# PERCONA

Databases run better with Percona

Bangkok, Thailand | March 11, 2026

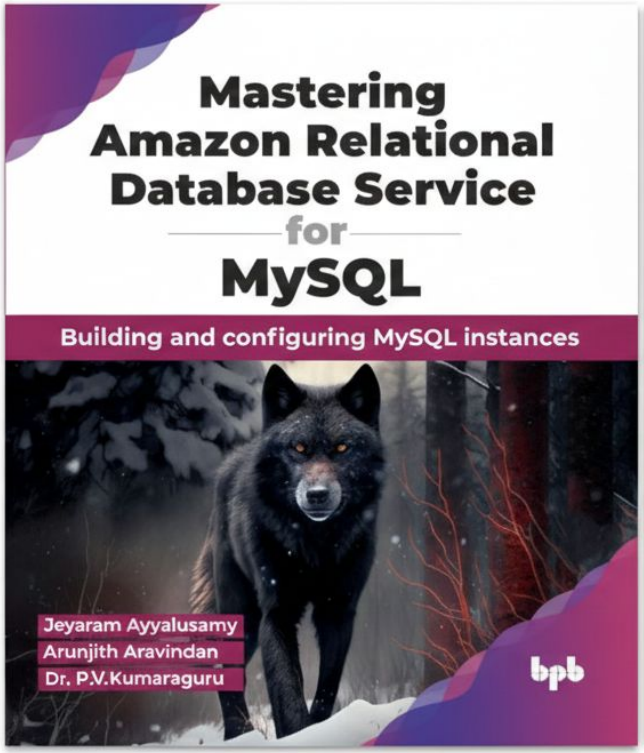
Arunjith Aravindan
Senior MySQL DBA at Percona

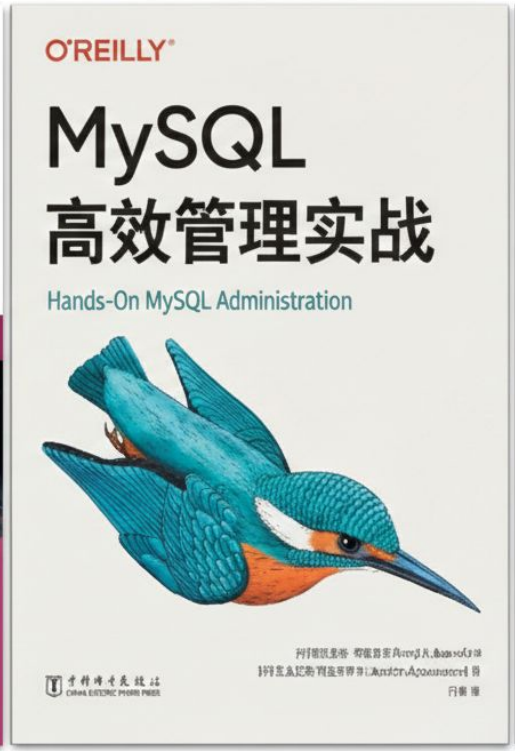
PERCONA UNIVERSITY

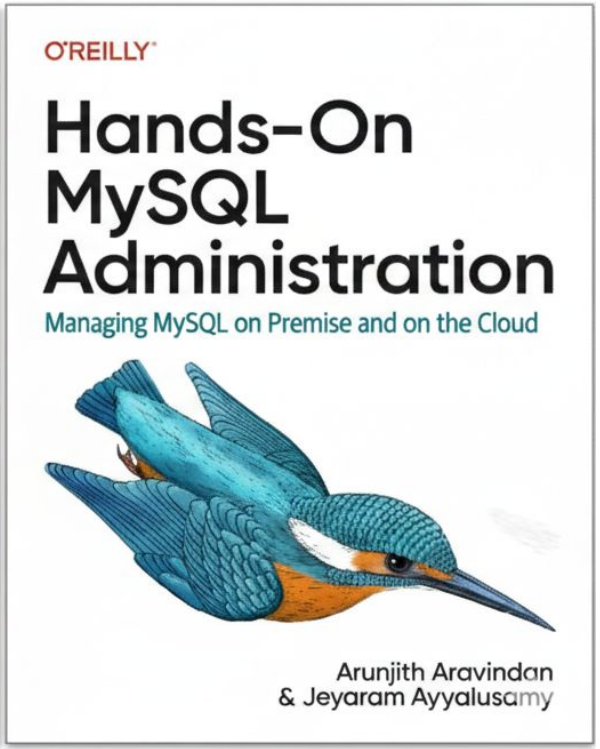
# MySQL in 2026: Lifecycle Realities, Upgrade Myths, and the Road to 9.7 LTS

# Bookauthority.org

Ranked #1 and #7 on the list of new MySQL books Redefining Database Expertise in 2025!



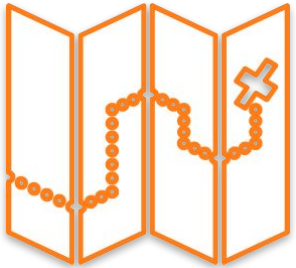



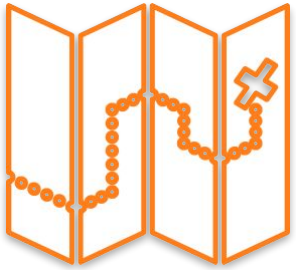

PERCONA

# Agenda

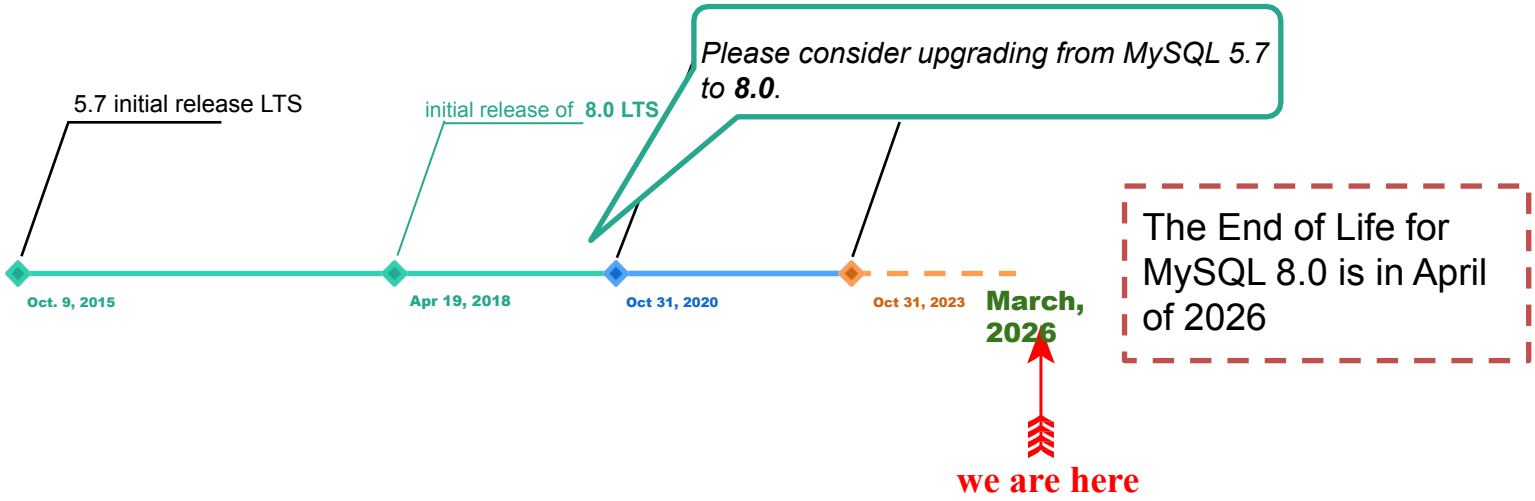
- Where are we in the MySQL lifecycle?
- Where are MySQL users today?
- Why do many MySQL instances still run older versions?
- Upgrading from MySQL 5.7 to 8.0 and 8.4: Facts or Urban Legends?
- Key changes in MySQL 8, 8.4, and 9!
- How to ease the major version upgrade! Tools which can help



