Fixed an issue where establishing a connection was taking a long time if there was a surge in the number of connections requested for an instance.
- Fixed an issue where a crash occurred if ALTER TABLE was run on a partitioned table that did not use InnoDB.
- Fixed an issue where heavy write workload can cause a failover.
- Fixed an erroneous assertion that caused a failure if RENAME TABLE was run on a partitioned table.
- Improved stability when rolling back a transaction during insert-heavy workload.
- Fixed an issue where full-text search indexes were not viable on an Aurora Replica.

Integration of MySQL bug fixes

- Improve scalability by partitioning LOCK_grant lock. (Port WL #8355)
- Opening cursor on SELECT in stored procedure causes segfault. (Port Bug #16499751)

- MySQL gives the wrong result with some special usage. (Bug #11751794)
- Crash in GET_SEL_ARG_FOR_KEYPART – caused by patch for bug #11751794. (Bug #16208709)
- Wrong results for a simple query with GROUP BY. (Bug #17909656)
- Extra rows on semijoin query with range predicates. (Bug #16221623)
- Adding an ORDER BY clause following an IN subquery could cause duplicate rows to be returned. (Bug #16308085)
- Crash with explain for a query with loose scan for GROUP BY, MyISAM. (Bug #16222245)
- Loose index scan with quoted int predicate returns random data. (Bug #16394084)
- If the optimizer was using a loose index scan, the server could exit while attempting to create a temporary table. (Bug #16436567)
- COUNT(DISTINCT) should not count NULL values, but they were counted when the optimizer used loose index scan. (Bug #17222452)
- If a query had both MIN()/MAX() and aggregate_function(DISTINCT) (for example, SUM(DISTINCT)) and was executed using loose index scan, the result values of MIN()/MAX() were set improperly. (Bug #17217128)

Aurora MySQL database engine updates: 2016-06-01 (version 1.6.5) (Deprecated)

Version: 1.6.5

New features

- **Efficient storage of Binary Logs** – Efficient storage of binary logs is now enabled by default for all Aurora MySQL DB clusters, and is not configurable. Efficient storage of binary logs was introduced in the April 2016 update. For more information, see Aurora MySQL database engine updates: 2016-04-06 (version 1.6) (Deprecated).

Improvements

- Improved stability for Aurora Replicas when the primary instance is encountering a heavy workload.
- Improved stability for Aurora Replicas when running queries on partitioned tables and tables with special characters in the table name.

- Fixed connection issues when using secure connections.

Integration of MySQL bug fixes

- SLAVE CAN'T CONTINUE REPLICATION AFTER MASTER'S CRASH RECOVERY (Port Bug #17632285)

Aurora MySQL database engine updates: 2016-04-06 (version 1.6) (Deprecated)

Version: 1.6

This update includes the following improvements:

New features

- **Parallel read-ahead** – Parallel read-ahead is now enabled by default for all Aurora MySQL DB clusters, and is not configurable. Parallel read-ahead was introduced in the December 2015 update. For more information, see Aurora MySQL database engine updates: 2015-12-03 (version 1.4) (Deprecated).

  In addition to enabling parallel read-ahead by default, this release includes the following improvements to parallel read-ahead:

  - Improved logic so that parallel read-ahead is less aggressive, which is beneficial when your DB cluster encounters many parallel workloads.
  - Improved stability on smaller tables.
- **Efficient storage of Binary Logs (lab mode)** – MySQL binary log files are now stored more efficiently in Aurora MySQL. The new storage implementation enables binary log files to be deleted much earlier and improves system performance for an instance in an Aurora MySQL DB cluster that is a binary log replication master.

  To enable efficient storage of binary logs, set the `aurora_lab_mode` parameter to `1` in the parameter group for your primary instance or Aurora Replica. The `aurora_lab_mode` parameter is an instance-level parameter that is in the `default.aurora5.6` parameter group by default. For information on modifying a DB parameter group, see Modifying parameters in a DB

parameter group in the *Amazon Aurora User Guide*. For information on parameter groups and Aurora MySQL, see Aurora MySQL configuration parameters in the *Amazon Aurora User Guide*.

Only turn on efficient storage of binary logs for instances in an Aurora MySQL DB cluster that are MySQL binary log replication master instances.

- **AURORA_VERSION system variable** – You can now get the Aurora version of your Aurora MySQL DB cluster by querying for the `AURORA_VERSION` system variable.

  To get the Aurora version, use one of the following queries:

  ```
  select AURORA_VERSION();
  select @@aurora_version;
  show variables like '%version';
  ```

  You can also see the Aurora version in the AWS Management Console when you modify a DB cluster, or by calling the describe-db-engine-versions AWS CLI command or the DescribeDBEngineVersions API operation.

- **Lock manager memory usage metric** – Information about lock manager memory usage is now available as a metric.

  To get the lock manager memory usage metric, use one of the following queries:

  ```
  show global status where variable_name in ('aurora_lockmgr_memory_used');
  select * from INFORMATION_SCHEMA.GLOBAL_STATUS where variable_name in
   ('aurora_lockmgr_memory_used');
  ```

Improvements

- Improved stability during binlog and XA transaction recovery.
- Fixed a memory issue resulting from a large number of connections.
- Improved accuracy in the following metrics: `Read Throughput`, `Read IOPS`, `Read Latency`, `Write Throughput`, `Write IOPS`, `Write Latency`, and `Disk Queue Depth`.
- Fixed a stability issue causing slow startup for large instances after a crash.
- Improved concurrency in the data dictionary regarding synchronization mechanisms and cache eviction.

- Stability and performance improvements for Aurora Replicas:
  - Fixed a stability issue for Aurora Replicas during heavy or burst write workloads for the primary instance.
  - Improved replica lag for db.r3.4xlarge and db.r3.8xlarge instances.
  - Improved performance by reducing contention between application of log records and concurrent reads on an Aurora Replica.
  - Fixed an issue for refreshing statistics on Aurora Replicas for newly created or updated statistics.
  - Improved stability for Aurora Replicas when there are many transactions on the primary instance and concurrent reads on the Aurora Replicas across the same data.
  - Improved stability for Aurora Replicas when running `UPDATE` and `DELETE` statements with `JOIN` statements.
  - Improved stability for Aurora Replicas when running `INSERT ... SELECT` statements.

Integration of MySQL bug fixes

- BACKPORT Bug #18694052 FIX FOR ASSERTION `!M_ORDERED_REC_BUFFER' FAILED TO 5.6 (Port Bug #18305270)
- SEGV IN MEMCPY(), HA_PARTITION::POSITION (Port Bug # 18383840)
- WRONG RESULTS WITH PARTITIONING,INDEX_MERGE AND NO PK (Port Bug # 18167648)
- FLUSH TABLES FOR EXPORT: ASSERTION IN HA_PARTITION::EXTRA (Port Bug # 16943907)
- SERVER CRASH IN VIRTUAL HA_ROWS HANDLER::MULTI_RANGE_READ_INFO_CONST (Port Bug # 16164031)
- RANGE OPTIMIZER CRASHES IN SEL_ARG::RB_INSERT() (Port Bug # 16241773)

Aurora MySQL database engine updates: 2016-01-11 (version 1.5) (Deprecated)

Version: 1.5

This update includes the following improvements:

Improvements

- Fixed a 10 second pause of write operations for idle instances during Aurora storage deployments.
- Logical read-ahead now works when `innodb_file_per_table` is set to No. For more information on logical read-ahead, see Aurora MySQL database engine updates: 2015-12-03 (version 1.4) (Deprecated).
- Fixed issues with Aurora Replicas reconnecting with the primary instance. This improvement also fixes an issue when you specify a large value for the `quantity` parameter when testing Aurora Replica failures using fault-injection queries. For more information, see Testing an Aurora replica failure in the *Amazon Aurora User Guide*.
- Improved monitoring of Aurora Replicas falling behind and restarting.
- Fixed an issue that caused an Aurora Replica to lag, become deregistered, and then restart.
- Fixed an issue when you run the `show innodb status` command during a deadlock.
- Fixed an issue with failovers for larger instances during high write throughput.

Integration of MySQL bug fixes

- Addressed incomplete fix in MySQL full text search affecting tables where the database name begins with a digit. (Port Bug #17607956)

Aurora MySQL database engine updates: 2015-12-03 (version 1.4) (Deprecated)

Version: 1.4

This update includes the following improvements:

New features

- **Fast Insert** – Accelerates parallel inserts sorted by primary key. For more information, see Amazon Aurora MySQL performance enhancements in the *Amazon Aurora User Guide*.
- **Large dataset read performance** – Aurora MySQL automatically detects an IO heavy workload and launches more threads in order to boost the performance of the DB cluster. The Aurora

scheduler looks into IO activity and decides to dynamically adjust the optimal number of threads in the system, quickly adjusting between IO heavy and CPU heavy workloads with low overhead.

- **Parallel read-ahead** – Improves the performance of B-Tree scans that are too large for the memory available on your primary instance or Aurora Replica (including range queries). Parallel read-ahead automatically detects page read patterns and pre-fetches pages into the buffer cache in advance of when they are needed. Parallel read-ahead works multiple tables at the same time within the same transaction.

Improvements:

- Fixed brief Aurora database availability issues during Aurora storage deployments.
- Correctly enforce the `max_connection` limit.
- Improve binlog purging where Aurora is the binlog master and the database is restarting after a heavy data load.
- Fixed memory management issues with the table cache.
- Add support for huge pages in shared memory buffer cache for faster recovery.
- Fixed an issue with thread-local storage not being initialized.
- Allow 16K connections by default.
- Dynamic thread pool for IO heavy workloads.
- Fixed an issue with properly invalidating views involving UNION in the query cache.
- Fixed a stability issue with the dictionary stats thread.
- Fixed a memory leak in the dictionary subsystem related to cache eviction.
- Fixed high read latency issue on Aurora Replicas when there is very low write load on the master.
- Fixed stability issues on Aurora Replicas when performing operations on DDL partitioned tables such as ALTER TABLE ... REORGANIZE PARTITION on the master.
- Fixed stability issues on Aurora Replicas during volume growth.
- Fixed performance issue for scans on non-clustered indexes in Aurora Replicas.
- Fix stability issue that makes Aurora Replicas lag and eventually get deregistered and re-started.

Integration of MySQL bug fixes

- SEGV in FTSPARSE(). (Bug #16446108)

- InnoDB data dictionary is not updated while renaming the column. (Bug #19465984)
- FTS crash after renaming table to different database. (Bug #16834860)
- Failed preparing of trigger on truncated tables cause error 1054. (Bug #18596756)
- Metadata changes might cause problems with trigger execution. (Bug #18684393)
- Materialization is not chosen for long UTF8 VARCHAR field. (Bug #17566396)
- Poor execution plan when ORDER BY with limit X. (Bug #16697792)
- Backport bug #11765744 TO 5.1, 5.5 AND 5.6. (Bug #17083851)
- Mutex issue in SQL/SQL_SHOW.CC resulting in SIG6. Source likely FILL_VARIABLES. (Bug #20788853)
- Backport bug #18008907 to 5.5+ versions. (Bug #18903155)
- Adapt fix for a stack overflow error in MySQL 5.7. (Bug #19678930)

Aurora MySQL database engine updates: 2015-10-16 (versions 1.2, 1.3) (Deprecated)

Versions: 1.2, 1.3

This update includes the following improvements:

Fixes

- Resolved out-of-memory issue in the new lock manager with long-running transactions
- Resolved security vulnerability when replicating with non-RDS for MySQL databases
- Updated to ensure that quorum writes retry correctly with storage failures
- Updated to report replica lag more accurately
- Improved performance by reducing contention when many concurrent transactions are trying to modify the same row
- Resolved query cache invalidation for views that are created by joining two tables
- Disabled query cache for transactions with `UNCOMMITTED_READ` isolation

Improvements

- Better performance for slow catalog queries on warm caches

- Improved concurrency in dictionary statistics
- Better stability for the new query cache resource manager, extent management, files stored in Amazon Aurora smart storage, and batch writes of log records

Integration of MySQL bug fixes

- Killing a query inside innodb causes it to eventually crash with an assertion. (Bug #1608883)
- For failure to create a new thread for the event scheduler, event execution, or new connection, no message was written to the error log. (Bug #16865959)
- If one connection changed its default database and simultaneously another connection executed SHOW PROCESSLIST, the second connection could access invalid memory when attempting to display the first connection's default database memory. (Bug #11765252)
- PURGE BINARY LOGS by design does not remove binary log files that are in use or active, but did not provide any notice when this occurred. (Bug #13727933)
- For some statements, memory leaks could result when the optimizer removed unneeded subquery clauses. (Bug #15875919)
- During shutdown, the server could attempt to lock an uninitialized mutex. (Bug #16016493)
- A prepared statement that used GROUP_CONCAT() and an ORDER BY clause that named multiple columns could cause the server to exit. (Bug #16075310)
- Performance Schema instrumentation was missing for replica worker threads. (Bug #16083949)
- `STOP SLAVE` could cause a deadlock when issued concurrently with a statement such as SHOW STATUS that retrieved the values for one or more of the status variables `Slave_retried_transactions`, `Slave_heartbeat_period`, `Slave_received_heartbeats`, `Slave_last_heartbeat`, or `Slave_running`. (Bug #16088188)
- A full-text query using Boolean mode could return zero results in some cases where the search term was a quoted phrase. (Bug #16206253)
- The optimizer's attempt to remove redundant subquery clauses raised an assertion when executing a prepared statement with a subquery in the ON clause of a join in a subquery. (Bug #16318585)
- GROUP_CONCAT unstable, crash in ITEM_SUM::CLEAN_UP_AFTER_REMOVAL. (Bug #16347450)
- Attempting to replace the default InnoDB full-text search (FTS) stopword list by creating an InnoDB table with the same structure as

INFORMATION_SCHEMA.INNODB_FT_DEFAULT_STOPWORD would result in an error. (Bug #16373868)

