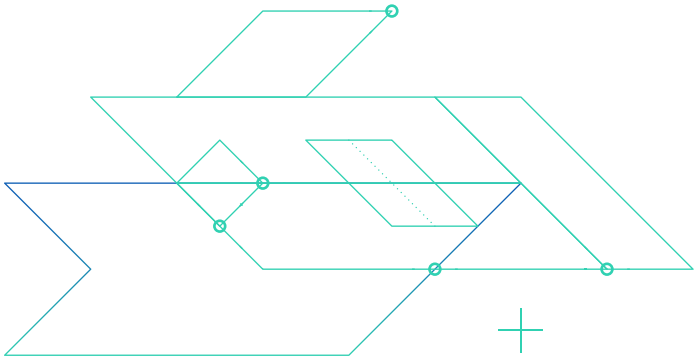
A Comprehensive Plan for Migration from MySQL 8.0 to MySQL 8.4

PERCONA®



I. Executive Summary: The Strategic Case for Upgrading to MySQL 8.4

This report provides a comprehensive, risk-mitigated plan for migrating a database from MySQL 8.0 to the MySQL 8.4 Long-Term Support (LTS) release. The strategic importance of adopting an LTS version for production environments lies in its extended support lifecycle, which ensures long-term stability, security patching, and predictability for business-critical applications. The migration from one major version to another is not a trivial task and requires meticulous planning to avoid operational disruptions. This document outlines the necessary preliminary steps, details three distinct migration methodologies, and provides a final validation and optimization checklist to ensure a successful and smooth transition. The central thesis of this report is that a "simple" upgrade is a misnomer. The most critical phase is not the migration itself, but the preparation and pre-flight checks. The new server defaults in MySQL 8.4 may fundamentally alter performance characteristics, and deprecated features could break application or operational scripts.

For enterprises that operate in an open-source ecosystem, the migration process can be supported by third-party tools and services, such as those provided by Percona.

Tools like Percona Toolkit are specifically engineered to assist with complex tasks such as data migration and performance analysis. Percona XtraBackup also provides a hot backup utility for MySQL 8.4, which is a critical part of a robust migration strategy.

The core recommendations of this plan are to:

1. Proactively identify and address incompatibilities using automated tools;

2. Choose the appropriate migration method based on the specific use case, risk tolerance, and required downtime;

3. Perform a full dry run in a non-production environment to validate the entire process.

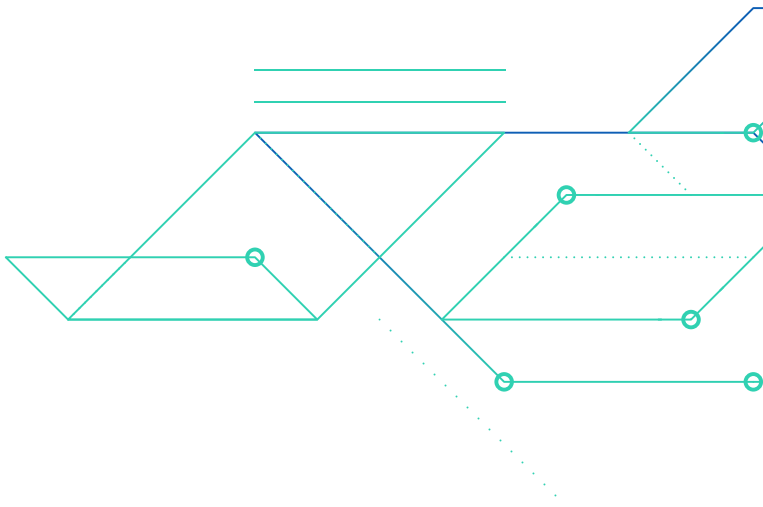
II. The Transition from MySQL 8.0 to MySQL 8.4: A Comprehensive Release Notes Analysis

Upgrading from MySQL 8.0 to MySQL 8.4 involves more than simply replacing binaries. The new version introduces a variety of incompatible changes, new default behaviors, and the removal of deprecated features, all of which can significantly impact existing applications and operational practices.4 A thorough understanding of these changes is essential for a successful migration.

A. Incompatible Changes and Breaking Behaviors

A number of critical changes in MySQL 8.4 necessitate a review of both the database schema and application code prior to migration.

