PlanetScale Navigating MySQL 5.7 End of Life Instructions

PlanetScale Navigating MySQL 5.7 End of Life Instructions

PlanetScale

With MySQL 5.7 EOL comes the end of:

- Security updates — putting your business at risk
- Technical support and reliability
- Compatibility with newer technology
- PCI DSS, GDPR, HIPAA, or SOX compliance

Running on EOL software puts your company at risk of being out of compliance with software security standards and the best practices used in your development environment. This can lead to your company operating out of PCI compliance and to performance issues that may impact customer-facing workloads.

What's more, if you don't plan for the upgrade in advance, forced upgrades to new versions of MySQL can lead to unintended downtime which pose financial and reputational detriments to your company.

On top of the risk around version upgrades, there are high operational costs associated with maintaining and debugging EOL software. The longer EOL software runs, the more demand your team will have for support as knowledge and technical support for the version decreases. As demand for support increases, maintenance costs increase in parallel with the risk of a security breach or downtime. This cost is incredibly impactful with the cost of downtime averaging about $300,000 per hour.*

If you're running on MySQL 5.7, now is the time to consider a pathway to upgrade with minimal disruption, minimal risk, and zero downtime.

Migration

Software best practices are to update as frequently as possible, but there are serious risks associated with updating on a time pressure. The time and effort it takes to pull off a major upgrade will drain internal engineering resources, and the risk associated with timing, security, and compliance requirements can seriously impact your company.

On top of this, many legacy providers and managed database solutions — including AWS Aurora and RDS — are becoming increasingly vocal about the downtime required to complete the version upgrade with their solution. Amazon RDS for MySQL will stop supporting the creation of new MySQL 5.7 instances starting October 2023 via both the AWS Management Console and AWS Command Line Interface. Amazon Aurora 5.7 will go end of life in October 2024 due to some Aurora-specific features being incompatible to 8.0.

Database engine upgrades require downtime.

The duration of the downtime varies based on the size of your database instance.

If your MySQL 5.7 database instance is using read replicas, you must upgrade all of the read replicas before upgrading the source instance. If your database instance is in a Multi-AZ deployment, both the primary and standby replicas are upgraded. Your database instance will not be available until the upgrade is complete.

If you don't plan for this upgrade, your database vendor may force an update. When a major engine version upgrade is forced, it can introduce changes that aren't backward-compatible with existing applications

What are your options to migrate?

1. Upgrade to 8.0 in your current environment — a timely, complex, and risky migration likely involving manual work and downtime.
2. Migrate to a new environment where you can run on completely updated versions of MySQL.

MySQL 5.7 and 8.0 incompatibilities
MySQL 8.0 includes a number of incompatibilities with MySQL 5.7. These incompatibilities can cause problems during an upgrade from MySQL 5.7 to MySQL 8.0.

If you choose to migrate on your own, you'll need to consider the following list of incompatibilities. You cannot have:

1. Tables that use obsolete data types or functions
2. Orphan *.frm files
3. Triggers with a missing or empty definer or an invalid creation context (PlanetScale does not support triggers)
4. Partitioned table that uses a storage engine that does not have native partitioning support
5. Keyword or reserved word violations. Some keywords might be reserved in MySQL 8.0 that were not reserved previously†
6. Tables in the MySQL 5.7 mysql system database that have the same name as a table used by the MySQL 8.0 data dictionary
7. Obsolete SQL modes defined in your sql_mode system variable setting
8. Tables or stored procedures with individual ENUM or SET column elements that exceed 255 characters or 1020 bytes in length (PlanetScale does not support stored procedures)
9. Table partitions that reside in shared InnoDB tablespaces
10. Queries and stored program definitions from MySQL 8.0.12 or lower that use ASC or DESC qualifiers for GROUP BY clauses
11. Other features that are not supported in MySQL 8.0
12. FOREIGN KEY constraint names longer than 64 characters (PlanetScale does not support foreign key constraints)
13. For improved Unicode support, consider converting objects that use the utf8mb3 charset to use the utf8mb4 charset. The utf8mb3 character set is deprecated. Also, consider using utf8mb4 for character set references instead of utf8, because currently utf8 is an alias for the utf8mb3 charset.

Accounting for these incompatibilities and anticipating downtime, preparation will be required on your database for the upgrade to be successful.

One-click imports and zero downtime upgrades

With PlanetScale, you can migrate from your current database solution with one-click imports and without downtime. We will manage all version upgrades automatically for you so you won't need to worry about incompatibility issues or the security, reliability, or financial risks associated with version upgrades.

PlanetScale is built on top of open-source Vitess, a database clustering system for horizontal scaling of MySQL. Consequently, PlanetScale is only compatible with MySQL databases. The PlanetScale imports tool supports MySQL database versions 5.7 through 8.0. We are conscious about our MySQL compatibility, to learn more about this check out our documentation.*

With a migration to PlanetScale, you get the ease of mind knowing that you're running on the latest major version of MySQL:

- You do not need to worry about future upgrades
- Migrating to PlanetScale never requires downtime
- We provide dedicated support and database expertise
- You benefit from GitHub-style developer workflows, including branching, non-blocking schema changes, and more.

With the downtime required to pull off a version upgrade with solutions like AWS RDS, you would have less downtime by migrating out of AWS than attempting to upgrade to 8.0 in your current environment. The increased financial cost of running on EOL software, or the general cost of application downtime, can be a detriment to your company.

Migrating to PlanetScale can reduce your overall cost of migration and the management of your database

Trusted by

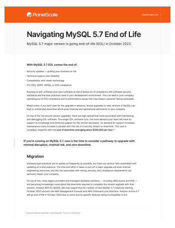
Get started today with PlanetScale,
the most reliable way to scale your
MySQL database in the cloud.
Call us at or
send an email to

1-408-214-1997
sales@planetscale.com

PlanetScale

Contents

Documents / Resources

 Navigating MySQL 5.7 End of Life	PlanetScale Navigating MySQL 5.7 End of Life [pdf] Instructions Navigating MySQL 5.7 End of Life

References

- PlanetScale: The world's most advanced database platform
- Contact us — PlanetScale
- G blogs.gartner.com/andrew-lerner/2014/07/16/the-cost-of-downtime/
- Upgrading the MySQL DB engine - Amazon Relational Database Service
- Contact us — PlanetScale
- MySQL compatibility — PlanetScale Documentation