- After the client thread on a worker performed a FLUSH TABLES WITH READ LOCK and was followed by some updates on the master, the worker hung when executing `SHOW SLAVE STATUS`. (Bug #16387720)
- When parsing a delimited search string such as "abc-def" in a full-text search, InnoDB now uses the same word delimiters as MyISAM. (Bug #16419661)
- Crash in FTS_AST_TERM_SET_WILDCARD. (Bug #16429306)
- SEGFAULT in FTS_AST_VISIT() for FTS RQG test. (Bug # 16435855)
- For debug builds, when the optimizer removed an Item_ref pointing to a subquery, it caused a server exit. (Bug #16509874)
- Full-text search on InnoDB tables failed on searches for literal phrases combined with + or - operators. (Bug #16516193)
- `START SLAVE` failed when the server was started with the options --master-info-repository=TABLE relay-log-info-repository=TABLE and with autocommit set to 0, together with `--skip-slave-start`. (Bug #16533802)
- Very large InnoDB full-text search (FTS) results could consume an excessive amount of memory. (Bug #16625973)
- In debug builds, an assertion could occur in OPT_CHECK_ORDER_BY when using binary directly in a search string, as binary might include NULL bytes and other non-meaningful characters. (Bug #16766016)
- For some statements, memory leaks could result when the optimizer removed unneeded subquery clauses. (Bug #16807641)
- It was possible to cause a deadlock after issuing FLUSH TABLES WITH READ LOCK by issuing `STOP SLAVE` in a new connection to the worker, then issuing `SHOW SLAVE STATUS` using the original connection. (Bug #16856735)
- GROUP_CONCAT() with an invalid separator could cause a server exit. (Bug #16870783)
- The server did excessive locking on the LOCK_active_mi and active_mi->rli->data_lock mutexes for any SHOW STATUS LIKE 'pattern' statement, even when the pattern did not match status variables that use those mutexes (`Slave_heartbeat_period`, `Slave_last_heartbeat`, `Slave_received_heartbeats`, `Slave_retried_transactions`, `Slave_running`). (Bug #16904035)
- A full-text search using the IN BOOLEAN MODE modifier would result in an assertion failure. (Bug #16927092)

- Full-text search on InnoDB tables failed on searches that used the + boolean operator. (Bug #17280122)
- 4-way deadlock: zombies, purging binlogs, show processlist, show binlogs. (Bug #17283409)
- When an SQL thread which was waiting for a commit lock was killed and restarted it caused a transaction to be skipped on worker. (Bug #17450876)
- An InnoDB full-text search failure would occur due to an "unended" token. The string and string length should be passed for string comparison. (Bug #17659310)
- Large numbers of partitioned InnoDB tables could consume much more memory when used in MySQL 5.6 or 5.7 than the memory used by the same tables used in previous releases of the MySQL Server. (Bug #17780517)
- For full-text queries, a failure to check that num_token is less than max_proximity_item could result in an assertion. (Bug #18233051)
- Certain queries for the INFORMATION_SCHEMA TABLES and COLUMNS tables could lead to excessive memory use when there were large numbers of empty InnoDB tables. (Bug #18592390)
- When committing a transaction, a flag is now used to check whether a thread has been created, rather than checking the thread itself, which uses more resources, particularly when running the server with master_info_repository=TABLE. (Bug #18684222)
- If a client thread on a worker executed FLUSH TABLES WITH READ LOCK while the master executed a DML, executing `SHOW SLAVE STATUS` in the same client became blocked, causing a deadlock. (Bug #19843808)
- Ordering by a GROUP_CONCAT() result could cause a server exit. (Bug #19880368)

Aurora MySQL database engine updates: 2015-08-24 (version 1.1) (Deprecated)

Version: 1.1

This update includes the following improvements:

- Replication stability improvements when replicating with a MySQL database (binlog replication). For information on Aurora MySQL replication with MySQL, see Replication with Amazon Aurora in the *Amazon Aurora User Guide*.

- A 1 gigabyte (GB) limit on the size of the relay logs accumulated for an Aurora MySQL DB cluster that is a replication worker. This improves the file management for the Aurora DB clusters.
- Stability improvements in the areas of read ahead, recursive foreign-key relationships, and Aurora replication.
- Integration of MySQL bug fixes.
  - InnoDB databases with names beginning with a digit cause a full-text search (FTS) parser error. (Bug #17607956)
  - InnoDB full-text searches fail in databases whose names began with a digit. (Bug #17161372)
  - For InnoDB databases on Windows, the full-text search (FTS) object ID is not in the expected hexadecimal format. (Bug #16559254)
  - A code regression introduced in MySQL 5.6 negatively impacted DROP TABLE and ALTER TABLE performance. This could cause a performance drop between MySQL Server 5.5.x and 5.6.x. (Bug #16864741)
- Simplified logging to reduce the size of log files and the amount of storage that they require.

MySQL bugs fixed by Aurora MySQL database engine updates

The following sections identify MySQL bugs that have been fixed by Aurora MySQL database engine updates.

Topics

MySQL bugs fixed by Aurora MySQL 3.x database engine updates

MySQL 8.0-compatible version Aurora contains all MySQL bug fixes through their corresponding MySQL Compatibility version. The following table identifies additional MySQL bugs that have been fixed by Aurora MySQL database engine updates, and which update they were fixed in.

Database engine update	MySQL compatible version	Version	MySQL bugs fixed
Aurora MySQL database engine updates 2024-06-04 (version 3.07.0, compatible with	8.0.36	3.07.0	• Fixed an issue where cache line value can be calculated incorrectly, causing a failure during database restart on Graviton-based instances. (Community Bug Fix #35479763) • Fixed an issue where some instances of subqueries within stored routines were not always handled correctly. (Community Bug Fix #35377192)

Database engine update	MySQL compatible version	Version	MySQL bugs fixed
MySQL 8.0.36)			• Fixed an issue that can cause higher CPU usage due to background TLS certificate rotation (Community Bug Fix #34284186). • Fixed an issue where InnoDB allowed the addition of `INSTANT` columns to tables in the MySQL system schema in Aurora MySQL versions lower than 3.05, which could lead to the server unexpectedly closing (database instance restarting) after upgrading to Aurora MySQL version 3.05.0. (Community Bug Fix #35625510).

Database engine update	MySQL compatible version	Version	MySQL bugs fixed
Aurora MySQL database engine updates 2024-03-07 (version 3.06.0, compatible with MySQL 8.0.34)	8.0.34	3.06.0	• Fixed an issue where cache line value can be incorrectly calculated, causing a failure during database restart on a Graviton instance. (Community Bug Fix #35479763) • Fixed an issue where some instances of subqueries within stored routines were not always handled correctly. (Community Bug Fix #35377192) • Fixed an issue that can cause higher CPU usage due to background TLS certificate rotation. (Community Bug Fix #34284186) • Fixed an issue where InnoDB allowed the addition of `INSTANT` columns to tables in the MySQL system schema in Aurora MySQL versions lower than 3.05, which could lead to the server unexpectedly closing (database instance restarting) after upgrading to Aurora MySQL version 3.05.0. (Community Bug Fix #35625510)

Database engine update	MySQL compatible version	Version	MySQL bugs fixed
Aurora MySQL database engine updates 2024-01-31 (version 3.05.2, compatible with MySQL 8.0.32) Default	8.0.32	3.05.2	• Repeated running of a stored routine, having as a subquery a SELECT statement containing multiple AND, OR, or XOR conditions, led to excessive consumption and possibly eventual exhaustion of virtual memory. (Community Bug Fix #33852530)
Aurora MySQL database engine updates 2023-11-21 (version 3.05.1, compatible with MySQL 8.0.32)	8.0.32	3.05.1	• Fixed an issue in InnoDB when, if a MySQL table in a system schema had an `INSTANT ADD` column added between Aurora MySQL versions 3.01 through Aurora MySQL versions 3.04, and after Aurora MySQL was upgraded to version 3.05.0, DMLs on these tables would result in the server unexpectedly closing. (Community Bug Fix #35625510)

Database engine update	MySQL compatible version	Version	MySQL bugs fixed
Aurora MySQL database engine updates 2023-10-25 (version 3.05.0, compatible with MySQL 8.0.32)	8.0.32	3.05.0	• Fixed an issue which can cause higher CPU utilization due to background TLS certificate rotation (Community Bug Fix #34284186)
Aurora MySQL database engine updates 2024-03-15 (version 3.04.2, compatible with MySQL 8.0.28)	8.0.28	3.04.2	• Fixed an issue where cache line value can be calculated incorrectly, causing a failure during database restart on Graviton-based instances. (Community Bug Fix #35479763) • Repeated running of a stored routine, having as a subquery a SELECT statement containing multiple AND, OR, or XOR conditions, led to excessive consumption and possibly eventual exhaustion of virtual memory. (Community Bug Fix #33852530)

Database engine update	MySQL compatible version	Version	MySQL bugs fixed
Aurora MySQL database engine updates 2023-11-13 (version 3.04.1, compatible with MySQL 8.0.28)	8.0.28	3.04.1	• Fixed an issue which can cause higher CPU utilization due to background TLS certificate rotation (Community Bug Fix #34284186)
Aurora MySQL database engine updates 2023-07-31 (version 3.04.0, compatible with MySQL 8.0.28)	8.0.28	3.04.0	• Fixed an issue where a buffer block containing intrinsic temporary table page was relocated during page traversal , causing an assertion failure (Bug# 33715694) • InnoDB: Prevent online DDL operations from accessing out-of-bounds memory (Bug# 34750489, Bug# 108925) • Fixed an issue which can sometimes produce incorrect query results while processin g complex SQL statements consisting of multiple nested Common Table Expressions (CTEs) (Bug# 34572040, Bug# 34634469, Bug# 33856374)

Database engine update	MySQL compatible version	Version	MySQL bugs fixed
Aurora MySQL database engine updates 2023-12-08 (version 3.03.3, compatible with MySQL 8.0.26)	8.0.26	3.03.3	• Fixed an issue which can cause higher CPU utilization due to background TLS certificate rotation (Community Bug Fix #34284186)
Aurora MySQL database engine updates 2023-08-29 (version 3.03.2, compatible with MySQL 8.0.26)	8.0.26	3.03.2	• Fixed an issue which can sometimes produce incorrect query results while processing complex SQL statements consisting of multiple nested Common Table Expressions (CTEs) (Bug #34572040, Bug #34634469, Bug #33856374) • InnoDB: A race condition between threads attempting to deinitialize and initialize statistics for the same table raised an assertion failure (Bug #33135425) • InnoDB: Prevent online DDL operations from accessing out-of-bounds memory (Bug #34750489, Bug #108925)

Database engine update	MySQL compatible version	Version	MySQL bugs fixed
Aurora MySQL database engine updates 2023-05-11 (version 3.03.1, compatible with MySQL 8.0.26)	8.0.26	3.03.1	• Fixed an issue where a buffer block containing intrinsic temporary table page was relocated during page traversal , causing an assertion failure (Bug #33715694)

Database engine update	MySQL compatible version	Version	MySQL bugs fixed

Aurora MySQL database engine updates 2023-03-01 (version 3.03.0, compatible with MySQL 8.0.26) Upgrades to this version aren't supported.	8.0.26	3.03.0	• Fixed an issue where sorts of some column types, including JSON and TEXT, sometimes exhausted the sort buffer if its size was not at least 15 times that of the largest row in the sort. Now the sort buffer need only be 15 times as large as the largest sort key. (Bug #103325, Bug #105532, Bug #32738705, Bug #33501541)
			• Fixed an issue wherein InnoDB did not always handle some legal names for table partitions correctly. (Bug #32208630)
			• Fixed an issue which, in certain conditions, may return incorrect results due to an inaccurate calculation of the nullability property when executing a query with an OR condition. (Bug #34060289)
			• Fixed an issue which, in certain conditions, may return incorrect results when the following two conditions are met:
			  • a derived table is merged into the outer query block.
			  • the query includes a left join and an IN subquery.
			(Bug #34060289)

Database engine update	MySQL compatible version	Version	MySQL bugs fixed
			• Fixed an issue where incorrect AUTO_INCREMENT values were generated when the maximum integer column value was exceeded. The error was due to the maximum column value not being considered. The previous valid AUTO_INCREMENT value should have been returned in this case, causing a duplicate key error. (Bug #87926, Bug #26906787) • Fixed an issue where it was not possible to revoke the DROP privilege on the Performance Schema. (Bug #33578113) • Fixed an issue where a stored procedure containing an IF statement using EXISTS, which acted on one or more tables that were deleted and recreated between executions, did not execute correctly for subsequen t invocations following the first one. (Bug #32855634). • Fixed an issue where a query that references a view in a subquery and an outer query block can cause an unexpected restart (Bug#32324234)

Database engine update	MySQL compatible version	Version	MySQL bugs fixed

Aurora MySQL database engine updates 2022-11-18 (version 3.02.2, compatible with MySQL 8.0.23) The end of standard support is January 15, 2024.	8.0.23	3.02.2	• Fixed an issue which, in certain conditions, may return incorrect results due to an inaccurate calculation of the nullability property when executing a query with an OR condition. (Bug #34060289)
			• Fixed an issue which, in certain conditions, may return incorrect results when the following two conditions are met:
			  • A derived table is merged into the outer query block.
			  • The query includes a left join and an IN subquery. (Bug #34060289)
			• Fixed an issue where it wasn't possible to revoke the DROP privilege on the Performance Schema. (Bug #33578113)
			• Fixed an issue where a stored procedure containing an IF statement using EXISTS, which acted on one or more tables that were deleted and recreated between executions, did not execute correctly for subsequent invocations following the first one. (MySQL Bug #32855634).
			• Incorrect AUTO_INCREMENT values were generated when

Database engine update	MySQL compatible version	Version	MySQL bugs fixed
			the maximum integer column value was exceeded. The error was due to the maximum column value not being considered. The previous valid AUTO_INCREMENT value should have been returned in this case, causing a duplicate key error. (Bug #87926, Bug #26906787) • Fixed an issue which can lead to a failure while upgrading an Aurora MySQL version 1 (Compatible with MySQL 5.6) database cluster containin g user-created table with certain table IDs. Assignment of these table IDs may result in conflicting data dictionar y table IDs while upgrading from Aurora MySQL version 2 (Compatible with MySQL 5.7) to Aurora MySQL version 3 (Compatible with MySQL 8.0). (Bug #33919635)

Database engine update	MySQL compatible version	Version	MySQL bugs fixed
Aurora MySQL database engine updates 2022-04-20 (version 3.02.0, compatible with MySQL 8.0.23) The end of standard support is January 15, 2024. Upgrades to this version aren't supported.	8.0.23	3.02.0	Fixed the improper handling of temporary tables used for cursors within stored procedures that could result in unexpected server behavior. (Bug #32416811)

Database engine update	MySQL compatible version	Version	MySQL bugs fixed
Aurora MySQL database engine updates 2022-04-15 (version 3.01.1, compatible with MySQL 8.0.23) The end of standard support is January 15, 2024. Upgrades to this version aren't supported.	8.0.23	3.01.1	Fixed the improper handling of temporary tables used for cursors within stored procedures that could result in unexpected server behavior. (Bug #32416811)

MySQL bugs fixed by Aurora MySQL 2.x database engine updates

MySQL 5.7-compatible version Aurora contains all MySQL bug fixes through MySQL 5.7.40. The following table identifies additional MySQL bugs that have been fixed by Aurora MySQL database engine updates, and which update they were fixed in.