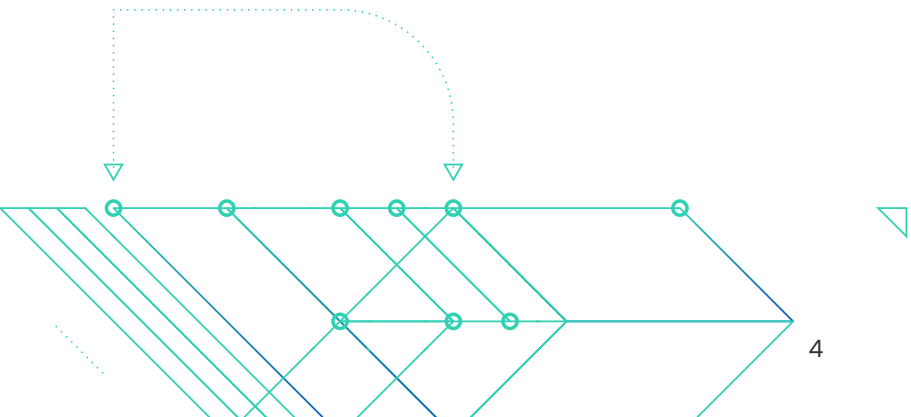
- **Authentication and User Management:** The most significant change for application connectivity is the shift in default authentication. The deprecated *mysql_native_password* authentication plugin is now disabled by default as of MySQL 8.4.0.6. This means any application or user account that relies on this legacy authentication method will fail to connect after the upgrade unless the plugin is explicitly re-enabled via the *mysql-native-password=ON* server option. This is a deliberate security enhancement, pushing users toward the more secure *caching_sha2_password* plugin. Additionally, the *default_authentication_plugin* variable has been removed entirely, with the system now defaulting to caching_sha2_password for all new user accounts. This change represents a clear strategic directive from Oracle to modernize and secure the MySQL ecosystem, moving away from older, less secure protocols. The implication is that a seemingly straightforward upgrade could lead to widespread application downtime if a proactive analysis is not performed.



- **Replication Terminology:** The MySQL documentation confirms a complete removal of the MASTER and SLAVE terminology from the SQL syntax. All related commands, such as START SLAVE, SHOW SLAVE STATUS, and CHANGE MASTER TO have been removed and replaced with new REPLICA and SOURCE terminology. For example, START SLAVE is now START REPLICA, and CHANGE MASTER TO is now CHANGE REPLICATION SOURCE TO. This requires a comprehensive review and update of any custom scripts, applications, or operational tooling that manages replication, as they will cease to function with the new syntax.
- **Spatial Indexes:** A known bug exists up to MySQL version 8.0.40, in which a spatial index could become corrupted after a minimal update to a geometry, followed by a delete operation. To mitigate this risk and prevent data loss, the official recommendation is to drop any spatial indexes on tables before the upgrade and then re-create them after the migration is complete. This is a crucial step to ensure data integrity, or better upgrade to the latest available version of 8.0 to get the bug fix.
- **New Reserved Keywords:** The introduction of new reserved words, such as MANUAL, PARALLEL, QUALIFY, and TABLESAMPLE, can cause schema-related issues. If any database objects, such as table or column names, use these words as unquoted identifiers, the upgrade will fail or application queries will break. A pre-upgrade check is mandatory to identify and properly quote these identifiers.
- **Data Type Restrictions:** The AUTO_INCREMENT option is no longer permitted on FLOAT and DOUBLE data types. The presence of a table containing such a column will cause the upgrade process to fail, requiring a data type conversion prior to migration.

B. Changed Server Defaults and Their Performance Implications

MySQL 8.4 is pre-tuned for modern hardware, and several server variable defaults have been adjusted to reflect this. A simple in-place upgrade that carries over the old configuration file may not leverage these improvements and, in some cases, could lead to unexpected performance behavior.



The table below illustrates some of the most impactful changes in InnoDB system variable defaults.