# Where are we in the MySQL lifecycle?



# MySQL 5.7 & 8.0 lifecycle

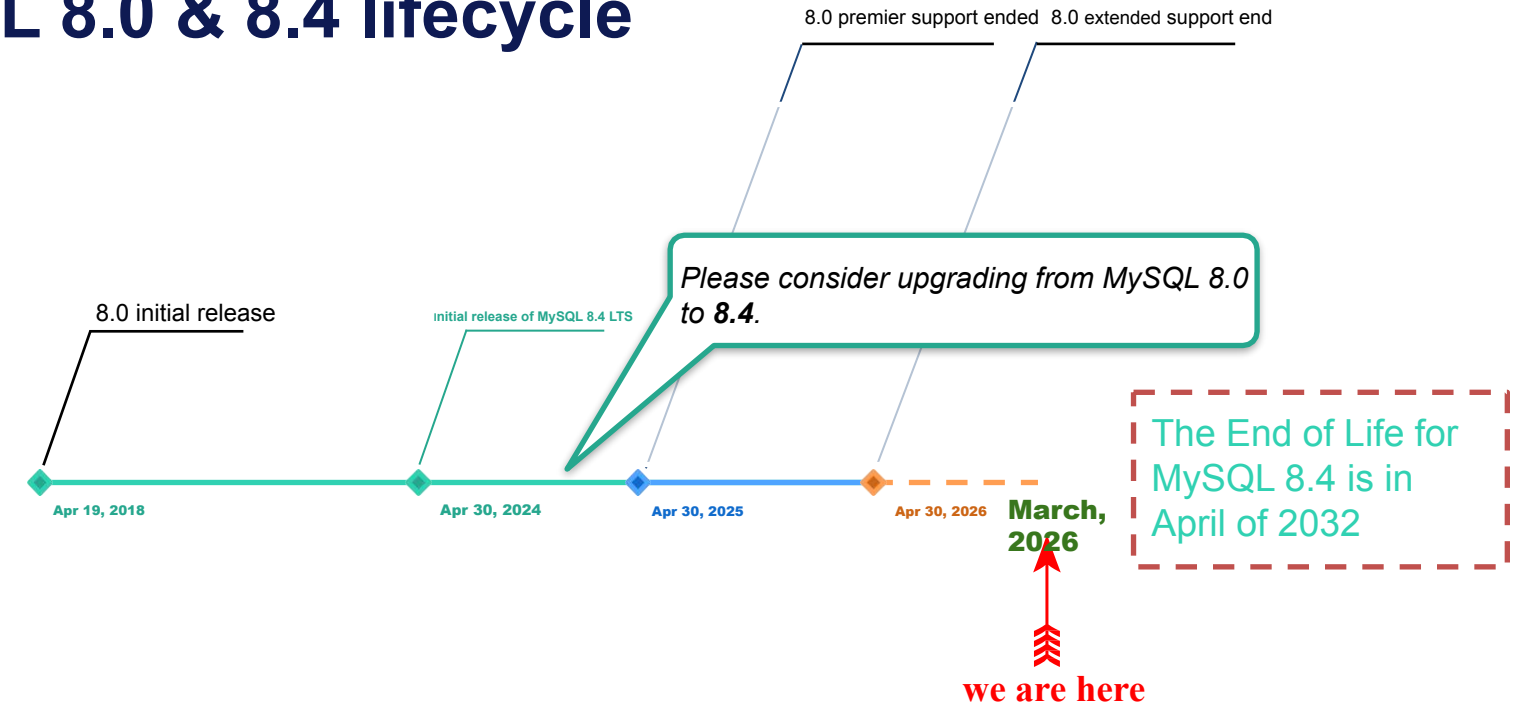



MySQL 8.4 is actively supported and ready for production use.



PERCONA

# MySQL 8.0 & 8.4 lifecycle

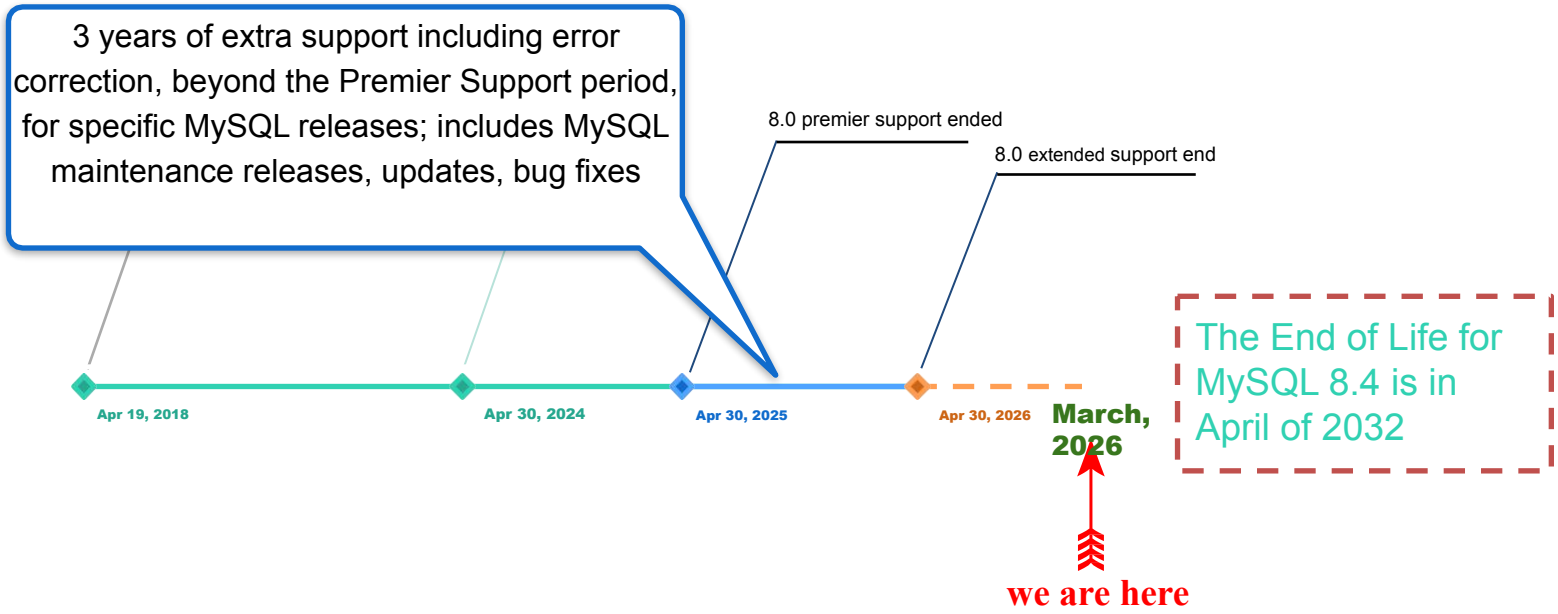



MySQL 8.4 is actively supported and ready for production use.



PERCONA

# MySQL 8.0 extra support from Percona




MySQL 8.4 is actively supported and ready for production use.
The next LTS release will be **MySQL 9.7.**



PERCONA

# MySQL 8.4: Long Term Support (LTS) Release

**Release Date:** April 2024

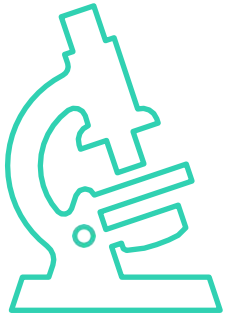
**LTS Designation:** MySQL 8.4.0 is designated as a Long Term Support (LTS) release.

**Support Duration:**

- Premier Support: 5 years
- Extended Support: 3 additional years
- Total Extended Support End: Approximately April 2032.

- The LTS release is the final minor version within its major release series.
- **8.4 is the 8.x LTS, then the next major release will be MySQL 9.7.**

# Why do many MySQL instances still run older versions?

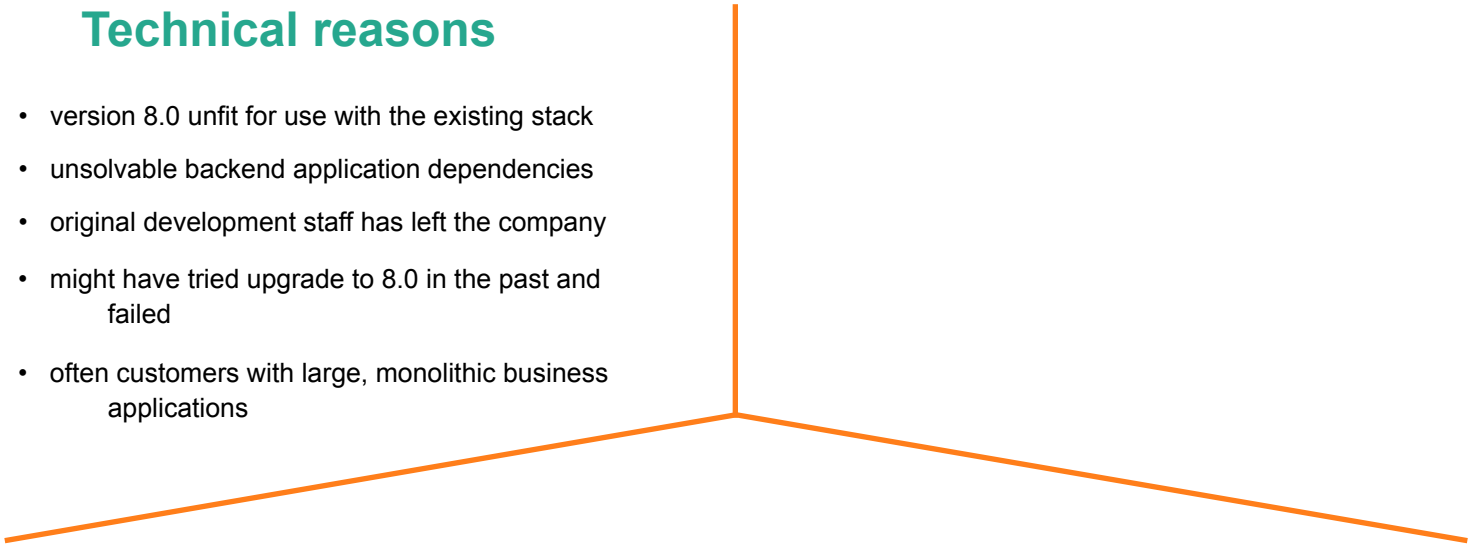


# What prevents you from upgrading to MySQL 8.0 / 8,4?

## answers from Percona customers

### Technical reasons

- version 8.0 unfit for use with the existing stack
- unsolvable backend application dependencies
- original development staff has left the company
- might have tried upgrade to 8.0 in the past and failed
- often customers with large, monolithic business applications