Database engine update	Version	MySQL bugs fixed
Aurora MySQL database engine updates 2023-12-28 (version 2.12.1, compatible with MySQL 5.7.40)	2.12.1	• Fixed an issue which can cause existing and new remote connections to stall when run concurrently with the `SHOW PROCESSLIST` statement (Community Bug #34857411) • Replication: Some binary log events were not always handled correctly (Bug #34617506) • Fixed processing of single character tokens by a Full-Text Search (FTS) parser plugin (Bug #35432973)
Aurora MySQL database engine updates 2023-07-25 (version 2.12.0, compatible with MySQL 5.7.40)	2.12.0	• Fixed an issue which can cause higher CPU utilization due to background TLS certificate rotation. (Community Bug Fix #34284186)
Aurora MySQL database engine updates 2023-10-17 (version 2.11.4, compatible with MySQL 5.7.12)	2.11.4	• Replication: Some binary log events were not always handled correctly. (Bug #34617506) • Fixed an issue which can cause higher CPU utilization due to background TLS certificate rotation. (Community Bug Fix #34284186) • In prepared statements, some types of subqueries could cause a server exit. (Bug #33100586)
Aurora MySQL database	2.11.0	• Fixed an issue where the code for reading character set information from Performance Schema statement events

Database engine update	Version	MySQL bugs fixed
engine updates 2022-10-25 (version 2.11.0, compatible with MySQL 5.7.12) This version isn't available for new creations.		tables (for example, events_statements_current) did not prevent simultaneous writing to that character set informati on. As a result, the SQL query text character set could be invalid, which could result in a server exit. With this fix, an invalid character set causes SQL_TEXT column truncation and prevents server exits. (Bug #23540008) • InnoDB: Backport of fix for Community Bug #25189192, Bug #84038. Fixed an issue where after a RENAME TABLE operation that moved a table to a different schema, InnoDB failed to update INNODB_SYS_DATAFILES data dictionary table. This resulted in an error on restart indicating that it could not locate the tablespace data file. • InnoDB: Fixed an issue where the server dropped an internall y defined foreign key index when adding a new index and attempted to use a secondary index defined on a virtual generated column as the foreign key index, causing a server exit. InnoDB now permits a foreign key constraint to reference a secondary index defined on a virtual generated column. (Bug #23533396) • Fixed an issue where two sessions concurrently executing an INSERT ... ON DUPLICATE KEY UPDATE operation generated a deadlock. During partial rollback of a tuple, another session could update it. The fix for this bug reverts the fixes for Bug #11758237, Bug #17604730, and Bug #20040791. (Bug #25966845) • Fixed an issue where the EXECUTE and ALTER ROUTINE privileges weren't correctly granted to routine creators even with automatic_sp_privileges enabled. (Bug #27407480) • Backport of fix for Community Bug#24671968: Fixed an issue where a query could produce incorrect results if the WHERE clause contained a dependent subquery, the table had a secondary index on the columns in the select list

Database engine update	Version	MySQL bugs fixed
		followed by the columns in the subquery, and `GROUP BY` or `DISTINCT` permitted the query to use a Loose Index Scan. • Fixed an issue where replication breaks if a multi-table delete statement is issued against multiple tables with foreign keys. (Bug #80821) • Fixed an issue where in special cases certain slave errors are not ignored even with slave_skip_errors enabled. In cases when opening and locking a table failed or when field conversions failed on a server running row-based replicati on, the error is considered critical and the state of slave_ski p_errors is ignored. The fix ensures that with slave_ski p_errors) enabled, all errors reported during applying a transaction are correctly handled. (Bug #70640, Bug #17653275) • Fixed an issue where a `SET PASSWORD` statement was replicated from a MySQL 5.6 master to a MySQL 5.7 slave, or from a MySQL 5.7 master with the log_builtin_as_ide ntified_by_password system variable set to ON to a MySQL 5.7 slave, the password hash was itself also hashed before being stored on the slave. The issue has now been fixed and the replicated password hash is stored as originally passed to the slave. (Bug#24687073) • Fixed an issue where serialization of a JSON value consistin g of a large sub-document wrapped in many levels of JSON arrays, objects, or both, sometimes required an excessive amount time to complete. (Bug #23031146) • Statements that cannot be parsed (due, for example, to syntax errors) are no longer written to the slow query log. (Bug #33732907)

Database engine update	Version	MySQL bugs fixed
Aurora MySQL database engine updates 2022-11-01 (version 2.10.3) (Deprecated)	2.10.3	• Fixed an issue where the code for reading character set information from Performance Schema statement events tables (for example, events_statements_current) did not prevent simultaneous writing to that character set informati on. As a result, the SQL query text character set could be invalid, which could result in a server exit. With this fix, an invalid character set causes SQL_TEXT column truncation and prevents server exits. (Bug #23540008) • Fixed an issue when an UPDATE required a temporary table having a primary key larger than 1024 bytes and that table was created using InnoDB, the server could exit. (Bug #25153670

Database engine update	Version	MySQL bugs fixed
Aurora MySQL database engine updates 2022-01-26 (version 2.10.2) (Deprecated)	2.10.2	• Fixed an issue in InnoDB where an error in code related to table statistics raised an assertion in the dict0stats.cc (http://dict0stats.cc/) source file. (Bug #24585978) • A secondary index over a virtual column became corrupted when the index was built online. For UPDATE (https://dev.mysql.com/doc/refman/5.7/en/update.html) statements, we fix this as follows: If the virtual column value of the index record is set to NULL, then we generate this value from the cluster index record. (Bug #30556595) • ASSERTION "!OTHER_LOCK" IN LOCK_REC_ADD_TO_QUEUE (Bug #29195848) • HANDLE_FATAL_SIGNAL (SIG=11) IN __STRCHR_SSE2 (Bug #28653104) • Fixed an issue which a query interruption during a lock wait can cause an error in InnoDB. (Bug #28068293) • Interleaved transactions could sometimes deadlock the replica applier when the transaction isolation level was set to REPEATABLE READ. (Bug #25040331) • Fixed an issue which can cause binlog replicas to stall due to lock wait timeout.(Bug #27189701)
Aurora MySQL database engine updates 2021-10-21 (version 2.10.1) (Deprecated)	2.10.1	CURRENT_TIMESTAMP PRODUCES ZEROS IN TRIGGER. (Bug #25209512)

Database engine update	Version	MySQL bugs fixed
Aurora MySQL database engine updates 2021-05-25 (version 2.10.0) (Deprecated)	2.10.0	• Interleaved transactions could sometimes deadlock the replica applier when the transaction isolation level was set to REPEATABLE READ. (Bug #25040331) • When a stored procedure contained a statement referring to a view which in turn referred to another view, the procedure could not be invoked successfully more than once. (Bug #87858, Bug #26864199) • For queries with many OR conditions, the optimizer now is more memory-efficient and less likely to exceed the memory limit imposed by the range_optimizer_max_mem_size system variable. In addition, the default value for that variable has been raised from 1,536,000 to 8,388,608. (Bug #79450, Bug #22283790) • *Replication:* In the `next_event()` function, which is called by a replica's SQL thread to read the next event from the relay log, the SQL thread did not release the `relaylog.log_lock` it acquired when it ran into an error (for example, due to a closed relay log), causing all other threads waiting to acquire a lock on the relay log to hang. With this fix, the lock is released before the SQL thread leaves the function under the situation. (Bug #21697821) • Fixing a memory corruption for `ALTER TABLE` with virtual column. (Bug #24961167; Bug #24960450) • *Replication:* Multithreaded replicas could not be configured with small queue sizes using slave_pending_jobs_size_max if they ever needed to process transactions larger than that size. Any packet larger than slave_pending_jobs_size_max was rejected with the error `ER_MTS_EVENT_BIGGER_PENDING_JOBS_SIZE_MAX`, even if the packet was smaller than the limit set by slave_max_allowed_packet. With this fix, slave_pending_jobs_size_max becomes a soft limit rather than a hard limit. If the size of a packet

Database engine update	Version	MySQL bugs fixed
		exceeds slave_pending_jobs_size_max but is less than slave_max_allowed_packet, the transaction is held until all the replica workers have empty queues, and then processed. All subsequent transactions are held until the large transacti on has been completed. The queue size for replica workers can therefore be limited while still allowing occasional larger transactions. (Bug #21280753, Bug #77406) • *Replication:* When using a multithreaded replica, applier errors displayed worker ID data that was inconsistent with data externalized in Performance Schema replication tables. (Bug #25231367) • *Replication:* On a GTID-based replication replica running with -gtid-mode=ON, -log-bin=OFF, and using -slave-skip-errors , when an error was encountered that should be ignored `Exec_Master_Log_Pos` was not being correctly updated, causing `Exec_Master_Log_Pos` to loose synchrony with `Read_master_log_pos` . If a `GTID_NEXT` was not specified, the replica would never update its GTID state when rolling back from a single statement transaction. The `Exec_Master_Log_Pos` would not be updated because even though the transaction was finished, its GTID state would show otherwise. The fix removes the restraint of updating the GTID state when a transaction is rolled back only if `GTID_NEXT` is specified. (Bug #22268777) • *Replication:* A partially failed statement was not correctly consuming an auto-generated or specified GTID when binary logging was disabled. The fix ensures that a partially failed DROP TABLE, a partially failed DROP USER, or a partially failed DROP VIEW consume respectively the relevant GTID and save it into `@@GLOBAL.GTID_EXECUTED` and `mysql.gtid_executed` table when binary logging is disabled. (Bug #21686749)

Database engine update	Version	MySQL bugs fixed
		• *Replication:* Replicas running MySQL 5.7 could not connect to a MySQL 5.5 source due to an error retrieving the server_uuid, which is not part of MySQL 5.5. This was caused by changes in the method of retrieving the `server_uuid` . (Bug #22748612) • Binlog replication: GTID transaction skipping mechanism was not working properly for XA transaction before this fix. Server has a mechanism to skip (silently) a GTID transacti on if it is already executed that particular transaction in the past. (BUG#25041920) • XA ROLLBACK statements that failed because an incorrect transaction ID was given, could be recorded in the binary log with the correct transaction ID, and could therefore be actioned by replication replicas. A check is now made for the error situation before binary logging takes place, and failed XA ROLLBACK statements are not logged. (Bug #26618925) • *Replication:* If a replica was set up using a CHANGE MASTER TO statement that did not specify the source log file name and source log position, then shut down before START SLAVE was issued, then restarted with the option -relay-log-recovery set, replication did not start. This happened because the receiver thread had not been started before relay log recovery was attempted, so no log rotation event was available in the relay log to provide the source log file name and source log position. In this situation, the replica now skips relay log recovery and logs a warning, then proceeds to start replication. (Bug #28996606, Bug #93397) • *Replication:* In row-based replication, a message that incorrectly displayed field lengths was returned when replicating from a table with a `utf8mb3` column to a table of the same definition where the column was defined with a `utf8mb4` character set. (Bug #25135304, Bug #83918)

Database engine update	Version	MySQL bugs fixed
		• *Replication:* When a RESET SLAVE statement was issued on a replication replica with GTIDs in use, the existing relay log files were purged, but the replacement new relay log file was generated before the set of received GTIDs for the channel had been cleared. The former GTID set was therefore written to the new relay log file as the `PREVIOUS_GTIDS` event, causing a fatal error in replication stating that the replica had more GTIDs than the source, even though the gtid_executed set for both servers was empty. Now, when `RESET SLAVE` is issued, the set of received GTIDs is cleared before the new relay log file is generated, so that this situation does not occur. (Bug #27411175) • *Replication:* With GTIDs in use for replication, transactions including statements that caused a parsing error (ER_PARSE_ERROR) could not be skipped manually by the recommended method of injecting an empty or replacement transaction with the same GTID. This action should result in the replica identifying the GTID as already used, and therefore skipping the unwanted transaction that shared its GTID. However, in the case of a parsing error, because the statement was parsed before the GTID was checked to see if it needed to be skipped, the replication applier thread stopped due to the parsing error, even though the intention was for the transaction to be skipped anyway. With this fix, the replication applier thread now ignores parsing errors if the transaction concerned needs to be skipped because the GTID was already used. Note that this behavior change does not apply in the case of workloads consisting of binary log output produced by `mysqlbinlog`. In that situation, there would be a risk that a transaction with a parsing error that immediately follows a skipped transaction would also be silently skipped, when it ought to raise an error. (Bug #27638268)

Database engine update	Version	MySQL bugs fixed

		• *Replication:* Enable the SQL thread to GTID skip a partial transaction. (Bug #25800025)
		• *Replication:* When a negative or fractional timeout parameter was supplied to `WAIT_UNTIL_SQL_THREAD_AFTER_GTIDS()`, the server behaved in unexpected ways. With this fix:
		  • A fractional timeout value is read as-is, with no round-off.
		  • A negative timeout value is rejected with an error if the server is on a strict SQL mode; if the server is not on a strict SQL mode, the value makes the function return `NULL` immediately without any waiting and then issue a warning. (Bug #24976304, Bug #83537)
		• *Replication*: If the `WAIT_FOR_EXECUTED_GTID_SET()` function was used with a timeout value including a fractional part (for example, 1.5), an error in the casting logic meant that the timeout was rounded down to the nearest whole second, and to zero for values less than 1 second (for example, 0.1). The casting logic has now been corrected so that the timeout value is applied as originally specified with no rounding. Thanks to Dirkjan Bussink for the contribution. (Bug #29324564, Bug #94247)
		• With GTIDs enabled, XA COMMIT on a disconnected XA transaction within a multiple-statement transaction raised an assertion. (Bug #22173903)
		• *Replication:* An assertion was raised in debug builds if an XA ROLLBACK statement was issued for an unknown transaction identifier when the gtid_next value had been set manually. The server now does not attempt to update the GTID state if an XA `ROLLBACK` statement fails with an error. (Bug #27928837, Bug #90640)
		• Fix wrong sorting order issue when multiple `CASE` functions are used in `ORDER BY` clause. (Bug#22810883)

Database engine update	Version	MySQL bugs fixed
		• Some queries that used ordering could access an uninitialized column during optimization and cause a server exit. (Bug #27389294)
Aurora MySQL database engine updates 2021-11-12 (version 2.09.3) (Deprecated)	2.09.3	• ASSERTION !M_PREBUILT->TRX->CHECK_FOREIGNS. (Bug #23533396) • Replication:* A locking issue in the WAIT_FOR_EXECUTED_GTID_SET() function could cause the server to hang in certain circumstances. The issue has now been corrected. (Bug #29550513)
Aurora MySQL database engine updates 2020-12-11 (version 2.09.1) (Deprecated)	2.09.1	• **Replication:** Interleaved transactions could sometimes deadlock the slave applier when the transaction isolation level was set to REPEATABLE READ. (Bug #25040331) • For a table having a TIMESTAMP or DATETIME column having a default of CURRENT_TIMESTAMP, the column could be initialized to `0000-00-00 00:00:00` if the table had a `BEFORE INSERT` trigger. (Bug #25209512, Bug #84077) • For an INSERT statement for which the VALUES list produced values for the second or later row using a subquery containing a join, the server could exit after failing to resolve the required privileges. (Bug #23762382)
Aurora MySQL database engine updates 2020-11-12 (version 2.08.3) (Deprecated)	2.08.3	• Bug #23762382 - INSERT VALUES QUERY WITH JOIN IN A SELECT CAUSES INCORRECT BEHAVIOR. • Bug #25209512 - CURRENT_TIMESTAMP PRODUCES ZEROS IN TRIGGER.