InnoDB System Variable Name	New Default Value (MySQL 8.4)w	Previous Default Value (MySQL 8.0)
innodb_adaptive_hash_index	OFF	ON
innodb_change_buffering	none	all
innodb_flush_method on Linux	O_DIRECT if supported, otherwise fsync	fsync
innodb_io_capacity	10000	200
innodb_log_buffer_size	67108864 (64 MiB)	16777216 (16 MiB)
innodb_numa_interleave	ON	OFF
temptable_max_ram	3% of total memory (1-4 GiB range)	1073741824 (1 GiB)
innodb_parallel_read_threads	available logical processors / 8 (min 4)	4

These changes are not arbitrary; they are a direct response to the evolution of server hardware and a reflection of common production workloads. The dramatic increase in *innodb_io_capacity* from 200 to 10000 signals a deliberate effort to leverage high-speed SSD and NVMe storage for I/O-bound operations. Similarly, a larger *innodb_log_buffer_size* reduces the frequency of physical writes to the redo log, which is beneficial for write-heavy workloads. The change in *innodb_adaptive_hash_index* default from ON to OFF represents a shift toward predictable performance, as the adaptive hash index can be a source of concurrency bottlenecks in certain scenarios.

An existing my.cnf file from an 8.0 installation will likely not be optimized for these new realities. A DBA who previously manually tuned their system to *innodb_io_capacity = 1000* for an SSD will find the new default of 10000 to be a significant improvement. Conversely, a server on legacy spinning disks might be negatively impacted by these aggressive defaults. This highlights that the old configuration file should not be blindly used; it must be reviewed and re-evaluated against the new defaults, or a new configuration should be generated to maximize the benefits of the upgrade.

C. Plugin and Component Upgrades

Percona Server for MySQL 8.4 introduces a shift from plugins to components for several key features, a change that requires a manual transition during the upgrade process. It is generally recommended to transition to the component version of a feature in the 8.0 series before upgrading to 8.4 if both a plugin and a component are available for that feature.

The configuration of these features also changes: plugins use system variables and the --early-plugin-load option, while components rely on a separate configuration file and are loaded using a manifest.

Key plugin changes and their recommended transition paths include:

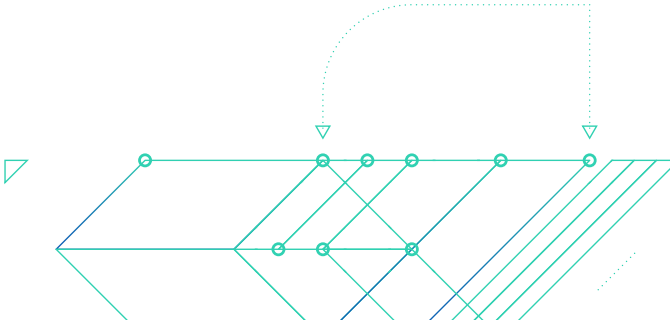
- **keyring_vault:** The primary differences between MySQL Keyring plugins and Keyring components center around their implementation, loading, configuration, and capabilities. Plugins rely on the older server plugin infrastructure, are loaded using the early-plugin-load option, and configured via plugin-specific system variables. In contrast, components use the newer component infrastructure, are loaded via a manifest, and are configured using individual component-specific configuration files rather than system variables. Furthermore, components offer fewer restrictions on supported key types and lengths, and, critically, they provide secure storage for persisted system variable values (like passwords and private keys), a feature that plugins do not support.
- **audit_log:** This plugin has been removed entirely in 8.4. The recommended replacement is component_audit_log_filter.
- **audit_log_filter:** This plugin has a corresponding component. The transition to the component should be performed after the upgrade to 8.4.
- **data_masking:** For this feature, it is specifically advised to transition to the component_masking_functions in the 8.0 series before upgrading to 8.4.
- **binlog_utils_udf and percona-udf:** These user-defined functions, previously installed via a plugin, are now available as components. After running INSTALL COMPONENT, all functions are automatically registered, which simplifies the process.

The general procedure for transitioning from a plugin to a component involves setting up the new component's configuration file, loading the component using a manifest, thoroughly testing its functionality in a staging environment, and then removing the original plugin.32 This process should be carefully planned to minimize downtime and ensure that all existing functionality is correctly transferred

D. Percona Toolkit Compatibility and Changes

Percona, a company that provides software and services for open-source databases, has updated its popular Percona Toolkit to support MySQL 8.4.15. These updates address the breaking changes introduced by MySQL 8.4, and any user relying on these tools should be aware of the following:

- **New Terminology:**
  Percona Toolkit now fully supports the new REPLICA and SOURCE terminology in MySQL 8.4.15
- **Renamed Tools:**
  To align with the new terminology, the tools pt-slave-find and pt-slave-restart have been renamed to pt-replica-find and pt-replica-restart, respectively. Aliases with the old names still exist for a transition period, but it is recommended to update scripts to use the new names.
- **Deprecated Tool:**
  The pt-slave-delay tool has been deprecated because its functionality can be achieved using built-in delayed replication features.
- **Improved Authentication Support:**
  Percona Toolkit has also received updates to improve SSL support and now consistently supports the caching_sha2_password and sha256_password authentication plugins.