# What prevents you from upgrading to MySQL 8.0?

## answers from Percona customers

### Technical reasons

- version 8.0 unfit for use with the existing stack
- unsolvable backend application dependencies
- original development staff has left the company
- might have tried upgrade to 8.0 in the past and failed
- often customers with large, monolithic business applications

### Perception of cost and risk

- have read about the horrors of the upgrade
- are afraid of the unnamed risks and issues
- understand that the cost of software and services' maintenance is non-zero

# What prevents you from upgrading to MySQL 8.0?

## answers from Percona customers

### Technical reasons

- version 8.0 unfit for use with the existing stack
- unsolvable backend application dependencies
- original development staff has left the company
- might have tried upgrade to 8.0 in the past and failed
- often customers with large, monolithic business applications

### Perception of cost and risk

- have read about the horrors of the upgrade
- are afraid of the unnamed risks and issues
- understand that the cost of software and services' maintenance is non-zero

### See no evil, hear no evil

- “let’s not fix things which ain’t broken”
- “let’s pretend this problem does not exist”



PERCONA

# Upgrading from MySQL 5.7 to 8.0 and 8.4: Facts or Urban Legends?

# Upgrading MySQL from 5.7 to 8.0 and 8.4

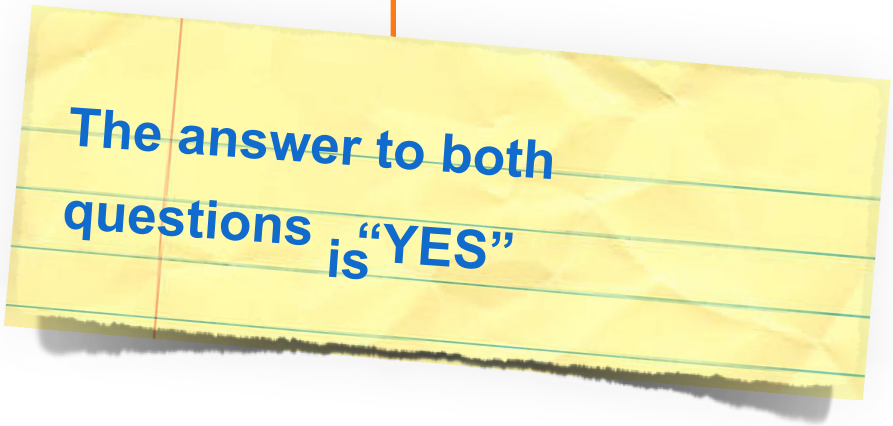
**Are there any real differences between MySQL 5.7 and 8.0 and 8.4?**

**Where are the missing 6.x and 7.x releases?**

# Upgrading MySQL from 5.7 to 8.0 and 8.4

**Are there any real differences between MySQL 5.7 and 8.0?**

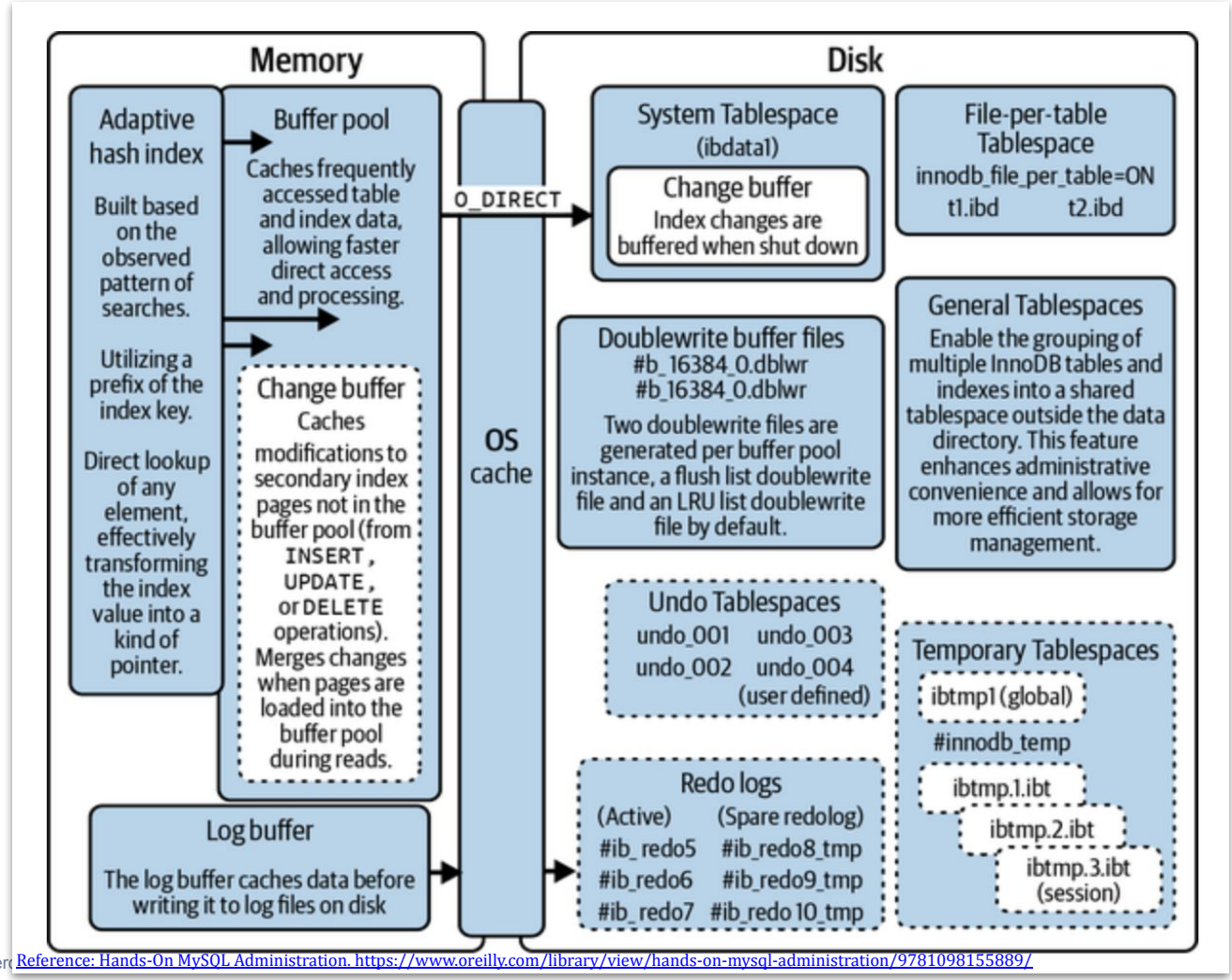
**Where are the missing 6.x and 7.x releases?**



# Review of key changes

- **Major new capabilities and changes incorporated**
  - General changes
  - InnoDB changes
  - Security and Account Management changes
- **Deprecations**
- **Removals**
- **Bonus upgrade impediment**

This is what can truly impact the upgrade process.




Reference: Hands-On MySQL Administration. https://www.oreilly.com/library/view/hands-on-mysql-administration/9781098155889/

# General changes in MySQL 8.4 vs. 8.0

**Full removal of MASTER and SLAVE:** You must use SOURCE and REPLICA. Commands like START SLAVE now throw errors.

Ex: CHANGE REPLICATION SOURCE TO..

**Unique Foreign Keys:** Stricter enforcement. 8.4 now requires a unique index on the parent table for foreign keys (matching the SQL standard).

**FLOAT and DOUBLE AUTO_INCREMENT Removal:** AUTO_INCREMENT is completely removed for FLOAT and DOUBLE types. Schema must be updated to INT/BIGINT before upgrade.