Database engine update	Version	MySQL bugs fixed
Aurora MySQL database engine updates 2020-06-02 (version 2.08.0) (Deprecated)	2.08.0	• Bug #25289359: A full-text cache lock taken when data is synchronized was not released if the full-text cache size exceeded the full-text cache size limit. • Bug #29138644: Manually changing the system time while the MySQL server was running caused page cleaner thread delays. • Bug #25222337: A NULL virtual column field name in a virtual index caused a server exit during a field name comparison that occurs while populating virtual columns affected by a foreign key constraint. • Bug #25053286: Executing a stored procedure containing a query that accessed a view could allocate memory that was not freed until the session ended. • Bug #25586773: Executing a stored procedure containing a statement that created a table from the contents of certain SELECT statements could result in a memory leak. • Bug #28834208: During log application, after an OPTIMIZE TABLE operation, InnoDB did not populate virtual columns before checking for virtual column index updates. • Bug #26666274: Infinite loop in performance schema buffer container due to 32-bit unsigned integer overflow.
Aurora MySQL database engine updates 2022-06-16 (version 2.07.8) (Deprecated)	2.07.8	When an UPDATE required a temporary table having a primary key larger than 1024 bytes and that table was created using InnoDB, the server could exit. (Bug #25153670)

Database engine update	Version	MySQL bugs fixed
Aurora MySQL database engine updates 2021-09-02 (version 2.07.6) (Deprecated)	2.07.6	• INSERTING 64K SIZE RECORDS TAKE TOO MUCH TIME. (Bug#23031146)
Aurora MySQL database engine updates 2021-03-04 (version 2.07.4) (Deprecated)	2.07.4	• Fixed an issue in the Full-text ngram parser when dealing with tokens containing ' ' (space), '%', or ','. Customers should rebuild their FTS indexes if using ngram parser. (Bug #25873310) • Fixed an issue that could cause engine restart during query execution with nested SQL views. (Bug #27214153, Bug #26864199)

Database engine update	Version	MySQL bugs fixed
Aurora MySQL database engine updates 2020-11-10 (version 2.07.3) (Deprecated)	2.07.3	• *InnoDB:* Concurrent XA transactions that ran successfully to the XA prepare stage on the master conflicted when replayed on the slave, resulting in a lock wait timeout in the applier thread. The conflict was due to the GAP lock range which differed when the transactions were replayed serially on the slave. To prevent this type of conflict, GAP locks taken by XA transactions in READ COMMITTED isolation level are now released (and no longer inherited) when XA transactions reach the prepare stage. (Bug #27189701, Bug #25866046) • *InnoDB:* A gap lock was taken unnecessarily during foreign key validation while using the READ COMMITTED isolation level. (Bug #25082593) • *Replication:* When using XA transactions, if a lock wait timeout or deadlock occurred for the applier (SQL) thread on a replication slave, the automatic retry did not work. The cause was that while the SQL thread would do a rollback, it would not roll the XA transaction back. This meant that when the transaction was retried, the first event was XA START which was invalid as the XA transaction was already in progress, leading to an XAER_RMFAIL error. (Bug #24764800) • *Replication:* Interleaved transactions could sometimes deadlock the slave applier when the transaction isolation level was set to REPEATABLE READ. (Bug #25040331) • *Replication:* The value returned by a SHOW SLAVE STATUS statement for the total combined size of all existing relay log files (Relay_Log_Space) could become much larger than the actual disk space used by the relay log files. The I/O thread did not lock the variable while it updated the value, so the SQL thread could automatically delete a relay log file and write a reduced value before the I/O thread finished updating the value. The I/O thread then wrote its original size calculation, ignoring the SQL thread's update and so

Database engine update	Version	MySQL bugs fixed
		adding back the space for the deleted file. The Relay_Log _Space value is now locked during updates to prevent concurrent updates and ensure an accurate calculation. (Bug #26997096, Bug #87832) • For an INSERT statement for which the VALUES list produced values for the second or later row using a subquery containin g a join, the server could exit after failing to resolve the required privileges. (Bug #23762382) • For a table having a TIMESTAMP or DATETIME column having a default of CURRENT_TIMESTAMP, the column could be initialized to `0000-00-00 00:00:00` if the table had a `BEFORE INSERT` trigger. (Bug #25209512, Bug #84077) • A server exit could result from simultaneous attempts by multiple threads to register and deregister metadata Performance Schema objects. (Bug #26502135) • Executing a stored procedure containing a statement that created a table from the contents of certain SELECT statements could result in a memory leak. (Bug #25586773) • Executing a stored procedure containing a query that accessed a view could allocate memory that was not freed until the session ended. (Bug #25053286) • Certain cases of subquery materialization could cause a server exit. These queries now produce an error suggesting that materialization be disabled. (Bug #26402045) • Queries with many left joins were slow if join buffering was used (for example, using the block nested loop algorithm). (Bug #18898433, Bug #72854) • The optimizer skipped the second column in a composite index when executing an inner join with a `LIKE` clause against the second column. (Bug #28086754)

Database engine update	Version	MySQL bugs fixed
Aurora MySQL database engine updates 2020-04-17 (version 2.07.2) (Deprecated)	2.07.2	• Bug #23104498: Fixed an issue in Performance Schema in reporting total memory used. (https://github.com/mysql/mysql-server/commit/20b6840df5452f47313c6f9a6ca075bfbc00a96b) • Bug #22551677: Fixed an issue in Performance Schema that could lead to the database engine crashing when attemptin g to take it offline. (https://github.com/mysql/mysql-server/commit/05e2386eccd32b6b444b900c9f8a87a1d8d531e9) • Bug #23550835, Bug #23298025, Bug #81464: Fixed an issue in Performance Schema that causes a database engine crash due to exceeding the capacity of an internal buffer. (https://github.com/mysql/mysql-server/commit/b4287f93857bf2f99b18fd06f555bbe5b12debfc, https://github.com/mysql/mysql-server/commit/b4287f93857bf2f99b18fd06f555bbe5b12debfc)
Aurora MySQL database engine updates 2019-11-25 (version 2.07.0) (Deprecated)	2.07.0	• Bug #26251621: INCORRECT BEHAVIOR WITH TRIGGER AND GCOL • Bug #22574695: ASSERTION `!TABLE || (!TABLE->READ_SET || BITMAP_IS_SET(TABLE->READ_SET, FIEL • Bug #25966845: INSERT ON DUPLICATE KEY GENERATE A DEADLOCK • Bug #23070734: CONCURRENT TRUNCATE TABLES CAUSE STALL • Bug #26191879: FOREIGN KEY CASCADES USE EXCESSIVE MEMORY • Bug #20989615: INNODB AUTO_INCREMENT PRODUCES SAME VALUE TWICE

Database engine update	Version	MySQL bugs fixed
Aurora MySQL database engine updates 2019-11-11 (version 2.05.0) (Deprecated)	2.05.0	• Bug#23054591: PURGE BINARY LOGS TO is reading the whole binlog file and causing MySql to stall

Database engine update	Version	MySQL bugs fixed
Aurora MySQL database engine updates 2020-08-14 (version 2.04.9) (Deprecated)	2.04.9	• Bug #23070734, Bug #80060: Concurrent TRUNCATE TABLEs cause stalls • Bug #23103937: PS_TRUNCATE_ALL_TABLES() DOES NOT WORK IN SUPER_READ_ONLY MODE • Bug#22551677: When taking the server offline, a race condition within the Performance Schema could lead to a server exit. • Bug #27082268: Invalid FTS sync synchronization. • BUG #12589870: Fixed an issues which causes a restart with multi-query statement when query cache is enabled. • Bug #26402045: Certain cases of subquery materialization could cause a server exit. These queries now produce an error suggesting that materialization be disabled. • Bug #18898433: Queries with many left joins were slow if join buffering was used (for example, using the block nested loop algorithm). • Bug #25222337: A NULL virtual column field name in a virtual index caused a server exit during a field name comparison that occurs while populating virtual columns affected by a foreign key constraint. (https://github.com/mysql/mysql-server/commit/273d5c9d7072c63b6c47dbef6963d7dc491d5131) • Bug #25053286: Executing a stored procedure containing a query that accessed a view could allocate memory that was not freed until the session ended. (https://github.com/mysql/mysql-server/commit/d7b37d4d141a95f577916448650c429f0d6e193d) • Bug #25586773: Executing a stored procedure containing a statement that created a table from the contents of certain SELECT (https://dev.mysql.com/doc/refman/5.7/en/select.html) statements could result in a memory leak. (https://

Database engine update	Version	MySQL bugs fixed
		github.com/mysql/mysql-server/commit/88301e5adab65f6750f66af284be410c4369d0c1) • Bug #26666274: INFINITE LOOP IN PERFORMANCE SCHEMA BUFFER CONTAINER. • Bug #23550835, Bug #23298025, Bug #81464: A SELECT Performance Schema tables when an internal buffer was full could cause a server exit.
Aurora MySQL database engine updates 2019-09-19 (version 2.04.6) (Deprecated)	2.04.6	• Bug#23054591: PURGE BINARY LOGS TO is reading the whole binlog file and causing MySql to stall
Aurora MySQL database engine updates 2019-05-02 (version 2.04.2) (Deprecated)	2.04.2	Bug #24829050 - INDEX_MERGE_INTERSECTION OPTIMIZATION CAUSES WRONG QUERY RESULTS
Aurora MySQL database engine updates 2018-10-11 (version 2.03) (Deprecated)	2.03	• REVERSE SCAN ON A PARTITIONED TABLE DOES ICP - ORDER BY DESC (Bug #24929748). • JSON_OBJECT CREATES INVALID JSON CODE (Bug#26867509). • INSERTING LARGE JSON DATA TAKES AN INORDINATE AMOUNT OF TIME (Bug #22843444). • PARTITIONED TABLES USE MORE MEMORY IN 5.7 THAN 5.6 (Bug #25080442).

Database engine update	Version	MySQL bugs fixed
Aurora MySQL database engine updates 2018-09-21 (version 2.02.4) (Deprecated)	2.02.4	• BUG#13651665 INNODB MAY BE UNABLE TO LOAD TABLE DEFINITION AFTER RENAME • BUG#21371070 INNODB: CANNOT ALLOCATE 0 BYTES. • BUG#21378944 FTS ASSERT ENC.SRC_ILIST_PTR != NULL, FTS_OPTIMIZE_WORD(), OPTIMIZE TABLE • BUG#21508537 ASSERTION FAILURE UT_A(!VIC TIM_TRX->READ_ONLY) • BUG#21983865 UNEXPECTED DEADLOCK WITH INNODB_AUTOINC_LOCK_MODE=0 • BUG#22679185 INVALID INNODB FTS DOC ID DURING INSERT • BUG#22899305 GCOLS: ASSERTION: !(COL->PR TYPE & 256). • BUG#22956469 MEMORY LEAK INTRODUCED IN 5.7.8 IN MEMORY/INNODB/OS0FILE • BUG#22996488 CRASH IN FTS_SYNC_INDEX WHEN DOING DDL IN A LOOP • BUG#23014521 GCOL:INNODB: ASSERTION: !IS_V • BUG#23021168 REPLICATION STOPS AFTER TRX IS ROLLED BACK ASYNC • BUG#23052231 ASSERTION: ADD_AUTOINC < DICT_TABLE_GET_N_USER_COLS • BUG#23149683 ROTATE INNODB MASTER KEY WITH KEYRING_OKV_CONF_DIR MISSING: SIGSEGV; SIGNAL 11 • BUG#23762382 INSERT VALUES QUERY WITH JOIN IN A SELECT CAUSES INCORRECT BEHAVIOR • BUG#25209512 CURRENT_TIMESTAMP PRODUCES ZEROS IN TRIGGER

Database engine update	Version	MySQL bugs fixed
		• BUG#26626277 BUG IN "INSERT... ON DUPLICATE KEY UPDATE" QUERY • BUG#26734162 INCORRECT BEHAVIOR WITH INSERT OF BLOB + ON DUPLICATE KEY UPDATE • BUG#27460607 INCORRECT WHEN INSERT SELECT's SOURCE TABLE IS EMPTY
Aurora MySQL database engine updates 2018-05-03 (version 2.02) (Deprecated)	2.02.0	Left join returns incorrect results on the outer side (Bug #22833364)

MySQL bugs fixed by Aurora MySQL 1.x database engine updates

MySQL 5.6-compatible version Aurora contains all MySQL bug fixes through MySQL 5.6.10. The following table identifies additional MySQL bugs that have been fixed by Aurora MySQL database engine updates, and which update they were fixed in.