E. Deprecated and Removed Features

A number of features have been removed to streamline the server, simplify the codebase, and align with modern standards.

- **Replication Statements & Variables:** A wide array of MASTER/SLAVE commands and status variables have been removed. This is part of a broader terminology clean-up, but it also means that any scripts or tools that rely on these status variables (e.g., Com_show_slave_status) will no longer function.
- **Authentication Variables:** The default_authentication_plugin variable is now removed.
- **Removed Functions:** The WAIT_UNTIL_SQL_THREAD_AFTER_GTIDS() SQL function, which was deprecated in 8.0, has been removed; its replacement is WAIT_FOR_EXECUTED_GTID_SET().
- **System Variables:** The *expire_logs_days* system variable, used for binary log purging, has been removed in favor of *binlog_expire_logs_seconds*.
- **Memcached-related Features:** All variables related to the built-in memcached functionality, such as daemon_memcached and innodb_api, have been entirely removed. This signifies a deprecation of this feature within the core offering.

The removal of these features and variables is not arbitrary. It represents a clear strategic direction for MySQL as a modern LTS product. Removing redundant, obsolete, or tangential features simplifies the server and reduces the surface area for bugs and maintenance. It is the user's responsibility to ensure that their operational scripts and applications do not rely on these now-obsolete components.

F. Upgrade path

Upgrade Path	Path Examples	Supported Upgrade Methods
Within an LTS or Bugfix series	8.0.37 to 8.0.41 or 8.4.0 to 8.4.4	In-place upgrade, logical dump and load, replication, and MySQL Clone
From an LTS or Bugfix series to the next LTS series	8.0.37 to 8.4.x LTS	In-place upgrade, logical dump and load, and replication
From an LTS or Bugfix release to an Innovation release before the next LTS series	8.0.34 to 8.3.0 or 8.4.0 to 9.0.0	In-place upgrade, logical dump and load, and replication
From the Innovation series to the next LTS series	8.3.0 to 8.4 LTS	In-place upgrade, logical dump and load, and replication
From an Innovation series to an Innovation release after the next LTS series	Not allowed, two steps are required: 8.3.0 to 8.4 LTS, and 8.4 LTS to 9.x Innovation	In-place upgrade, logical dump and load, and replication

III. Removed Parameters and Functions in MySQL 8.4

A key part of a successful migration is identifying and addressing all removed parameters, variables, and functions. Using any of these in a configuration file or application code after the upgrade will result in an error and prevent the server from starting or the application from running. The mysqlsh upgrade checker utility can identify many of these issues, but a manual review of this list is also a critical part of the preparation.

A. Removed Server and Replication System Variables

Variable Name	Description	Replacement
avoid_temporal_upgrade	Whether ALTER TABLE should upgrade pre-5.6.4 temporal columns.	N/A
binlog_transaction_ dependency_tracki ng	Source of dependency information from which the multithreaded applier assesses parallel execution.	Functionality is now internal.
character-set-client-handshake	Do not ignore client-side character set value sent during handshake.	N/A
default_authentication_ plugin	Default authentication plugin.	authentication_policy
expire_logs_days	Purge binary logs after a number of days.	binlog_expire_logs_ second s
group_replication_primary_ member	Primary member UUID when in single-primary mode.	N/A
group_replication_ recovery_complete_ at	Recovery policies when handling cached transactions.	N/A
have_openssl	Whether the server supports SSL connections.	N/A
have_ssl	Whether the server supports SSL connections.	N/A
innodb_api_... variables	All innodb_api variables related to the built-in memcached functionality.	N/A