**New reserved words:** MANUAL, PARALLEL, QUALIFY, TABLESAMPLE. Unquoted columns with these names will break.

**Binlog Retention Change:** MySQL 8.4 has officially removed the legacy expire_logs_days variable; you must now use binlog_expire_logs_seconds for more granular control over binary log retention.

**Automatic Histograms:** Histograms can now auto-update during ANALYZE TABLE, keeping query plans fresh without manual intervention.

Histograms are data maps that tell the database how many rows fall into different value ranges, helping it choose the fastest way to run your query.

In MySQL 8.0, histograms are static after creation and can become outdated as data changes, requiring manual ANALYZE TABLE ... UPDATE HISTOGRAM to avoid suboptimal query plans.

**Dropped Server Options:** The --ssl and --admin-ssl server options, as well as the have_ssl and have_openssl server system variables, were deprecated in MySQL 8.0.26. They are all removed in this release. Use --tls-version and --admin-tls-version instead.

Reference : https://dev.mysql.com/doc/refman/8.4/en/mysql-nutshell.html

# InnoDB changes in MySQL 8.4 vs. 8.0

**InnoDB Buffer Pool Instances:** In MySQL 8.4, innodb_buffer_pool_instances is auto-calculated based on your hardware (CPU cores and RAM) instead of using the old fixed default of 8, ensuring better performance out-of-the-box.

If innodb_buffer_pool_size <= 1 GiB, then 1
If innodb_buffer_pool_size > 1 GiB, then this is the minimum value from the following two calculated hints in the range of 1-64:
Buffer pool hint: Calculated as 1/2 of (innodb_buffer_pool_size / innodb_buffer_pool_chunk_size)
CPU hint: Calculated as 1/4 of the number of available logical processors

**Change Buffering Disabled**: The default for innodb_change_buffering is now none (previously all). On modern SSDs, the overhead of managing the change buffer often outweighs the performance gain of merging writes.

**InnoDB Adaptive Hash Index:** In MySQL 8.4, the innodb_adaptive_hash_index default has been changed to OFF (it was ON in MySQL 8.0) to reduce mutex contention and improve stability on modern SSD-based hardware.

**InnoDB Flush Method:** Shifted from fsync to O_DIRECT as the default on Linux systems (where supported), to bypass the OS cache, preventing "double buffering" and improving I/O efficiency for data files.

**Higher I/O Capacity:** The default innodb_io_capacity jumped from 200 to 10,000, and innodb_io_capacity_max 2 * innodb_io_capacity. This allows InnoDB to flush dirty pages much faster on SSDs.

**Log Buffer Growth:** The default innodb_log_buffer_size increased from 16 MB to 64 MB, reducing disk I/O for high-concurrency writes.

Reference : https://dev.mysql.com/doc/refman/8.4/en/mysql-nutshell.html

# Security and account management changes in MySQL 8.4 vs. 8.0

**Authentication Plugin:** The mysql_native_password plugin is disabled by default.

- Effect: Legacy clients that cannot handle caching_sha2_password (the 8.0/8.4 default) will fail to connect.
- Workaround: You can re-enable it manually, but it is officially deprecated and slated for removal in MySQL 9.0.

**Granular Privileges:** A new FLUSH_PRIVILEGES privilege was introduced. Previously, you needed broad SUPER or RELOAD rights; now you can grant this specific task to a service account.

Granting only the FLUSH_PRIVILEGES privilege allows a user to run FLUSH PRIVILEGES and nothing else; they cannot run FLUSH TABLES or FLUSH LOGS.

**Definer Rights:** New privileges SET_ANY_DEFINER and ALLOW_NONEXISTENT_DEFINER give DBAs better control over who can create views and stored procedures without needing the "Root" user.

**GTID Tagging:** You can now "tag" specific transactions with a string (UUID:TAG:NUMBER). This allows for easier auditing and filtering of replication streams (e.g., tagging all transactions coming from a specific "Order-Entry" service).

## Innovation Releases

# General / InnoDB changes in MySQL 9 vs. 8.4

**Native VECTOR Data Type:** Introduced in 9.0. Specifically for AI/LLM embeddings. It stores up to 16,383 single-precision floats and includes functions like DISTANCE(), VECTOR_DIM(), and STRING_TO_VECTOR().

mysql> CREATE TABLE v1 (c1 VECTOR(5000));
Query OK, 0 rows affected (0.03 sec)

**Autopilot Index Advisor:** (HeatWave/Cloud-focused) Helps optimize OLTP workloads on MySQL 9.0+ by automatically analyzing real query patterns from Performance Schema and recommending secondary indexes to create or drop. It balances read performance gains against write overhead, providing actionable DDL.

**Saving JSON output from EXPLAIN ANALYZE INTO user variable:**
Support is now provided for saving JSON output from EXPLAIN ANALYZE into a user variable, using the syntax shown here:

```
EXPLAIN ANALYZE FORMAT=JSON
INTO @plan
SELECT * FROM orders WHERE customer_id = 123;
Later, you can inspect it: SELECT JSON_PRETTY(@plan);
```

**JSON duality views:** MySQL 9.4.0 adds support for JSON duality views, providing a way to view data stored in one or more relational tables as a JSON document. JSON duality views can be created, altered, dropped, and viewed using the new CREATE JSON DUALITY VIEW and ALTER JSON DUALITY VIEW statements, along with the existing DROP VIEW and SHOW CREATE VIEW statements, which now work with both JSON duality views and SQL views.

```
CREATE TABLE customers (id INT PRIMARY KEY, name VARCHAR(100)); CREATE TABLE orders (id INT PRIMARY KEY, customer_id INT, amount DECIMAL(10,2));

CREATE JSON DUALITY VIEW customer_json_view AS
SELECT
  c.id,
  c.name,
  (
    SELECT JSON_ARRAYAGG(
      JSON_OBJECT(
        'order_id', o.id,
        'amount', o.amount
      )
    )
    FROM orders o
    WHERE o.customer_id = c.id
  ) AS orders
FROM customers c;

SELECT * FROM customer_json_view;

{
  "id": 1,
  "name": "Alice",
  "orders": [
    { "order_id": 101, "amount": 250.00 },
    { "order_id": 102, "amount": 99.50 }
  ]
}
```

PERCONA

## Innovation Releases

# General / InnoDB changes in MySQL 9 vs. 8.4

**JavaScript Stored Programs:** (Enterprise/Cloud) MySQL 9 allows stored functions and procedures to be written in JavaScript using LANGUAGE JAVASCRIPT. JavaScript routines can be invoked directly from SQL and fully support MySQL data types (including DECIMAL/NUMERIC), input and output parameters, prepared-statement bindings, and return values—enabling precise business logic inside the database.

Percona Server 8.4 :js_lang stored procedure and function overview feature is currently a Tech Preview

**DECIMAL Support in JavaScript Stored Programs:**

MySQL 9.3.0 introduced full support for the DECIMAL (NUMERIC) data type in JavaScript stored programs. DECIMAL values can now be used seamlessly in JavaScript as:
Input arguments
Output arguments
bind() parameters in prepared statements
Return values
This enables precise numeric computations in JavaScript routines without precision loss.

**replica_parallel_workers changes:** As of the MySQL 9.3.0 release, the replica_parallel_workers system variable can no longer be set to 0; the minimum permitted value is now 1.

**mysql client --commands option**: By default, mysql client commands (like \!, system, source) are disabled in MySQL 9.4.0 and later.

```
mysql -u root -p

Inside the mysql client:
mysql> \! ls
ERROR: mysql client commands are disabled.

Enable client commands explicitly

mysql --commands -u root -p
or
mysql --commands=ON -u root -p
Now client commands work:

mysql> \! ls
mysql> system uptime
mysql> source init.sql
```

PERCONA

## Innovation Releases

# Security and account management changes in MySQL 9 vs. 8.4

**Removal of mysql_native_password:** While 8.4 merely disables it by default, 9.0 completely removes the plugin. If your app hasn't migrated to caching_sha2_password by the time you hit version 9.0, it will be physically unable to connect.

**Logical Dumps of Users:** 9.3+ introduced a massive improvement to mysqldump with the --users option. It can now generate CREATE USER and GRANT statements directly into the dump file.

To cause the CREATE USER statements to be preceded by DROP USER, include the --add-drop-user option as well.

**New Granular Privileges**(MySQL 9)**:**

**CREATE_SPATIAL_REFERENCE_SYSTEM:** Allows managing Spatial Reference System (SRS) without needing the dangerous SUPER privilege.

A Spatial Reference System (SRS) is a coordinate system used to define how geographic data (like maps, locations, and shapes) is positioned on Earth.

**SET_ANY_DEFINER:** Fully replaces old "root-only" requirements for creating views or procedures for other users.

**Connection Control Component:** The old plugins for limiting failed login attempts are replaced by the component_connection_control, which is more efficient and easier to configure.