Database engine update	Version	MySQL bugs fixed
Aurora MySQL database engine updates 2021-03-18 (version 1.23.2) (Deprecated)	1.23.2	• *Replication*: While a SHOW BINLOG EVENTS statement was executing, any parallel transaction was blocked. The fix ensures that the SHOW BINLOG EVENTS process now only acquires a lock for the duration of calculating the file's end position, therefore parallel transactions are not blocked for long durations. (Bug #76618, Bug #20928790)

Database engine update	Version	MySQL bugs fixed
Aurora MySQL database engine updates 2020-09-02 (version 1.23.0) (Deprecated)	1.23.0	• Binlog events with `ALTER TABLE ADD COLUMN ALGORITHM=QUICK` will be rewritten as `ALGORITHM =DEFAULT` to be compatible with the community edition. • BUG #22350047: IF CLIENT KILLED AFTER ROLLBACK TO SAVEPOINT PREVIOUS STMTS COMMITTED • Bug #29915479: RUNNING COM_REGISTER_SLAVE WITHOUT COM_BINLOG_DUMP CAN RESULTS IN SERVER EXIT • Bug #30441969: BUG #29723340: MYSQL SERVER CRASH AFTER SQL QUERY WITH DATA ?AST • Bug #30628268: OUT OF MEMORY CRASH • Bug #27081349: UNEXPECTED BEHAVIOUR WHEN DELETE WITH SPATIAL FUNCTION • Bug #27230859: UNEXPECTED BEHAVIOUR WHILE HANDLING INVALID POLYGON" • Bug #27081349: UNEXPECTED BEHAVIOUR WHEN DELETE WITH SPATIAL" • Bug #26935001: ALTER TABLE AUTO_INCREMENT TRIES TO READ INDEX FROM DISCARDED TABLESPACE • Bug #29770705: SERVER CRASHED WHILE EXECUTING SELECT WITH SPECIFIC WHERE CLAUSE • Bug #27659490: SELECT USING DYNAMIC RANGE AND INDEX MERGE USE TOO MUCH MEMORY(OOM) • Bug #24786290: REPLICATION BREAKS AFTER BUG #74145 HAPPENS IN MASTER • Bug #27703912: EXCESSIVE MEMORY USAGE WITH MANY PREPARE • Bug #20527363: TRUNCATE TEMPORARY TABLE CRASH: ! DICT_TF2_FLAG_IS_SET(TABLE, DICT_TF2_TEMPORARY) • Bug#23103937 PS_TRUNCATE_ALL_TABLES() DOES NOT WORK IN SUPER_READ_ONLY MODE

Database engine update	Version	MySQL bugs fixed
		• Bug #25053286: USE VIEW WITH CONDITION IN PROCEDURE CAUSES INCORRECT BEHAVIOR (fixed in 5.6.36) • Bug #25586773: INCORRECT BEHAVIOR FOR CREATE TABLE SELECT IN A LOOP IN SP (fixed in 5.6.39) • Bug #27407480: AUTOMATIC_SP_PRIVILEGES REQUIRES NEED THE INSERT PRIVILEGES FOR MYSQL.USER TABLE • Bug #26997096: `relay_log_space` value is not updated in a synchronized manner so that its value is sometimes much higher than the actual disk space used by relay logs. • Bug#15831300 SLAVE_TYPE_CONVERSIONS=ALL_NON_LOSSY NOT WORKING AS EXPECTED • SSL Bug backport Bug #17087862, Bug #20551271 • Bug #16894092: PERFORMANCE REGRESSION IN 5.6.6+ FOR INSERT INTO ... SELECT ... FROM (fixed in 5.6.15). • Port a bug fix related to `SLAVE_TYPE_CONVERSIONS` .

Database engine update	Version	MySQL bugs fixed
Aurora MySQL database engine updates 2020-11-09 (version 1.22.3) (Deprecated)	1.22.3	• Bug #26654685: A corrupt index ID encountered during a foreign key check raised an assertion • Bug #15831300: By default, when promoting integers from a smaller type on the master to a larger type on the slave (for example, from a SMALLINT column on the master to a BIGINT column on the slave), the promoted values are treated as though they are signed. Now in such cases it is possible to modify or override this behavior using one or both of `ALL_SIGNED` , `ALL_UNSIGNED` in the set of values specified for the slave_type_conversions server system variable. For more information, see Row-based replication: attribute promotion and demotion, as well as the description of the variable. • Bug #17449901: With `foreign_key_checks=0` , InnoDB permitted an index required by a foreign key constraint to be dropped, placing the table into an inconsistent and causing the foreign key check that occurs at table load to fail. InnoDB now prevents dropping an index required by a foreign key constraint, even with foreign_key_checks=0. The foreign key constraint must be removed before dropping the foreign key index. • BUG #20768847: An ALTER TABLE ... DROP INDEX operation on a table with foreign key dependencies raised an assertion.

Database engine update	Version	MySQL bugs fixed
Aurora MySQL database engine updates 2019-11-25 (version 1.22.0) (Deprecated)	1.22.0	• Bug#16346241 - SERVER CRASH IN ITEM_PARAM::QUERY_VAL_STR • Bug#17733850 - NAME_CONST() CRASH IN ITEM_NAME_CONST::ITEM_NAME_CONST() • Bug #20989615 - INNODB AUTO_INCREMENT PRODUCES SAME VALUE TWICE • Bug #20181776 - ACCESS CONTROL DOESN'T MATCH MOST SPECIFIC HOST WHEN IT CONTAINS WILDCARD • Bug #27326796 - MYSQL CRASH WITH INNODB ASSERTION FAILURE IN FILE PARS0PARS.CC • Bug #20590013 - IF YOU HAVE A FULLTEXT INDEX AND DROP IT YOU CAN NO LONGER PERFORM ONLINE DDL
Aurora MySQL database engine updates 2019-11-25 (version 1.21.0) (Deprecated)	1.21.0	• Bug #19929406: HANDLE_FATAL_SIGNAL (SIG=11) IN __MEMMOVE_SSSE3_BACK FROM STRING::COPY • Bug #17059925: For UNION statements, the rows-examined value was calculated incorrectly. This was manifested as too-large values for the `ROWS_EXAMINED` column of Performance Schema statement tables (such as events_statements_current). • Bug #11827369: Some queries with `SELECT ... FROM DUAL` nested subqueries raised an assertion. • Bug #16311231: Incorrect results were returned if a query contained a subquery in an `IN` clause that contained an XOR operation in the `WHERE` clause.

Database engine update	Version	MySQL bugs fixed
Aurora MySQL database engine updates 2019-11-11 (version 1.20.0) (Deprecated)	1.20.0	• Bug #19929406: HANDLE_FATAL_SIGNAL (SIG=11) IN __MEMMOVE_SSSE3_BACK FROM STRING::COPY • Bug #17059925: For UNION statements, the rows-examined value was calculated incorrectly. This was manifested as too-large values for the ROWS_EXAMINED column of Performance Schema statement tables (such as events_statements_current). • Bug #11827369: Some queries with SELECT ... FROM DUAL nested subqueries raised an assertion. • Bug #16311231: Incorrect results were returned if a query contained a subquery in an IN clause that contained an XOR operation in the WHERE clause.
Aurora MySQL database engine updates 2019-09-19 (version 1.19.5) (Deprecated)	1.19.5	• CVE-2018-2696 • CVE-2015-4737 • Bug #19929406: HANDLE_FATAL_SIGNAL (SIG=11) IN __MEMMOVE_SSSE3_BACK FROM STRING::COPY • Bug #17059925: For UNION statements, the rows-examined value was calculated incorrectly. This was manifest as too-large values for the ROWS_EXAMINED column of Performance Schema statement tables (such as events_statements_current). • Bug #11827369: Some queries with SELECT ... FROM DUAL nested subqueries raised an assertion. • Bug #16311231: Incorrect results were returned if a query contained a subquery in an IN clause which contained an XOR operation in the WHERE clause.

Database engine update	Version	MySQL bugs fixed
Aurora MySQL database engine updates 2019-02-07 (version 1.19.0) (Deprecated)	1.19.0	• BUG #32917: DETECT ORPHAN TEMP-POOL FILES, AND HANDLE GRACEFULLY • BUG #63144 CREATE TABLE IF NOT EXISTS METADATA LOCK IS TOO RESTRICTIVE
Aurora MySQL database engine updates 2019-01-17 (version 1.17.8) (Deprecated)	1.17.8	• BUG #13418638: CREATE TABLE IF NOT EXISTS METADATA LOCK IS TOO RESTRICTIVE
Aurora MySQL database engine updates 2018-10-08 (version 1.17.7) (Deprecated)	1.17.7	• Drop index on a foreign key column leads to missing table. (Bug #16208542) • Memory leak in add_derived_key(). (Bug #76349) • For partitioned tables, queries could return different results depending on whether Index Merge was used. (Bug #16862316) • Queries using the index_merge optimization (see Index merge optimization) could return invalid results when run against tables that were partitioned by HASH. (Bug #17588348)
Aurora MySQL database engine updates 2018-09-06 (version 1.17.6) (Deprecated)	1.17.6	• For an ALTER TABLE statement that renamed or changed the default value of a BINARY column, the alteration was done using a table copy and not in place. (Bug #67141, Bug #14735373, Bug #69580, Bug #17024290) • An outer join between a regular table and a derived table that is implicitly groups could cause a server exit. (Bug #16177639)

Database engine update	Version	MySQL bugs fixed
Aurora MySQL database engine updates 2018-03-13 (version 1.17) (Deprecated)	1.17.0	• LAST_INSERT_ID is replicated incorrectly if replication filters are used (Bug #69861) • Query returns different results depending on whether INDEX_MERGE setting (Bug #16862316) • Query proc re-execute of stored routine, inefficient query plan (Bug #16346367) • InnoDB FTS : Assert in FTS_CACHE_APPEND_DELETED_DOC_IDS (Bug #18079671) • Assert RBT_EMPTY(INDEX_CACHE->WORDS) in ALTER TABLE CHANGE COLUMN (Bug #17536995) • InnoDB fulltext search doesn't find records when savepoints are involved (Bug #70333, Bug #17458835)
Aurora MySQL database engine updates 2017-11-20 (version 1.15.1) (Deprecated)	1.15.1	• Reverted — MySQL instance stalling "doing SYNC index" (Bug #73816) • Reverted — Assert RBT_EMPTY(INDEX_CACHE->WORDS) in ALTER TABLE CHANGE COLUMN (Bug #17536995) • Reverted — InnoDB Fulltext search doesn't find records when savepoints are involved (Bug #70333)

Database engine update	Version	MySQL bugs fixed
Aurora MySQL database engine updates 2017-10-24 (version 1.15) (Deprecated)	1.15.0	• CREATE USER accepts plugin and password hash, but ignores the password hash (Bug #78033) • The partitioning engine adds fields to the read bit set to be able to return entries sorted from a partitioned index. This leads to the join buffer will try to read unneeded fields. Fixed by not adding all partitioning fields to the read_set, but instead only sort on the already set prefix fields in the read_set. Added a DBUG_ASSERT that if doing key_cmp, at least the first field must be read (Bug #16367691). • MySQL instance stalling "doing SYNC index" (Bug #73816) • Assert RBT_EMPTY(INDEX_CACHE->WORDS) in ALTER TABLE CHANGE COLUMN (Bug #17536995) • InnoDB Fulltext search doesn't find records when savepoints are involved (Bug #70333)
Aurora MySQL database engine updates: 2018-03-13 (version 1.14.4) (Deprecated)	1.14.4	• Ignorable events don't work and are not tested (Bug #74683) • NEW->OLD ASSERT FAILURE 'GTID_MODE > 0' (Bug #20436436)
Aurora MySQL database engine updates: 2017-08-07 (version 1.14) (Deprecated)	1.14.0	A full-text search combined with derived tables (subqueries in the FROM clause) caused a server exit. Now, if a full-text operation depends on a derived table, the server produces an error indicating that a full-text search cannot be done on a materialized table. (Bug #68751, Bug #16539903)

Database engine update	Version	MySQL bugs fixed
Aurora MySQL database engine updates: 2017-05-15 (version 1.13) (Deprecated)	1.13.0	• Reloading a table that was evicted while empty caused an AUTO_INCREMENT value to be reset. (Bug #21454472, Bug #77743) • An index record was not found on rollback due to inconsist encies in the purge_node_t structure. The inconsistency resulted in warnings and error messages such as "error in sec index entry update", "unable to purge a record", and "tried to purge sec index entry not marked for deletion". (Bug #19138298, Bug #70214, Bug #21126772, Bug #21065746) • Wrong stack size calculation for qsort operation leads to stack overflow. (Bug #73979) • Record not found in an index upon rollback. (Bug #70214, Bug #72419) • ALTER TABLE add column TIMESTAMP on update CURRENT_TIMESTAMP inserts ZERO-datas (Bug #17392)
Aurora MySQL database engine updates: 2017-04-05 (version 1.12) (Deprecated)	1.12.0	• Reloading a table that was evicted while empty caused an AUTO_INCREMENT value to be reset. (Bug #21454472, Bug #77743) • An index record was not found on rollback due to inconsist encies in the purge_node_t structure. The inconsistency resulted in warnings and error messages such as "error in sec index entry update", "unable to purge a record", and "tried to purge sec index entry not marked for deletion". (Bug #19138298, Bug #70214, Bug #21126772, Bug #21065746) • Wrong stack size calculation for qsort operation leads to stack overflow. (Bug #73979) • Record not found in an index upon rollback. (Bug #70214, Bug #72419) • ALTER TABLE add column TIMESTAMP on update CURRENT_TIMESTAMP inserts ZERO-datas (Bug #17392)