B. Removed Server Options, SQL Statements, and Status Variables

Item Name	Type	Replacement
admin-ssl	Server Option	--tls-version and--admin-tls-version
authentication_fido_rp_id	Server Option	N/A
--language	Server Option	N/A
--old and --new	Server Option	N/A
Com_change_master	Status Variable	Com_change_replication_source
Com_show_master_status	Status Variable	Com_show_binary_log_status
Com_show_slave_status	Status Variable	Com_show_replica_status
Com_slave_start	Status Variable	Com_replica_start
Com_slave_stop	Status Variable	Com_replica_stop
CHANGE MASTER TO	SQL Statement	CHANGE REPLICATION SOURCE TO
SHOW SLAVE STATUS	SQL Statement	SHOW REPLICA STATUS
START SLAVE	SQL Statement	START REPLICA
STOP SLAVE	SQL Statement	STOP REPLICA
WAIT_UNTIL_SQL_THREAD_AFTER_GTID S()	SQL Function	WAIT_FOR_EXECUTED_GTID_S ET()

IV. Ecosystem Compatibility: A Review of Third-Party Tools

A major version upgrade impacts not only the database server but also the entire ecosystem of tools and applications that interact with it. Ensuring compatibility with third-party tools like backup utilities, proxies, and monitoring solutions is a critical step in a successful migration.

A. Percona XtraBackup (PXB)

Percona XtraBackup is a widely used open-source hot backup utility. It has a specific versioning and compatibility model that must be considered before upgrading.

- **Version-Specific Backups:** Percona XtraBackup 8.4 is designed to create backups of data from MySQL 8.4, Percona Server for MySQL 8.4, and Percona XtraDB Cluster 8.4. It does not support backing up databases from MySQL 8.0 or 9.x servers.

B. Percona Operator for MySQL upgrade

The upgrade process from Percona XtraDB Cluster (PXC) 8.0 to PXC 8.4 using the Percona Operator for MySQL on Kubernetes involves creating a new installation of the Percona Operator solution using the Percona Operator for PXC 8.4. Then, recover the data from an original 8.0 backup, then establish asynchronous replication between the two clusters to keep data in sync.

In-place upgrade is not suggested at the time of writing, and while it may work, there is no guarantee it will.

C. ProxySQL

ProxySQL is an open-source, high-performance proxy for MySQL. Its ability to route and manage traffic makes it a critical component in many high-availability and sharded environments. Compatibility with MySQL 8.4 is essential for seamless operation.

MySQL 8.4 Support:

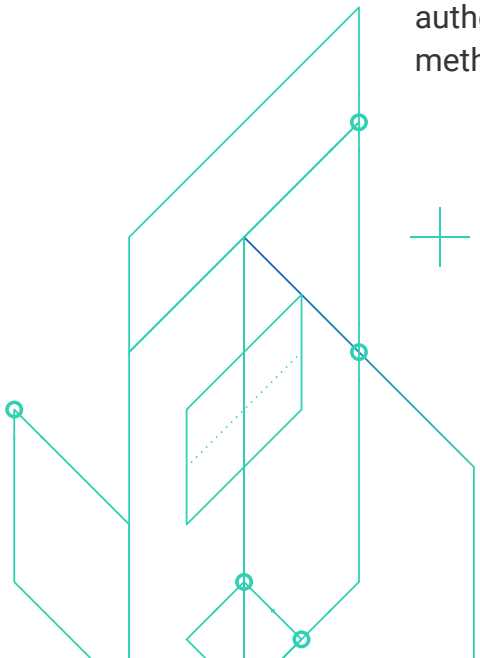
Recent versions of ProxySQL, including those available in Percona's Operator for MySQL, have added support for MySQL 8.4. ProxySQL now includes Group Replication support for MySQL 8.4 and 9.x

Authentication Plugin:

ProxySQL versions 2.6 and later are compatible with the caching_sha2_password authentication plugin, which is the new default in MySQL 8.4. This is a crucial compatibility check, as applications connecting through the proxy will need to support the new authentication method.