**Potential upcoming feature :** PGO-optimized (Profile Guided Optimization) community binaries, vector functions,the hypergraph optimizer, and JSON duality improvements focused on DML usability. OpenTelemetry for modern observability, plus multi-threaded applier and flow-control metrics to enhance HA/DR analytics.
https://blogs.oracle.com/mysql/a-new-era-of-mysql-community-engagement-public-community-roadmap-webinar-highlights

# General changes in MySQL 8.0 vs. 5.7

- new transaction data dictionary storing information about database objects
- atomic DDL statement now combines data dictionary updates, storage engine operations and binary log writes associated with a DDL operation
- MySQL server now automatically runs all necessary upgrade tasks (system tables, objects in other schemas like sys schema and user schema) at the next startup, so no need to manually run "mySQL_upgrade" as of version 8.0.16
- SSL session reuse supported by default with the timeout setting
- MySQL server now supports for creation and management of resource groups and permits assigning threads to particular groups
- table encryption can be managed globally by defining and enforcing encryption defaults
- default character set is changed from "latin1" to "utf8mb4", which now has several new collations available
- added support for expressions as default values for the BLOB, TEXT, GEOMETRY and JSON data types
- new type of backup lock added, which permits DMLs during an online backup, while preventing operations which could result in an inconsistent snapshot
- TCP/IP port can be now configured specifically for administrative connections, even if "max_connections" level is already reached on the primary port
- added support for invisible indexes, which are not used by the optimiser and make it possible to test the effect of removing an index without actually removing it
- added support for document store, for developing both SQL and NoSQL applications with a single DB engine
- it's now possible to persist global, dynamic server variables using the SET PERSIST command instead of SET GLOBAL. (`mysqld-auto.cnf`)

# InnoDB changes in MySQL 8.0 vs. 5.7

- the maximum auto-increment counter value is now persistent across the server restarts
- in an event of index tree corruption, InnoDB writes a corruption flag to the redo log, which makes the corruption flag crash-safe
- new dynamic variable “innodb_deadlock_detect” may be used to disable deadlock detection
- InnoDB temporary tables are now created in session temporary tablespaces (*.ibt files)
- system tables and data dictionary tables are now created in a single InnoDB tablespace names mysql.ibd, in the MySQL data directory
- by default undo-logs now reside in two undo tablespaces that are created when the MySQL instance is initialised; undo logs are no longer created in the system tablespace
- new “innodb_dedicated_server” variable (disabled by default) can be used to automatically configure the options based on the detected memory availability
- tablespace files can be moved or restored to a new location while the server is offline using the “innodb_directories” option

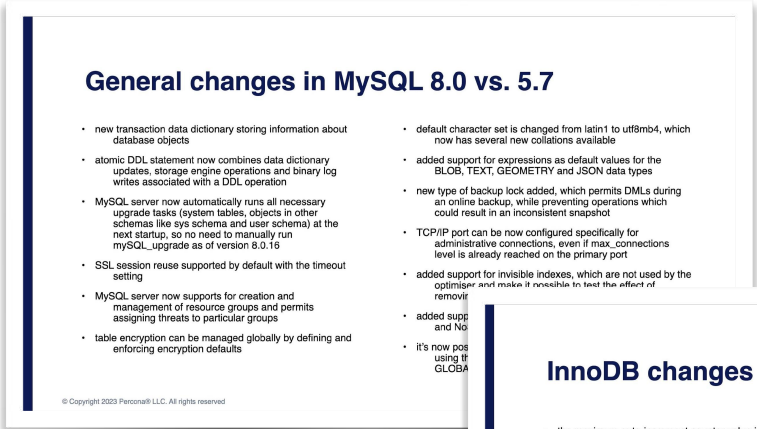
# Security and account management changes in MySQL 8.0 vs. 5.7

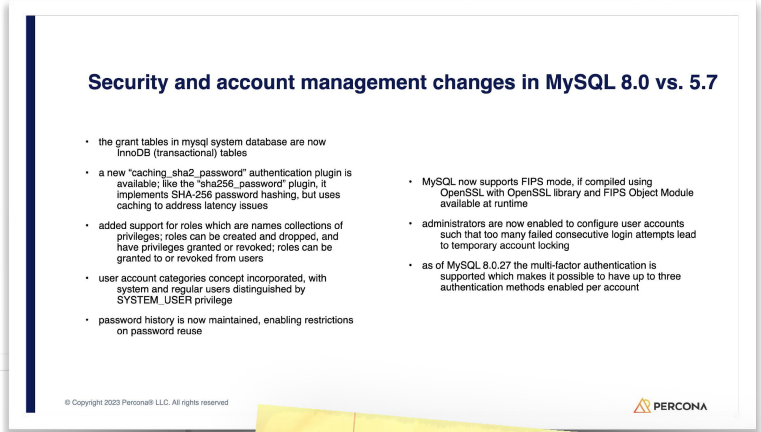
- the grant tables in mysql system database are now InnoDB (transactional) tables
- a new “caching_sha2_password” authentication plugin is available; like the “sha256_password” plugin, it implements SHA-256 password hashing, but uses caching to address latency issues
- added support for roles which are names collections of privileges; roles can be created and dropped, and have privileges granted or revoked; roles can be granted to or revoked from users
- user account categories concept incorporated, with system and regular users distinguished by SYSTEM_USER privilege
- password history is now maintained, enabling restrictions on password reuse

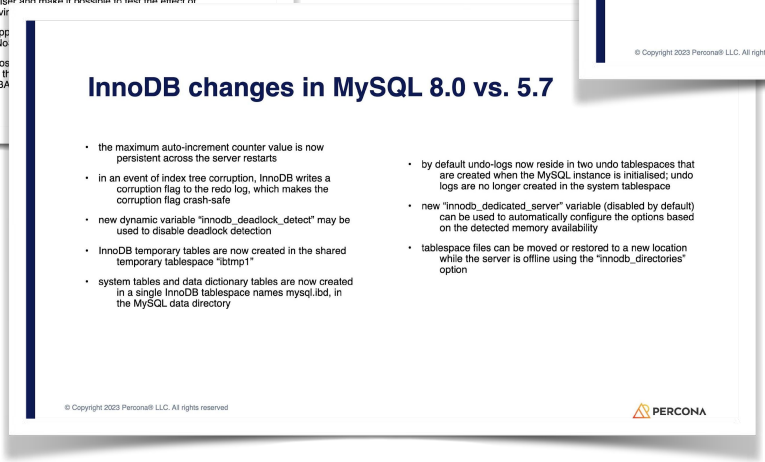
- MySQL now supports FIPS mode (Federal Information Processing Standards), if compiled using OpenSSL with OpenSSL library and FIPS Object Module available at runtime
- administrators are now enabled to configure user accounts such that too many failed consecutive login attempts lead to temporary account locking
- as of MySQL 8.0.27 the multi-factor authentication is supported which makes it possible to have up to three authentication methods enabled per account

somewhat outdated list of differences on MySQL website: https://dev.mysql.com/blog-archive/the-complete-list-of-new-features-in-mysql-8-0/