Database engine update	Version	MySQL bugs fixed
Aurora MySQL database engine updates: 2017-02-23 (version 1.11) (Deprecated)	1.11.0	• Running ALTER table DROP foreign key simultaneously with another DROP operation causes the table to disappear. (Bug #16095573) • Some INFORMATION_SCHEMA queries that used ORDER BY did not use a filesort optimization as they did previously. (Bug #16423536) • FOUND_ROWS () returns the wrong count of rows on a table. (Bug #68458) • The server fails instead of giving an error when too many temp tables are open. (Bug #18948649)
Aurora MySQL database engine updates: 2016-12-14 (version 1.10) (Deprecated)	1.10.0	• UNION of derived tables returns wrong results with '1=0/fals e'-clauses. (Bug #69471) • Server crashes in ITEM_FUNC_GROUP_CONCAT::FIX_FIELDS on 2nd execution of stored procedure. (Bug #20755389) • Avoid MySQL queries from stalling for too long during FTS cache sync to disk by offloading the cache sync task to a separate thread, as soon as the cache size crosses 10% of the total size. (Bugs #22516559, #73816)
Aurora MySQL database engine updates: 2016-10-26 (version 1.8.1) (Deprecated)	1.8.1	• OpenSSL changed the Diffie-Hellman key length parameters due to the LogJam issue. (Bug #18367167)

Database engine update	Version	MySQL bugs fixed
Aurora MySQL database engine updates: 2016-10-18 (version 1.8) (Deprecated)	1.8.0	• When dropping all indexes on a column with multiple indexes, InnoDB failed to block a DROP INDEX operation when a foreign key constraint requires an index. (Bug #16896810) • Solve add foreign key constraint crash. (Bug #16413976) • Fixed a crash when fetching a cursor in a stored procedure, and analyzing or flushing the table at the same time. (Bug # 18158639) • Fixed an auto-increment bug when a user alters a table to change the AUTO_INCREMENT value to less than the maximum auto-increment column value. (Bug # 16310273)

Database engine update	Version	MySQL bugs fixed
Aurora MySQL database engine updates: 2016-08-30 (version 1.7.0) (Deprecated)	1.7.0	• Improve scalability by partitioning LOCK_grant lock. (Port WL #8355) • Opening cursor on SELECT in stored procedure causes segfault. (Port Bug #16499751) • MySQL gives the wrong result with some special usage. (Bug #11751794) • Crash in GET_SEL_ARG_FOR_KEYPART – caused by patch for bug #11751794. (Bug #16208709) • Wrong results for a simple query with GROUP BY. (Bug #17909656) • Extra rows on semijoin query with range predicates. (Bug #16221623) • Adding an ORDER BY clause following an IN subquery could cause duplicate rows to be returned. (Bug #16308085) • Crash with explain for a query with loose scan for GROUP BY, MyISAM. (Bug #16222245) • Loose index scan with quoted int predicate returns random data. (Bug #16394084) • If the optimizer was using a loose index scan, the server could exit while attempting to create a temporary table. (Bug #16436567) • COUNT(DISTINCT) should not count NULL values, but they were counted when the optimizer used loose index scan. (Bug #17222452) • If a query had both MIN()/MAX() and aggregate_function (DISTINCT) (for example, SUM(DISTINCT)) and was executed using loose index scan, the result values of MIN()/MAX() were set improperly. (Bug #17217128)

Database engine update	Version	MySQL bugs fixed
Aurora MySQL database engine updates: 2016-06-01 (version 1.6.5) (Deprecated)	1.6.5	• SLAVE CAN'T CONTINUE REPLICATION AFTER MASTER'S CRASH RECOVERY (Port Bug #17632285)
Aurora MySQL database engine updates: 2016-04-06 (version 1.6) (Deprecated)	1.6.0	• BACKPORT Bug #18694052 FIX FOR ASSERTION `!M_ORDERED_REC_BUFFER' FAILED TO 5.6 (Port Bug #18305270) • SEGV IN MEMCPY(), HA_PARTITION::POSITION (Port Bug # 18383840) • WRONG RESULTS WITH PARTITIONING,INDEX_MERGE AND NO PK (Port Bug # 18167648) • FLUSH TABLES FOR EXPORT: ASSERTION IN HA_PARTITION::EXTRA (Port Bug # 16943907) • SERVER CRASH IN VIRTUAL HA_ROWS HANDLER::MULTI_RANGE_READ_INFO_CONST (Port Bug # 16164031) • RANGE OPTIMIZER CRASHES IN SEL_ARG::RB_INSERT() (Port Bug # 16241773)
Aurora MySQL database engine updates: 2016-01-11 (version 1.5) (Deprecated)	1.5.0	• Addressed incomplete fix in MySQL full text search affecting tables where the database name begins with a digit. (Port Bug #17607956)

Database engine update	Version	MySQL bugs fixed
Aurora MySQL database engine updates: 2015-12-03 (version 1.4) (Deprecated)	1.4	• SEGV in FTSPARSE(). (Bug #16446108) • InnoDB data dictionary is not updated while renaming the column. (Bug #19465984) • FTS crash after renaming table to different database. (Bug #16834860) • Failed preparing of trigger on truncated tables cause error 1054. (Bug #18596756) • Metadata changes might cause problems with trigger execution. (Bug #18684393) • Materialization is not chosen for long UTF8 VARCHAR field. (Bug #17566396) • Poor execution plan when ORDER BY with limit X. (Bug #16697792) • Backport bug #11765744 TO 5.1, 5.5 AND 5.6. (Bug #17083851) • Mutex issue in SQL/SQL_SHOW.CC resulting in SIG6. Source likely FILL_VARIABLES. (Bug #20788853) • Backport bug #18008907 to 5.5+ versions. (Bug #18903155) • Adapt fix for a stack overflow error in MySQL 5.7. (Bug #19678930)

Database engine update	Version	MySQL bugs fixed
Aurora MySQL database engine updates: 2015-10-16 (versions 1.2, 1.3) (Deprecated)	1.2, 1.3	• Killing a query inside innodb causes it to eventually crash with an assertion. (Bug #1608883) • For failure to create a new thread for the event scheduler, event execution, or new connection, no message was written to the error log. (Bug #16865959) • If one connection changed its default database and simultaneously another connection executed SHOW PROCESSLIST, the second connection could access invalid memory when attempting to display the first connection's default database memory. (Bug #11765252) • PURGE BINARY LOGS by design does not remove binary log files that are in use or active, but did not provide any notice when this occurred. (Bug #13727933) • For some statements, memory leaks could result when the optimizer removed unneeded subquery clauses. (Bug #15875919) • During shutdown, the server could attempt to lock an uninitialized mutex. (Bug #16016493) • A prepared statement that used GROUP_CONCAT() and an ORDER BY clause that named multiple columns could cause the server to exit. (Bug #16075310) • Performance Schema instrumentation was missing for replica worker threads. (Bug #16083949) • `STOP SLAVE` could cause a deadlock when issued concurrently with a statement such as SHOW STATUS that retrieved the values for one or more of the status variables `Slave_retried_transactions`, `Slave_heartbeat_period`, `Slave_received_heartbeats`, `Slave_last_heartbeat`, or `Slave_running`. (Bug #16088188)

Database engine update	Version	MySQL bugs fixed
		• A full-text query using Boolean mode could return zero results in some cases where the search term was a quoted phrase. (Bug #16206253) • The optimizer's attempt to remove redundant subquery clauses raised an assertion when executing a prepared statement with a subquery in the ON clause of a join in a subquery. (Bug #16318585) • GROUP_CONCAT unstable, crash in ITEM_SUM::CLEAN_UP _AFTER_REMOVAL. (Bug #16347450) • Attempting to replace the default InnoDB full-text search (FTS) stopword list by creating an InnoDB table with the same structure as INFORMATION_SCHEMA.INNODB_F T_DEFAULT_STOPWORD would result in an error. (Bug #16373868) • After the client thread on a worker performed a FLUSH TABLES WITH READ LOCK and was followed by some updates on the master, the worker hung when executing `SHOW SLAVE STATUS`. (Bug #16387720) • When parsing a delimited search string such as "abc-def" in a full-text search, InnoDB now uses the same word delimiters as MyISAM. (Bug #16419661) • Crash in FTS_AST_TERM_SET_WILDCARD. (Bug #16429306) • SEGFAULT in FTS_AST_VISIT() for FTS RQG test. (Bug # 16435855) • For debug builds, when the optimizer removed an Item_ref pointing to a subquery, it caused a server exit. (Bug #16509874) • Full-text search on InnoDB tables failed on searches for literal phrases combined with + or - operators. (Bug #16516193) • `START SLAVE` failed when the server was started with the options`--master-info-repository=TABLE relay-`

Database engine update	Version	MySQL bugs fixed
		log-info-repository=TABLE and with autocommit set to 0, together with `--skip-slave-start` . (Bug #16533802) • Very large InnoDB full-text search (FTS) results could consume an excessive amount of memory. (Bug #16625973) • In debug builds, an assertion could occur in OPT_CHECK _ORDER_BY when using binary directly in a search string, as binary might include NULL bytes and other non-meaningful characters. (Bug #16766016) • For some statements, memory leaks could result when the optimizer removed unneeded subquery clauses. (Bug #16807641) • It was possible to cause a deadlock after issuing FLUSH TABLES WITH READ LOCK by issuing `STOP SLAVE` in a new connection to the worker, then issuing `SHOW SLAVE STATUS` using the original connection. (Bug #16856735) • GROUP_CONCAT() with an invalid separator could cause a server exit. (Bug #16870783) • The server did excessive locking on the LOCK_active_mi and active_mi->rli->data_lock mutexes for any SHOW STATUS LIKE 'pattern' statement, even when the pattern did not match status variables that use those mutexes (`Slave_heartbeat_period` , `Slave_last_heartbeat` , `Slave_received_heartbeats` , `Slave_retried_transactions` , `Slave_running`). (Bug #16904035) • A full-text search using the IN BOOLEAN MODE modifier would result in an assertion failure. (Bug #16927092) • Full-text search on InnoDB tables failed on searches that used the + boolean operator. (Bug #17280122) • 4-way deadlock: zombies, purging binlogs, show processlist, show binlogs. (Bug #17283409)

Database engine update	Version	MySQL bugs fixed
		• When an SQL thread which was waiting for a commit lock was killed and restarted it caused a transaction to be skipped on worker. (Bug #17450876) • An InnoDB full-text search failure would occur due to an "unended" token. The string and string length should be passed for string comparison. (Bug #17659310) • Large numbers of partitioned InnoDB tables could consume much more memory when used in MySQL 5.6 or 5.7 than the memory used by the same tables used in previous releases of the MySQL Server. (Bug #17780517) • For full-text queries, a failure to check that num_token is less than max_proximity_item could result in an assertion. (Bug #18233051) • Certain queries for the INFORMATION_SCHEMA TABLES and COLUMNS tables could lead to excessive memory use when there were large numbers of empty InnoDB tables. (Bug #18592390) • When committing a transaction, a flag is now used to check whether a thread has been created, rather than checking the thread itself, which uses more resources, particularly when running the server with master_info_repository=TABLE. (Bug #18684222) • If a client thread on a worker executed FLUSH TABLES WITH READ LOCK while the master executed a DML, executing `SHOW SLAVE STATUS` in the same client became blocked, causing a deadlock. (Bug #19843808) • Ordering by a GROUP_CONCAT() result could cause a server exit. (Bug #19880368)

Database engine update	Version	MySQL bugs fixed
Aurora MySQL database engine updates: 2015-08-24 (version 1.1) (Deprecated)	1.1	• InnoDB databases with names beginning with a digit cause a full-text search (FTS) parser error. (Bug #17607956) • InnoDB full-text searches fail in databases whose names began with a digit. (Bug #17161372) • For InnoDB databases on Windows, the full-text search (FTS) object ID is not in the expected hexadecimal format. (Bug #16559254) • A code regression introduced in MySQL 5.6 negatively impacted DROP TABLE and ALTER TABLE performance. This could cause a performance drop between MySQL Server 5.5.x and 5.6.x. (Bug #16864741)

Security vulnerabilities fixed in Aurora MySQL

Common Vulnerabilities and Exposures (CVE) is a list of entries for publicly known cybersecurity vulnerabilities. Each entry contains an identification number, a description, and at least one public reference.

You can find on this page a list of security vulnerabilities fixed in Aurora MySQL. For general information about security for Aurora, see Security in Amazon Aurora in the *Amazon Aurora User Guide*. For additional security information for Aurora MySQL, see Security with Amazon Aurora MySQL in the *Amazon Aurora User Guide*.

We recommend that you always upgrade to the latest Aurora release to be protected against known vulnerabilities. You can use this page to verify whether a particular version of Aurora MySQL has a fix for a specific security vulnerability. If your cluster doesn't have the security fix, you can see which Aurora MySQL version you should upgrade to for that fix.

CVEs fixed in Aurora MySQL version 1, 2 and 3 are also listed in the release notes for that version:

- Database engine updates for Amazon Aurora MySQL version 3
- Database engine updates for Amazon Aurora MySQL version 2
- Database engine updates for Amazon Aurora MySQL version 1 (Deprecated)

> **Note**
>
> The initial release of Aurora MySQL version 3 includes all CVEs fixed up to community MySQL 8.0.23. For future CVEs that are fixed, look for them listed here and in the Aurora MySQL version 3 release notes.