# New capabilities and changes. Already worried?




we are only getting started

PERCONA

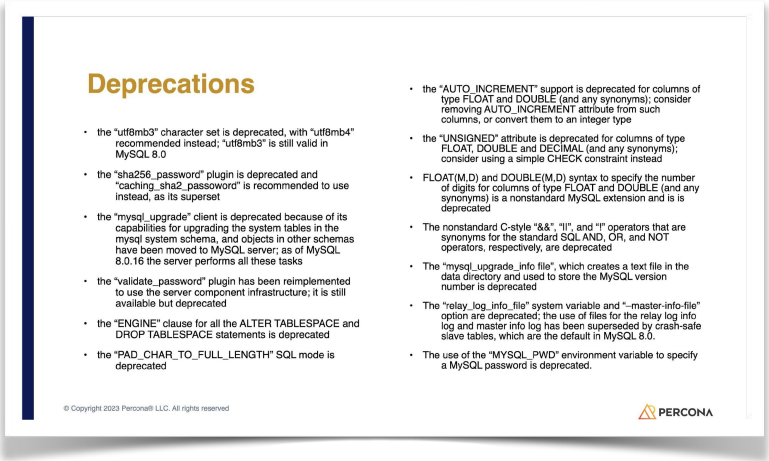
# Deprecations

- the “utf8mb3” character set is deprecated, with “utf8mb4” recommended instead; “utf8mb3” is still valid in MySQL 8.0
- the “sha256_password” plugin is deprecated and “caching_sha2_passoword” is recommended to use instead, as its superset
- the “mysql_upgrade” client is deprecated because of its capabilities for upgrading the system tables in the mysql system schema, and objects in other schemas have been moved to MySQL server; as of MySQL 8.0.16 the server performs all these tasks
- the “validate_password” plugin has been reimplemented to use the server component infrastructure; it is still available but deprecated
- the “ENGINE” clause for all the ALTER TABLESPACE and DROP TABLESPACE statements is deprecated
- the “PAD_CHAR_TO_FULL_LENGTH” SQL mode is deprecated
- the “AUTO_INCREMENT” support is deprecated for columns of type FLOAT and DOUBLE (and any synonyms); consider removing AUTO_INCREMENT attribute from such columns, or convert them to an integer type
- the “UNSIGNED” attribute is deprecated for columns of type FLOAT, DOUBLE and DECIMAL (and any synonyms); consider using a simple CHECK constraint instead
- FLOAT(M,D) and DOUBLE(M,D) syntax to specify the number of digits for columns of type FLOAT and DOUBLE (and any synonyms) is a nonstandard MySQL extension and is is deprecated
- The nonstandard C-style “&&”, “||”, and “!” operators that are synonyms for the standard SQL AND, OR, and NOT operators, respectively, are deprecated
- The “mysql_upgrade_info file”, which creates a text file in the data directory and used to store the MySQL version number is deprecated
- The “relay_log_info_file” system variable and “–master-info-file” option are deprecated; the use of files for the relay log info log and master info log has been superseded by crash-safe slave tables, which are the default in MySQL 8.0.
- The use of the “MYSQL_PWD” environment variable to specify a MySQL password is deprecated.



PERCONA

# **Deprecations.** You must act ASAP!



## Deprecations

- the “utf8mb3” character set is deprecated, with “utf8mb4” recommended instead; “utf8mb3” is still valid in MySQL 8.0
- the “sha256_password” plugin is deprecated and “caching_sha2_password” is recommended to use instead, as its superset
- the “mysql_upgrade” client is deprecated because of its capabilities for upgrading the system tables in the mysql system schema, and objects in other schemas have been moved to MySQL server; as of MySQL 8.0.16 the server performs all these tasks
- the “validate_password” plugin has been reimplemented to use the server component infrastructure; it is still available but deprecated
- the “ENGINE” clause for all the ALTER TABLESPACE and DROP TABLESPACE statements is deprecated
- the “PAD_CHAR_TO_FULL_LENGTH” SQL mode is deprecated
- the “AUTO_INCREMENT” support is deprecated for columns of type FLOAT and DOUBLE (and any synonyms); consider removing AUTO_INCREMENT attribute from such columns, or convert them to an integer type
- the “UNSIGNED” attribute is deprecated for columns of type FLOAT, DOUBLE and DECIMAL (and any synonyms); consider using a simple CHECK constraint instead
- FLOAT(M,D) and DOUBLE(M,D) syntax to specify the number of digits for columns of type FLOAT and DOUBLE (and any synonyms) is a nonstandard MySQL extension and is is deprecated
- The nonstandard C-style “&&”, “||”, and “!” operators that are synonyms for the standard SQL AND, OR, and NOT operators, respectively, are deprecated
- The “mysql_upgrade_info file”, which creates a text file in the data directory and used to store the MySQL version number is deprecated
- The “relay_log_info_file” system variable and “--master-info-file” option are deprecated; the use of files for the relay log info log and master info log has been superseded by crash-safe slave tables, which are the default in MySQL 8.0.
- The use of the “MYSQL_PWD” environment variable to specify a MySQL password is deprecated.



PERCONA

- Deprecations are changes in the MySQL which will lead to Removals in future releases
- While the features still exist, their usage will spill errors into log files and console output

PERCONA

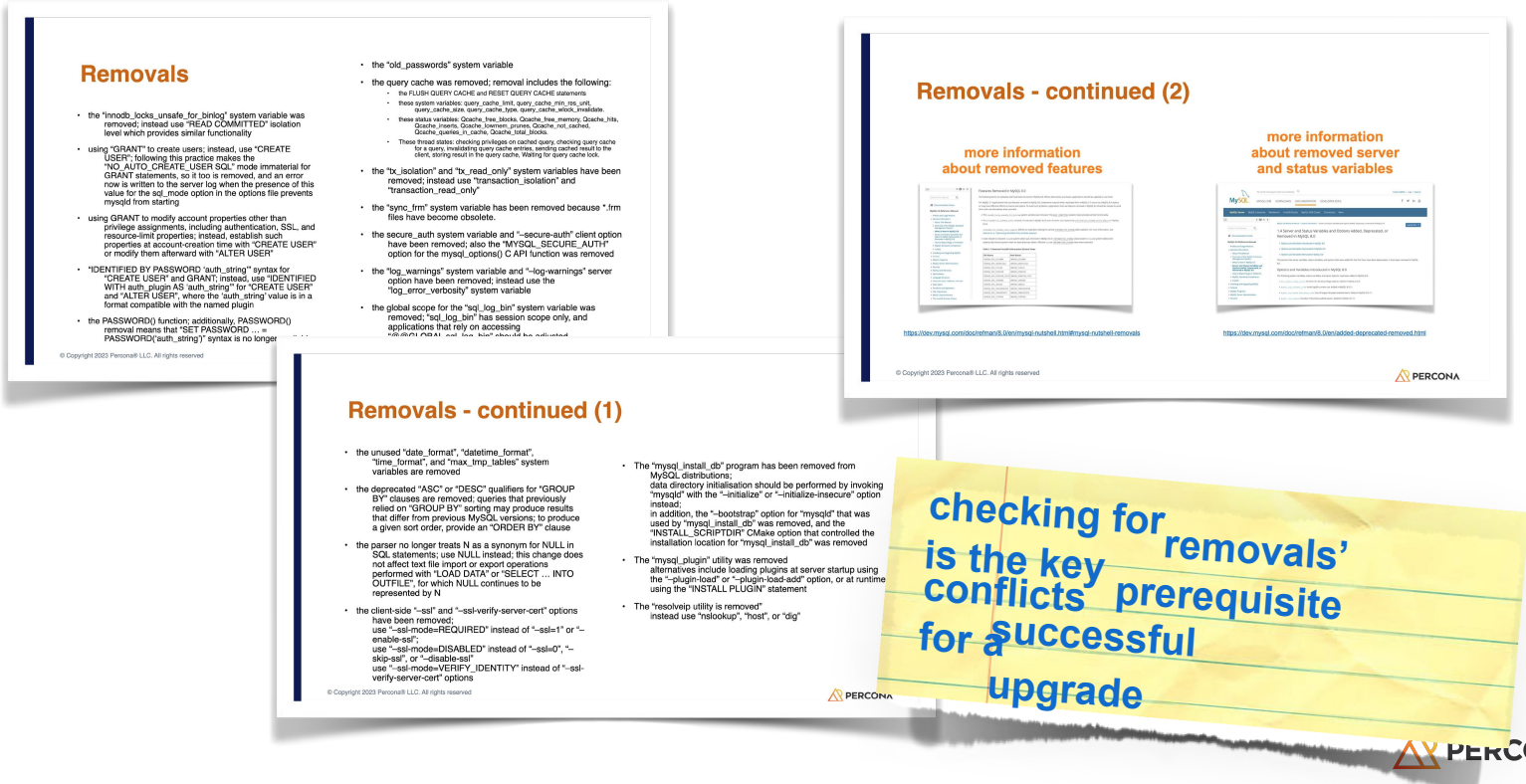
# Removals

- the “innodb_locks_unsafe_for_binlog” system variable was removed; instead use “READ COMMITTED” isolation level which provides similar functionality
- using “GRANT” to create users; instead, use “CREATE USER”; following this practice makes the “NO_AUTO_CREATE_USER SQL” mode immaterial for GRANT statements, so it too is removed, and an error now is written to the server log when the presence of this value for the sql_mode option in the options file prevents mysqld from starting
- using GRANT to modify account properties other than privilege assignments, including authentication, SSL, and resource-limit properties; instead, establish such properties at account-creation time with “CREATE USER” or modify them afterward with “ALTER USER”
- “IDENTIFIED BY PASSWORD ‘auth_string’” syntax for “CREATE USER” and GRANT; instead, use “IDENTIFIED WITH auth_plugin AS ‘auth_string’” for “CREATE USER” and “ALTER USER”, where the ‘auth_string’ value is in a format compatible with the named plugin
- the PASSWORD() function; additionally, PASSWORD() removal means that “SET PASSWORD … = PASSWORD(‘auth_string’)” syntax is no longer available
- the “old_passwords” system variable
- the query cache was removed; removal includes the following:
  - the FLUSH QUERY CACHE and RESET QUERY CACHE statements
  - these system variables: query_cache_limit, query_cache_min_res_unit, query_cache_size, query_cache_type, query_cache_wlock_invalidate.
  - these status variables: Qcache_free_blocks, Qcache_free_memory, Qcache_hits, Qcache_inserts, Qcache_lowmem_prunes, Qcache_not_cached, Qcache_queries_in_cache, Qcache_total_blocks.
  - These thread states: checking privileges on cached query, checking query cache for a query, invalidating query cache entries, sending cached result to the client, storing result in the query cache, Waiting for query cache lock.
- the “tx_isolation” and “tx_read_only” system variables have been removed; instead use “transaction_isolation” and “transaction_read_only”
- the “sync_frm” system variable has been removed because *.frm files have become obsolete.
- the secure_auth system variable and “–secure-auth” client option have been removed; also the “MYSQL_SECURE_AUTH” option for the mysql_options() C API function was removed
- the “log_warnings” system variable and “–log-warnings” server option have been removed; instead use the “log_error_verbosity” system variable
- the global scope for the “sql_log_bin” system variable was removed; “sql_log_bin” has session scope only, and applications that rely on accessing

# Removals - continued (1)

- the unused “date_format”, “datetime_format”, “time_format”, and “max_tmp_tables” system variables are removed
- the deprecated “ASC” or “DESC” qualifiers for “GROUP BY” clauses are removed; queries that previously relied on “GROUP BY” sorting may produce results that differ from previous MySQL versions; to produce a given sort order, provide an “ORDER BY” clause
- the parser no longer treats N as a synonym for NULL in SQL statements; use NULL instead; this change does not affect text file import or export operations performed with “LOAD DATA” or “SELECT … INTO OUTFILE”, for which NULL continues to be represented by N
- the client-side “–ssl” and “–ssl-verify-server-cert” options have been removed;
  use “–ssl-mode=REQUIRED” instead of “–ssl=1” or “–enable-ssl”;
  use “–ssl-mode=DISABLED” instead of “–ssl=0”, “–skip-ssl”, or “–disable-ssl”
  use “–ssl-mode=VERIFY_IDENTITY” instead of “–ssl-verify-server-cert” options
- The “mysql_install_db” program has been removed from MySQL distributions;
  data directory initialisation should be performed by invoking “mysqld” with the “–initialize” or “–initialize-insecure” option instead;
  in addition, the “–bootstrap” option for “mysqld” that was used by “mysql_install_db” was removed, and the “INSTALL_SCRIPTDIR” CMake option that controlled the installation location for “mysql_install_db” was removed
- The “mysql_plugin” utility was removed
  alternatives include loading plugins at server startup using the “–plugin-load” or “–plugin-load-add” option, or at runtime using the “INSTALL PLUGIN” statement
- The “resolveip utility is removed”
  instead use “nslookup”, “host”, or “dig”

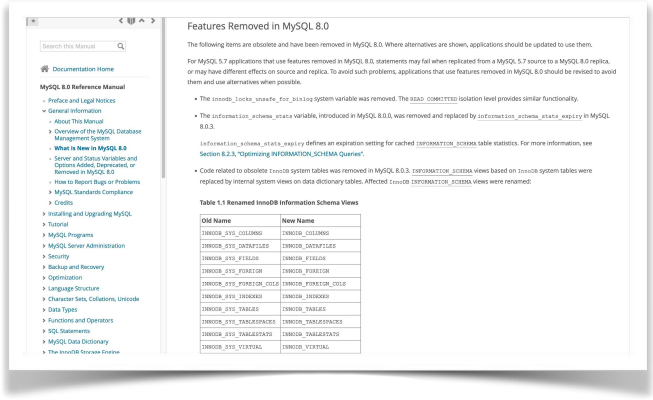
# **Removals.** This is your homework before upgrade.




PERCONA

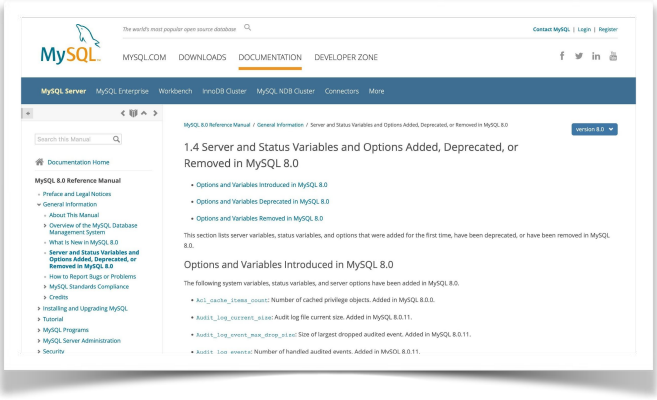
# Removals - continued (2)

more information
about removed features




https://dev.mysql.com/doc/refman/8.0/en/mysql-nutshell.html#mysql-nutshell-removals

more information
about removed server
and status variables




https://dev.mysql.com/doc/refman/8.0/en/added-deprecated-removed.html

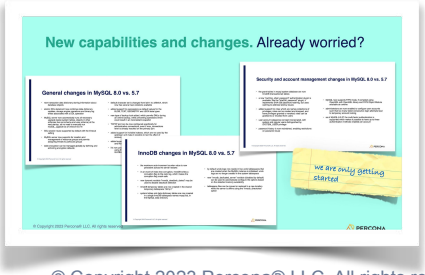


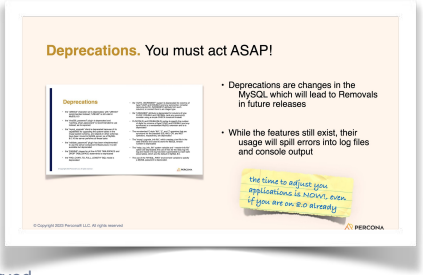
PERCONA

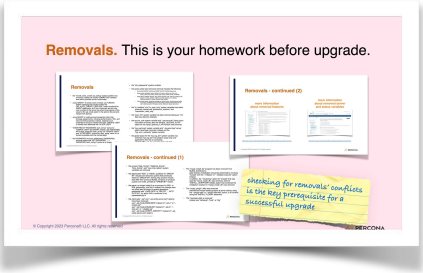
# Upgrading MySQL from 5.7 to 8.0 and 8.4

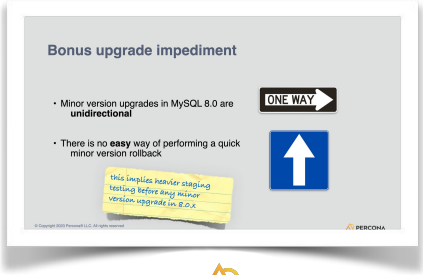
**Is it tricky to upgrade from MySQL 5.7 to 8.0 and 8.4?**

**Can it be done safely and without a system outage?**



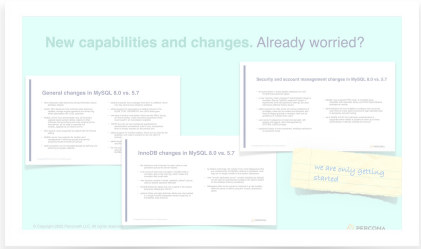


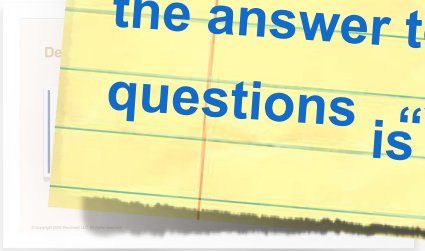

PERCONA

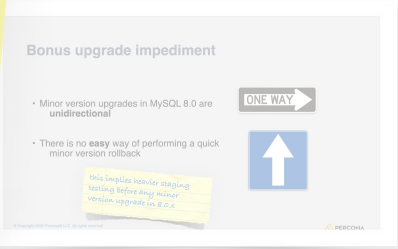
# Upgrading MySQL from 5.7 to 8.0 and 8.4

**Is it tricky to upgrade from MySQL 5.7 to 8.0 and 8.4?**

**Can it be done safely and without a system outage?**






**the answer to both questions is "YES"... again**



PERCONA

# How to ease the major version upgrade!

## Tools that can help!

### MySQL Shell Upgrade Checker Utility

Validates database compatibility and detects potential upgrade issues before major MySQL version upgrades.

### pt-config-diff

Compares MySQL configuration files or server variables to find differences between instances.

### pt-upgrade

Runs queries on multiple server versions to compare results and identify compatibility differences.

### pt-query-digest

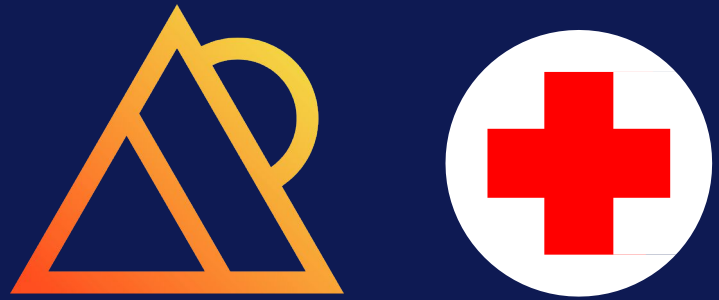
Analyzes query logs and performance schema data to identify slow or resource-intensive SQL statements.