CVEs and minimum fixed Aurora MySQL versions

- CVE-2024-20963: 2.12.2, 2.11.5
- CVE-2024-0853: 3.06.1, 3.04.3
- CVE-2023-44487: 3.06.1
- CVE-2023-39975: 3.07.0, 3.06.0, 3.05.2, 3.04.2, 2.12.2, 2.11.5
- CVE-2023-38546: 3.07.0, 3.06.0, 3.04.2, 2.11.5

- CVE-2023-38545: 3.07.0, 3.06.0, 3.05.2, 3.05.1, 3.05.0.1, 3.04.2, 3.03.3, 2.12.2, 2.12.1, 2.12.0.1, 2.11.5
- CVE-2023-22084: 2.11.5
- CVE-2023-22053: 2.12.1
- CVE-2023-22028: 2.12.1, 2.11.5
- CVE-2023-22026: 2.12.1, 2.11.5
- CVE-2023-22015: 2.12.1, 2.11.5
- CVE-2023-21963: 3.04.0, 3.03.2, 2.12.0, 2.11.3
- CVE-2023-21912: 3.04.0, 3.03.2, 2.12.0, 2.11.3
- CVE-2023-21840: 2.12.0
- CVE-2023-0215: 3.04.0, 3.03.2, 2.12.0, 2.11.3
- CVE-2022-43551: 3.04.0, 3.03.2, 2.12.0, 2.11.3
- CVE-2022-37434: 3.05.0, 3.04.0, 3.03.2, 2.12.0, 2.11.3
- CVE-2022-32221: 3.03.0, 2.12.0, 2.11.1, 2.07.9
- CVE-2022-24407: 2.12.1, 2.11.4
- CVE-2022-21635: 3.04.0
- CVE-2022-21556: 3.04.0
- CVE-2022-21460: 2.11.0
- CVE-2022-21451: 3.03.0, 3.02.2, 2.11.0
- CVE-2022-21444: 3.03.0, 3.02.2, 2.11.0, 2.10.3
- CVE-2022-21417 : 2.11.0
- CVE-2022-21352: 3.04.0
- CVE-2022-21344: 2.10.3
- CVE-2022-21304: 2.11.0, 2.10.3
- CVE-2022-21303: 2.11.0
- CVE-2022-21245: 2.11.0, 2.10.3, 2.07.8
- CVE-2022-0778: 3.02.1, 2.11.0
- CVE-2021-36222: 3.02.2, 3.01.1, 2.12.0, 2.11.1, 2.11.0, 2.10.3, 2.10.2, 2.07.8
- CVE-2021-35630: 3.04.0
- CVE-2021-35624: 3.04.0, 2.10.2
- CVE-2021-35604: 2.10.2

- CVE-2021-28196: 2.11.0
- CVE-2021-23841: 2.11.0, 2.10.0, 2.09.3, 1.23.3
- CVE-2021-22946: 3.02.0, 3.01.1, 2.12.0
- CVE-2021-22926: 3.02.2, 3.01.1, 2.11.1, 2.11.0, 2.10.3, 2.10.22.07.8
- CVE-2021-3712: 2.09.3
- CVE-2021-3449: 2.11.0, 2.10.0, 2.09.3, 1.23.3
- CVE-2021-2390: 2.10.2
- CVE-2021-2389: 2.10.2
- CVE-2021-2385: 2.10.2
- CVE-2021-2356: 2.10.2
- CVE-2021-2307: 2.11.0, 2.10.1, 2.09.3, 1.23.4
- CVE-2021-2226: 2.11.0, 2.10.1, 2.09.3, 1.23.4
- CVE-2021-2202: 2.11.0
- CVE-2021-2194: 2.11.0, 2.10.1
- CVE-2021-2179: 2.11.0
- CVE-2021-2178: 2.11.0
- CVE-2021-2174: 2.11.0, 2.10.1, 2.09.3
- CVE-2021-2171: 2.11.0, 2.10.1, 2.09.3
- CVE-2021-2169: 2.12.0, 2.11.1, 2.11.0, 2.10.1, 2.09.3
- CVE-2021-2166: 2.11.0, 2.10.1, 2.09.3
- CVE-2021-2160: 2.11.0, 2.10.1, 1.23.4
- CVE-2021-2154: 2.11.0, 2.10.1, 2.09.3, 1.23.4
- CVE-2021-2144: 2.07.3
- CVE-2021-2060: 2.10.1, 2.09.3, 1.23.4
- CVE-2021-2032: 2.10.1, 2.09.3, 1.23.4
- CVE-2021-2001: 2.10.1, 2.09.3, 1.23.4
- CVE-2020-28196: 2.10.0, 2.09.3, 1.23.3
- CVE-2020-14867: 1.23.2, 1.22.4
- CVE-2020-14812: 2.09.2, 2.07.4, 1.23.2, 1.22.4
- CVE-2020-14793: 2.09.2, 2.07.4, 1.23.2, 1.22.4

- CVE-2020-14790: 2.10.0, 2.09.2, 2.07.4
- CVE-2020-14776: 2.10.0
- CVE-2020-14775: 2.09.2, 2.07.4
- CVE-2020-14769: 2.09.3, 2.09.2, 2.07.4, 1.23.2, 1.22.4
- CVE-2020-14765: 2.09.2, 2.07.4, 1.23.2, 1.22.4
- CVE-2020-14760: 2.09.2, 2.07.4
- CVE-2020-14672: 2.09.2, 2.07.4, 1.23.2, 1.22.4
- CVE-2020-14567: 2.10.0, 2.09.1, 2.08.3, 2.07.3
- CVE-2020-14559: 2.10.0, 2.09.1, 2.08.3, 2.07.3, 1.23.1, 1.22.3
- CVE-2020-14553: 2.10.0, 2.09.1, 2.08.3, 2.07.3
- CVE-2020-14547: 2.10.0, 2.09.1, 2.08.3, 2.07.3
- CVE-2020-14540: 2.10.0, 2.09.1, 2.08.3, 2.07.3
- CVE-2020-14539: 2.10.0, 1.23.1, 1.22.3
- CVE-2020-11105: 3.07.0, 3.06.0, 3.05.2, 3.04.2, 2.12.1, 2.11.5
- CVE-2020-11104: 3.07.0, 3.06.0, 3.05.2, 3.04.2, 2.12.1, 2.11.5
- CVE-2020-2812: 2.09.1, 2.08.3, 2.07.3, 1.22.3
- CVE-2020-2806: 2.09.1, 2.08.3, 2.07.3
- CVE-2020-2780: 2.09.1, 2.08.3, 2.07.3, 1.22.3
- CVE-2020-2765: 2.09.1, 2.08.3, 2.07.3
- CVE-2020-2763: 2.09.1, 2.08.3, 2.07.3, 1.22.3
- CVE-2020-2760: 2.09.1, 2.08.3, 2.07.3, 2.04.9
- CVE-2020-2579: 2.09.1, 2.08.3, 2.07.3, 1.22.3
- CVE-2020-1971: 2.09.2, 2.07.4, 1.23.2, 1.22.4
- CVE-2019-17543: 2.10.2, 2.09.3, 2.07.7
- CVE-2019-5443: 2.08.0, 2.04.9
- CVE-2019-3822: 2.08.0, 2.04.9
- CVE-2019-2960: 2.10.2, 2.09.3, 2.07.7
- CVE-2019-2948: 2.09.0
- CVE-2019-2924: 2.07.0, 2.04.9, 1.22.0
- CVE-2019-2923: 2.07.0, 2.04.9, 1.22.0

- CVE-2019-2922: 2.07.0, 2.04.9, 1.22.0
- CVE-2019-2911: 2.09.0, 2.04.9, 1.23.0
- CVE-2019-2910: 2.07.0, 2.04.9, 1.22.0
- CVE-2019-2805: 2.06.0, 2.04.9, 1.22.0
- CVE-2019-2791: 2.06.0, 2.04.9
- CVE-2019-2778: 2.06.0, 2.04.9
- CVE-2019-2758: 2.06.0, 2.04.9
- CVE-2019-2740: 2.07.3, 2.04.9, 1.22.0
- CVE-2019-2739: 2.06.0, 2.04.9
- CVE-2019-2731: 2.09.0
- CVE-2019-2730: 2.06.0, 2.04.9, 1.22.0
- CVE-2019-2628: 2.04.9
- CVE-2019-2581: 2.09.0
- CVE-2019-2537: 2.09.0, 1.23.0
- CVE-2019-2534: 2.05.0, 2.04.3, 1.21.0, 1.20.0, 1.19.1
- CVE-2019-2482: 2.09.0
- CVE-2019-2434: 2.09.0
- CVE-2019-2420: 2.09.0
- CVE-2018-3284: 2.09.0
- CVE-2018-3251: 2.10.0
- CVE-2018-3156: 2.10.0
- CVE-2018-3155: 2.05.0, 2.04.3
- CVE-2018-3143: 2.10.0, 1.23.2
- CVE-2018-3065: 2.09.0
- CVE-2018-3064: 2.06.0, 2.04.9, 1.22.0
- CVE-2018-3058: 2.06.0, 2.04.9, 1.22.0
- CVE-2018-3056: 2.05.0, 2.04.4
- CVE-2018-2813: 2.04.9
- CVE-2018-2787: 2.09.0, 1.23.0
- CVE-2018-2786: 2.06.0, 2.04.9

- CVE-2018-2784: 2.09.0, 1.23.0
- CVE-2018-2696: 2.05.0, 2.04.5, 1.21.0, 1.20.0, 1.19.5
- CVE-2018-2645: 2.09.0, 1.23.0
- CVE-2018-2640: 2.09.0, 1.23.0
- CVE-2018-2612: 2.05.0, 2.04.3, 1.21.0, 1.20.0, 1.19.1
- CVE-2018-2562: 2.05.0, 2.04.4, 1.21.0, 1.20.0, 1.19.2
- CVE-2018-0734: 2.05.0, 2.04.3, 1.21.0, 1.20.0, 1.19.1
- CVE-2017-3653: 2.06.0, 2.04.9, 1.22.0
- CVE-2017-3599: 2.05.0, 2.04.3, 1.21.0, 1.20.0, 1.19.1
- CVE-2017-3465: 2.06.0, 2.04.9
- CVE-2017-3464: 1.22.0, 2.04.9
- CVE-2017-3455: 2.06.0, 2.04.9
- CVE-2017-3329: 2.05.0, 2.04.4, 1.21.0, 1.20.0, 1.19.2
- CVE-2017-3244: 2.06.0, 2.04.9, 1.22.0
- CVE-2016-8287: 2.07.2
- CVE-2016-5634: 2.07.2
- CVE-2016-5612: 2.06.0, 2.04.9, 1.22.0
- CVE-2016-5440: 2.10.0
- CVE-2016-5439: 1.22.0, 2.03.3
- CVE-2016-5436: 2.04.9, 2.03.3
- CVE-2016-3518: 2.04.5
- CVE-2016-3495: 2.03.2
- CVE-2016-3486: 2.02.2
- CVE-2016-0606: 1.22.0
- CVE-2015-4904: 1.22.0
- CVE-2015-4879: 1.22.0
- CVE-2015-4864: 1.22.0
- CVE-2015-4830: 1.22.0
- CVE-2015-4826: 1.22.0
- CVE-2015-4737: 1.21.0, 1.20.0, 1.19.5

- CVE-2015-2620: 1.22.0
- CVE-2015-0382: 1.22.0
- CVE-2015-0381: 1.22.0
- CVE-2014-6555: 1.22.0
- CVE-2014-6489: 1.22.0
- CVE-2014-4260: 1.22.0
- CVE-2014-4258: 1.22.0
- CVE-2014-2444: 1.22.0
- CVE-2014-2436: 1.22.0
- CVE-2014-0393: 1.22.0
- CVE-2013-5908: 1.22.0
- CVE-2013-5881: 1.22.0
- CVE-2013-5807: 1.22.0
- CVE-2013-3811: 1.22.0
- CVE-2013-3807: 1.22.0
- CVE-2013-3806: 1.22.0
- CVE-2013-3804: 1.22.0
- CVE-2013-2381: 1.22.0
- CVE-2013-2378: 1.22.0
- CVE-2013-2375: 1.22.0
- CVE-2013-1523: 1.22.0
- CVE-2012-5615: 1.22.0

Document history for the Aurora MySQL Release Notes

The following table describes the documentation releases for Aurora MySQL Release Notes.

Change	Description	Date
Aurora MySQL version 3.06.1, compatible with MySQL 8.0.34	Aurora MySQL version 3.06.1 is available. This version is compatible with MySQL 8.0.34. For full details, see Aurora MySQL version 3 compatible with MySQL 8.0.	June 26, 2024
Aurora MySQL version 3.04.3, compatible with MySQL 8.0.28	Aurora MySQL version 3.04.3 is available. This version is compatible with MySQL 8.0.28. For full details, see Aurora MySQL version 3 compatible with MySQL 8.0.	June 26, 2024
Aurora MySQL version 3.07.0, compatible with MySQL 8.0.36	Aurora MySQL version 3.07.0 is available. This version is compatible with MySQL 8.0.36. For full details, see Aurora MySQL version 3 compatible with MySQL 8.0.	June 4, 2024
Aurora MySQL version 2.11.5, compatible with MySQL 5.7.12	Aurora MySQL version 2.11.5 is available. This version is compatible with MySQL 5.7.12. For full details, see Aurora MySQL version 2 compatible with MySQL 5.7.	March 26, 2024
Aurora MySQL version 2.12.2, compatible with MySQL 5.7.44	Aurora MySQL version 2.12.2 is available. This version is compatible with MySQL	March 19, 2024

	5.7.44. For full details, see Aurora MySQL version 2 compatible with MySQL 5.7.	
Aurora MySQL version 3.04.2, compatible with MySQL 8.0.28	Aurora MySQL version 3.04.2 is available. This version is compatible with MySQL 8.0.28. For full details, see Aurora MySQL version 3 compatible with MySQL 8.0.	March 15, 2024
Aurora MySQL version 3.06.0, compatible with MySQL 8.0.34	Aurora MySQL version 3.06.0 is available. This version is compatible with MySQL 8.0.34. For full details, see Aurora MySQL version 3 compatible with MySQL 8.0.	March 7, 2024
Aurora MySQL version 3.05.2, compatible with MySQL 8.0.32	Aurora MySQL version 3.05.2 is available. This version is compatible with MySQL 8.0.32. For full details, see Aurora MySQL version 3 compatible with MySQL 8.0.	January 31, 2024
Aurora MySQL version 2.12.1, compatible with MySQL 5.7.40	Aurora MySQL version 2.12.1 is available. This version is compatible with MySQL 5.7.40. For full details, see Aurora MySQL version 2 compatible with MySQL 5.7.	December 28, 2023
Aurora MySQL version 3.03.3, compatible with MySQL 8.0.26	Aurora MySQL version 3.03.3 is available. This version is compatible with MySQL 8.0.26. For full details, see Aurora MySQL version 3 compatible with MySQL 8.0.	December 8, 2023