Enable the slow log for a suitable period to capture most queries, then use pt-query-digest to summarize the large log for analysis and testing.

### pt-show-grants

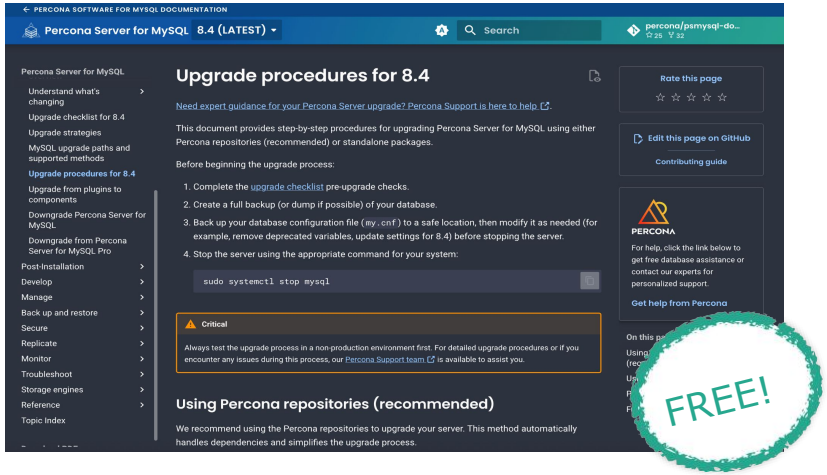
Extracts and displays user privileges as GRANT statements for backup or migration.

# How can Percona help?



# How to upgrade MySQL to 8.4 with Percona?

## by yourself
## using Percona guidelines



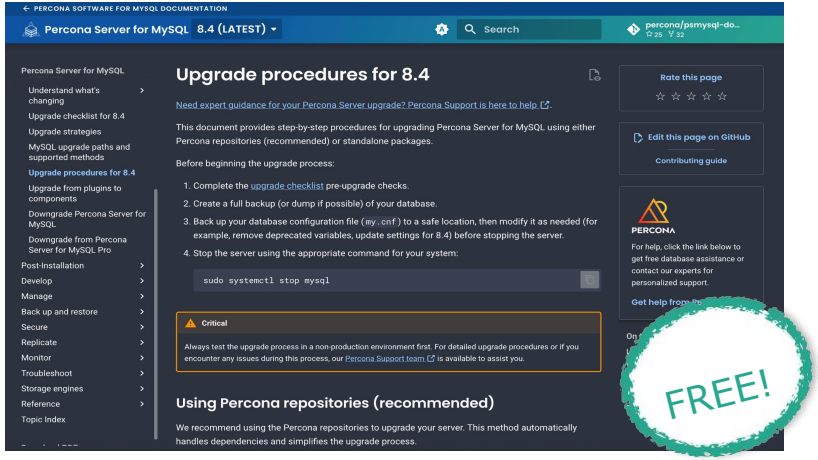

https://docs.percona.com/percona-server/8.4/upgrade-procedures.html



PERCONA

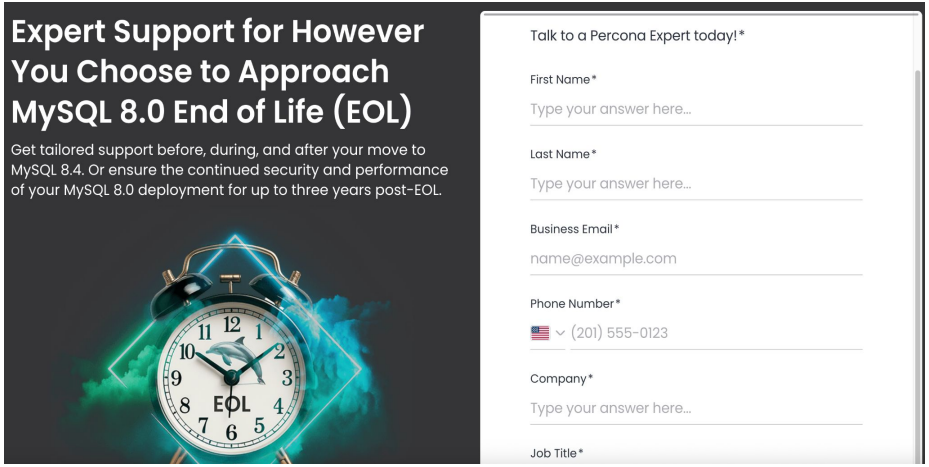
# How to upgrade MySQL to 8.4 with Percona?

## by yourself
## using Percona guidelines




https://docs.percona.com/percona-server/8.4/upgrade-procedures.html

## with help from Percona support and professional services / post-EOL




https://www.percona.com/upgrading-to-mysql-8-0-with-percona



PERCONA

# How to upgrade MySQL to 8.4 with Percona?

## 1. Let us handle the upgrade process

Percona experts have dealt with many MySQL upgrades.

We can perform:

- a complete upgrade,
- assist and advise you during the upgrade process,
- or you can call us to resolve failed upgrades

## 2. Start at any phase of an upgrade

Whether you have started the process already, or want Percona to handle it for you end-to-end.

## 3. Percona or non-Percona distributions

Percona provides professional upgrade services and support regardless if you are using the MySQL Community edition, or the Pecona Server for MySQL.

## 4. We can also help migrate to Percona

If your current distribution of MySQL does not meet your criteria, we can migrate you over to Percona Distribution for MySQL.

## 5. Percona experts are there to help

Percona experts are ready to assist you in the self-managed upgrade process as well.



PERCONA

# Is there life after ~~death~~ 8.0 EOL?



PERCONA

# “What if I need to stay on MySQL 8.0 longer?”

## MySQL 8.0 EOL Support

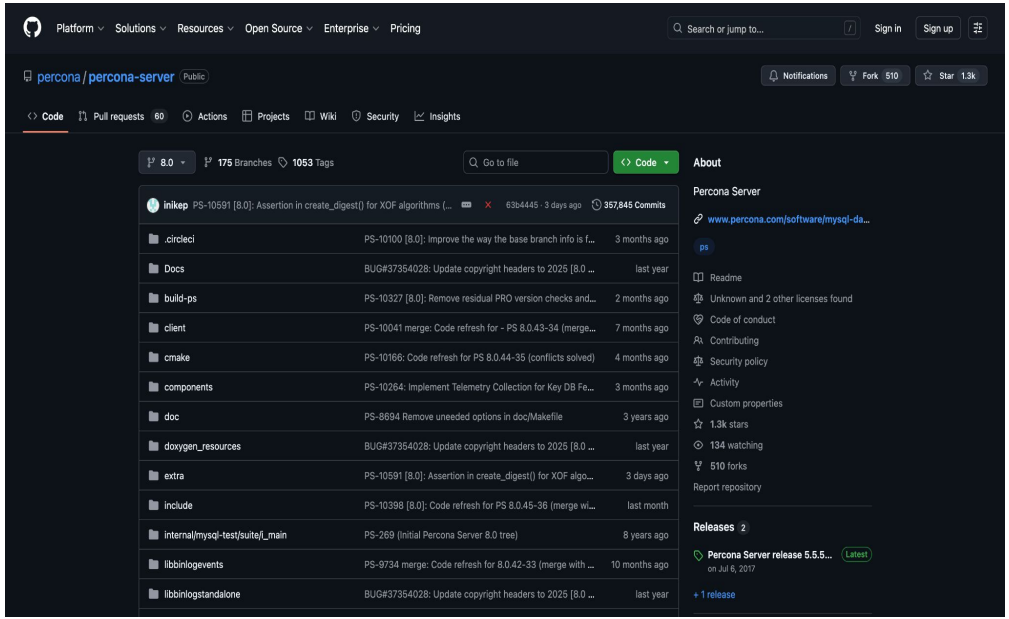
Stay secure on MySQL 8.0 with Percona’s post-EOL support. Our experts deliver ongoing security updates, bug fixes, and hands-on assistance for up to three years after the official End of Life date.

Includes:

- Expert-engineered fixes for detected, reproducible CVEs
- 24x7x365 hands-on monitoring, support, and issue resolution
- Reports on additional testing of CVEs
- Guaranteed SLAs

https://www.percona.com/mysql-8-0-eol-support

# “What if I cannot afford Percona support?”




https://github.com/percona/percona-server/tree/8.0

- Percona is a true open-source company
- all 8.0 patches will be published through GitHub
- changes will be applied only to Percona MySQL fork
- you can contribute to post-EOL code as well




PERCONA

# Thank you!

# PERCONA Live 26

BAY AREA

May 27-29, 2026
Computer History Museum,
Mountain View, California

perconalive.com

PERCONA