Aurora MySQL version 3.05.1, compatible with MySQL 8.0.32	Aurora MySQL version 3.05.1 is available. This version is compatible with MySQL 8.0.32. For full details, see Aurora MySQL version 3 compatible with MySQL 8.0.	November 21, 2023
Aurora MySQL version 3.04.1, compatible with MySQL 8.0.28	Aurora MySQL version 3.04.1 is available. This version is compatible with MySQL 8.0.28. For full details, see Aurora MySQL version 3 compatible with MySQL 8.0.	November 13, 2023
Aurora MySQL version 3.05.0.1, compatible with MySQL 8.0.32, Beta	Aurora MySQL version 3.05.0.1 is available. This version is compatible with MySQL 8.0.32. For full details, see Aurora MySQL version 3 compatible with MySQL 8.0.	October 30, 2023
Aurora MySQL version 3.05.0, compatible with MySQL 8.0.32	Aurora MySQL version 3.05.0 is available. This version is compatible with MySQL 8.0.32. For full details, see Aurora MySQL version 3 compatible with MySQL 8.0.	October 25, 2023
Aurora MySQL version 2.12.0.1, compatible with MySQL 5.7.40, Beta	Aurora MySQL version 2.12.0.1 Beta is available. This version is compatible with MySQL 5.7.40. For full details, see Aurora MySQL version 2 compatible with MySQL 5.7.	October 25, 2023

Aurora MySQL version 2.11.4, compatible with MySQL 5.7.12	Aurora MySQL version 2.11.4 is available. This version is compatible with MySQL 5.7.12. For full details, see Aurora MySQL version 2 compatible with MySQL 5.7.	October 17, 2023
Aurora MySQL version 3.03.2, compatible with MySQL 8.0.26	Aurora MySQL version 3.03.2 is available. This version is compatible with MySQL 8.0.26. For full details, see Aurora MySQL version 3 compatible with MySQL 8.0.	August 29, 2023
Aurora MySQL version 2.07.10, compatible with MySQL 5.7.12	Aurora MySQL version 2.07.10 is available. This version is compatible with MySQL 5.7.12. For full details, see Aurora MySQL version 2 compatible with MySQL 5.7.	August 15, 2023
Aurora MySQL version 3.04.0, compatible with MySQL 8.0.28	Aurora MySQL version 3.04.0 is available. This version is compatible with MySQL 8.0.28. For full details, see Aurora MySQL version 3 compatible with MySQL 8.0.	July 31, 2023
Aurora MySQL version 2.12.0, compatible with MySQL 5.7.40	Aurora MySQL version 2.12.0 is available. This version is compatible with MySQL 5.7.40. For full details, see Aurora MySQL version 2 compatible with MySQL 5.7.	July 25, 2023

Aurora MySQL version 2.11.3, compatible with MySQL 5.7.12	Aurora MySQL version 2.11.3 is available. This version is compatible with MySQL 5.7.12. For full details, see Aurora MySQL version 2 compatible with MySQL 5.7.	June 9, 2023
Aurora MySQL version 3.03.1, compatible with MySQL 8.0.26	Aurora MySQL version 3.03.1 is available. This version is compatible with MySQL 8.0.26. For full details, see Aurora MySQL version 3 compatible with MySQL 8.0.	May 11, 2023
Aurora MySQL version 2.07.9, compatible with MySQL 5.7.12.	Aurora MySQL version 2.07.9 is available. This version is compatible with MySQL 5.7.12. For full details, see Aurora MySQL version 2 compatible with MySQL 5.7.	May 4, 2023
Aurora MySQL version 3.02.3, compatible with MySQL 8.0.23	Aurora MySQL version 3.02.3 is available. This version is compatible with MySQL 8.0.23. For full details, see Aurora MySQL version 3 compatible with MySQL 8.0.	April 17, 2023
Aurora MySQL version 2.11.2, compatible with MySQL 5.7.12	Aurora MySQL version 2.11.2 is available. This version is compatible with MySQL 5.7.12. For full details, see Aurora MySQL version 2 compatible with MySQL 5.7.	March 24, 2023

Aurora MySQL version 3.03.0, compatible with MySQL 8.0.26	Aurora MySQL version 3.03.0 is available. This version is compatible with MySQL 8.0.26. For full details, see Aurora MySQL version 3 compatible with MySQL 8.0.	March 1, 2023
Aurora MySQL version 2.11.1, compatible with MySQL 5.7.12.	Aurora MySQL version 2.11.1 is available. This version is compatible with MySQL 5.7.12. For full details, see Aurora MySQL version 2 compatible with MySQL 5.7.	February 14, 2023
Aurora MySQL version 3.02.2, compatible with MySQL 8.0.23	Aurora MySQL version 3.02.2 is available. This version is compatible with MySQL 8.0.23. For full details, see Aurora MySQL version 3 compatible with MySQL 8.0.	November 18, 2022
Aurora MySQL version 2.10.3, compatible with MySQL 5.7	Aurora MySQL version 2.10.3 is available. This version is compatible with MySQL 5.7. For full details, see Aurora MySQL version 2 compatible with MySQL 5.7.	November 1, 2022
Aurora MySQL version 2.11.0, compatible with MySQL 5.7.12	Aurora MySQL version 2.11.0 is available. This version is compatible with MySQL 5.7.12. For full details, see Aurora MySQL version 2 compatible with MySQL 5.7.	October 25, 2022

Aurora MySQL version 3.02.1, compatible with MySQL 8.0.23	Aurora MySQL version 3.02.1 is available. This version is compatible with MySQL 8.0.23. For full details, see Aurora MySQL version 3 compatible with MySQL 8.0.	September 7, 2022
In-place upgrade for MySQL 5.6-compatible Aurora Serverless v1	You can perform an in-place upgrade for a MySQL 5.6-compatible Aurora Serverless v1 cluster to change an existing cluster into a MySQL 5.7-compatible Aurora Serverless v1 cluster. For more information, see Aurora MySQL Serverless 5.7 engine updates 2020-06-18 (version 2.07.1).	June 16, 2022
In-place upgrade for MySQL 5.6-compatible Aurora Serverless v1	You can perform an in-place upgrade for a MySQL 5.6-compatible Aurora Serverless v1 cluster to change an existing cluster into a MySQL 5.7-compatible Aurora Serverless v1 cluster. For more information, see Aurora MySQL Serverless 5.7 engine updates 2020-06-18 (version 2.07.1).	June 16, 2022
Aurora MySQL database engine updates 2022-06-16 (version 2.07.8) is available.	Aurora MySQL version 2.07.8 is available.	June 16, 2022

Aurora MySQL version 3.02.0, compatible with MySQL 8.0.23	Aurora MySQL version 3.02.0 is available. This version is compatible with MySQL 8.0.23. For full details, see Aurora MySQL version 3 compatible with MySQL 8.0.	April 20, 2022
Aurora MySQL version 3.01.1, compatible with MySQL 8.0.23	Aurora MySQL version 3.01.1 is available. This version is compatible with MySQL 8.0.23. For full details, see Aurora MySQL version 3 compatible with MySQL 8.0.	April 15, 2022
Initial release	Initial release of the *Aurora MySQL Release Notes*.	March 22, 2022
Aurora MySQL version 2.10.2	Aurora MySQL version 2.10.2 is available.	January 26, 2022
Aurora MySQL version 2.08.4	Aurora MySQL version 2.08.4 is available.	January 6, 2022
Aurora MySQL version 2.07.7	Aurora MySQL version 2.07.7 is available.	November 24, 2021
Aurora MySQL version 3.01.0, compatible with MySQL 8.0.23	Aurora MySQL version 3.01.0 is available. This version is compatible with MySQL 8.0.23. For full details, see Aurora MySQL version 3 compatible with MySQL 8.0.	November 18, 2021
Aurora MySQL version 2.09.3	Aurora MySQL version 2.09.3 is available.	November 12, 2021
Aurora MySQL version 2.10.1	Aurora MySQL version 2.10.1 is available.	October 21, 2021

Aurora MySQL version 1.23.4	Aurora MySQL version 1.23.4 is available.	September 30, 2021
Aurora MySQL version 2.07.6	Aurora MySQL version 2.07.6 is available.	September 2, 2021
Aurora MySQL version 2.07.5	Aurora MySQL version 2.07.5 is available.	July 6, 2021
Aurora MySQL version 1.23.3	Aurora MySQL version 1.23.3 is available.	June 28, 2021
Aurora MySQL version 1.22.5	Aurora MySQL version 1.22.5 is available.	June 3, 2021
Aurora MySQL version 2.10.0	Aurora MySQL version 2.10.0 is available. Some of the highlights include higher availability of reader instances during writer restarts, improvements to zero-downtime patching (ZDP), improvements to zero-down time restart (ZDR), and the binlog I/O cache optimizat ion.	May 25, 2021
Aurora MySQL version 1.23.2	Aurora MySQL version 1.23.2 is available.	March 18, 2021
Aurora MySQL version 2.07.4	Aurora MySQL version 2.07.4 is available.	March 4, 2021
Aurora MySQL version 1.22.4	Aurora MySQL version 1.22.4 is available.	March 4, 2021
Aurora MySQL version 2.09.2	Aurora MySQL version 2.09.2 is available.	February 26, 2021

Aurora MySQL version 2.09.1	Aurora MySQL version 2.09.1 is available.	December 11, 2020
Aurora MySQL version 1.23.1	Aurora MySQL version 1.23.1 is available.	November 24, 2020
Aurora MySQL version 2.08.3	Aurora MySQL version 2.08.3 is available.	November 12, 2020
Aurora MySQL version 2.07.3	Aurora MySQL version 2.07.3 is available.	November 10, 2020
Aurora MySQL version 1.22.3	Aurora MySQL version 1.22.3 is available.	November 9, 2020
Aurora MySQL version 2.09.0	Aurora MySQL version 2.09.0 is available.	September 17, 2020
Aurora MySQL version 1.23.0	Aurora MySQL version 1.23.0 is available.	September 2, 2020
Aurora MySQL version 2.08.2	Aurora MySQL version 2.08.2 is available.	August 28, 2020
Aurora MySQL version 2.04.9	Aurora MySQL version 2.04.9 is available.	August 14, 2020
Aurora MySQL version 2.08.1	Aurora MySQL version 2.08.1 is available.	June 18, 2020
Aurora MySQL version 1.22.2 for parallel query clusters	Aurora MySQL version 1.22.2 is available when you create a parallel query cluster.	June 18, 2020
Aurora MySQL version 1.20.1 for parallel query clusters	Aurora MySQL version 1.20.1 is available when you create a parallel query cluster.	June 11, 2020
Aurora MySQL version 2.08.0	Aurora MySQL version 2.08.0 is available.	June 2, 2020

Aurora MySQL version 1.19.6 for parallel query clusters	Aurora MySQL version 1.19.6 is available when you create a parallel query cluster.	June 2, 2020
Aurora MySQL version 2.07.2	Aurora MySQL version 2.07.2 is available.	April 17, 2020
Aurora MySQL version 1.22.2	Aurora MySQL version 1.22.2 is available.	March 5, 2020
Aurora MySQL version 1.20.1	Aurora MySQL version 1.20.1 is available.	March 5, 2020
Aurora MySQL version 1.19.6	Aurora MySQL version 1.19.6 is available.	March 5, 2020
Aurora MySQL version 1.17.9	Aurora MySQL version 1.17.9 is available.	March 5, 2020
Aurora MySQL version 2.07.1	Aurora MySQL version 2.07.1 is available.	December 23, 2019
Aurora MySQL version 1.22.1	Aurora MySQL version 1.22.1 is available.	December 23, 2019
Aurora MySQL version 2.07.0	Aurora MySQL version 2.07.0 is available.	November 25, 2019
Aurora MySQL version 1.22.0	Aurora MySQL version 1.22.0 is available.	November 25, 2019
Aurora MySQL version 1.21.0	Aurora MySQL version 1.21.0 is available.	November 25, 2019
Aurora MySQL version 2.06.0	Aurora MySQL version 2.06.0 is available.	November 22, 2019
Aurora MySQL version 2.04.8	Aurora MySQL version 2.04.8 is available.	November 20, 2019

Aurora MySQL version 2.04.7	Aurora MySQL version 2.04.7 is available.	November 14, 2019
Aurora MySQL version 2.05.0	Aurora MySQL version 2.05.0 is available.	November 11, 2019
Aurora MySQL version 1.20.0	Aurora MySQL version 1.20.0 is available.	November 11, 2019
Aurora MySQL version 2.04.6	Aurora MySQL version 2.04.6 is available.	September 19, 2019
Aurora MySQL version 1.19.5	Aurora MySQL version 1.19.5 is available.	September 19, 2019
Aurora MySQL version 2.04.5	Aurora MySQL version 2.04.5 is available.	July 8, 2019
Aurora MySQL version 1.19.2	Aurora MySQL version 1.19.2 is available.	June 5, 2019
Aurora MySQL version 2.04.4	Aurora MySQL version 2.04.4 is available.	May 29, 2019
Aurora MySQL version 2.04.3	Aurora MySQL version 2.04.3 is available.	May 9, 2019
Aurora MySQL version 1.19.1	Aurora MySQL version 1.19.1 is available.	May 9, 2019
Aurora MySQL version 2.04.2	Aurora MySQL version 2.04.2 is available.	May 2, 2019
Aurora MySQL version 2.04.1	Aurora MySQL version 2.04.1 is available.	March 25, 2019
Aurora MySQL version 2.04	Aurora MySQL version 2.04 is available.	March 25, 2019

Aurora MySQL version 2.03.4	Aurora MySQL version 2.03.4 is available.	February 7, 2019
Aurora MySQL version 1.19.0	Aurora MySQL version 1.19.0 is available.	February 7, 2019
Aurora MySQL version 2.03.3	Aurora MySQL version 2.03.3 is available.	January 18, 2019
Aurora MySQL version 1.17.8	Aurora MySQL version 1.17.8 is available.	January 17, 2019
Aurora MySQL version 2.03.2	Aurora MySQL version 2.03.2 is available.	January 9, 2019
Aurora MySQL version 2.03.1	Aurora MySQL version 2.03.1 is available.	October 24, 2018
Aurora MySQL version 2.03	Aurora MySQL version 2.03 is available.	October 11, 2018
Aurora MySQL version 2.02.5	Aurora MySQL version 2.02.5 is available.	October 8, 2018
Aurora MySQL version 1.17.7	Aurora MySQL version 1.17.7 is available.	October 8, 2018
Aurora MySQL version 2.02.4	Aurora MySQL version 2.02.4 is available.	September 21, 2018
Aurora MySQL version 1.18.0	Aurora MySQL version 1.18.0 is available.	September 20, 2018
Aurora MySQL version 1.17.6	Aurora MySQL version 1.17.6 is available.	September 6, 2018