Replication Terminology:

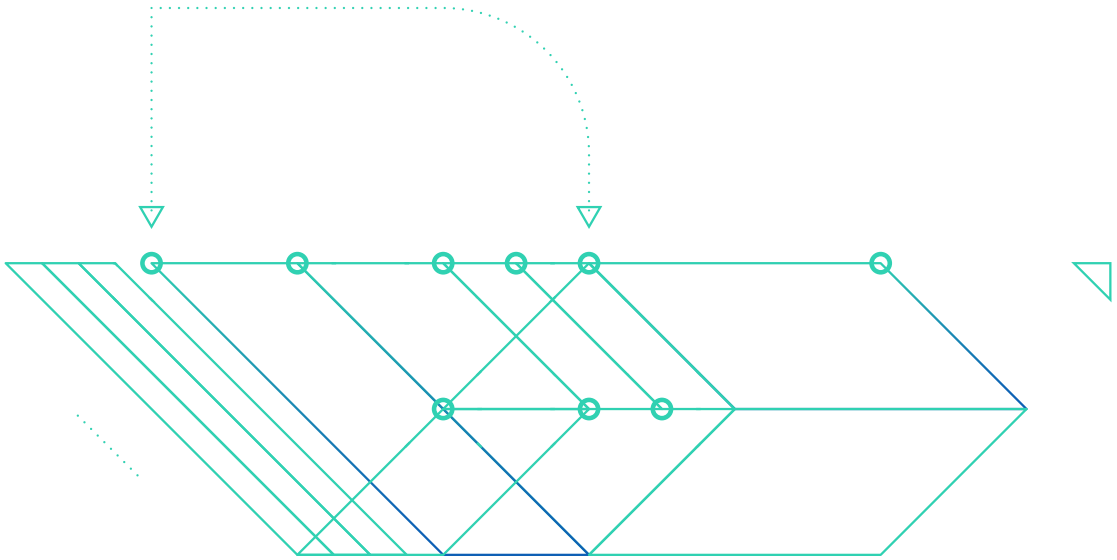
As a key component in replication setups, ProxySQL's compatibility with the new REPLICA/SOURCE terminology is vital. The research indicates that a replication setup using MySQL 8.4 with ProxySQL can be configured with the new syntax. Additionally, ProxySQL can be configured to monitor replica lag and stop sending traffic to a replica if its replication is broken or has a high lag.



D. Percona Monitoring and Management (PMM)

Percona Monitoring and Management (PMM) is an open-source solution for monitoring the health and performance of database systems. Its compatibility ensures that performance metrics, query analytics, and alerting continue to function after the upgrade.

- **MySQL 8.4 Monitoring:** PMM is designed to monitor a variety of MySQL variants, including Percona Server for MySQL, Percona XtraDB Cluster, and Oracle MySQL Community Edition. A recent community blog post announced "complete MySQL 8.4 replication monitoring" in PMM version 3.4.0. This confirms that PMM has been updated to handle the new syntax and features of the 8.4 release.
- **Tool Integrations:** PMM supports monitoring for a wide range of technologies, including ProxySQL and Percona XtraDB Cluster, ensuring that the entire database ecosystem can be observed from a single pane of glass.



V. Phase 1: A Production-Grade Pre-Migration Strategy

The success of a major version upgrade is determined not by the speed of the migration itself, but by the thoroughness of the preliminary planning and testing phases. This is the single most important part of the entire process.

A.The Golden Rules

#1: Test, Test, Test

A migration should never be attempted on a production server without first performing a dry run in a staging or test environment. This environment must be a clone of the production system, using the same dataset, hardware specifications, and representative application workload. This is the only way to validate the entire upgrade procedure, from pre-flight checks to post-migration application testing, without risk of production downtime or data loss. Percona also strongly recommends creating a test environment, noting that there is no supported downgrade procedure from MySQL 8.4. For clustered environments, the best practice is to upgrade one node at a time, starting with the secondary nodes to minimize impact.

#2 Upgrade to Latest Minor Version

To ensure stability and security, it is mandatory to upgrade your MySQL or Percona Server installation to the latest minor version available for your current major release. This action is critical for receiving all accumulated bug fixes. For example, the spatial index corruption issue (Bug #36452528) was officially resolved in the MySQL 8.0.41 release.

#3 Use Percona Toolkit's pt-upgrade Utility:

Function: This critical diagnostic tool enables pre-upgrade performance validation. It systematically compares query plans and execution behavior between your current production environment (e.g., Percona Server 8.0) and the target release (e.g., Percona Server 8.4). Benefit: Allows the early detection of performance regressions or unexpected changes in query optimization before deployment.

B. Automated Pre-Flight Checklist: The mysqlsh Upgrade Checker

The cornerstone of a safe migration is the mysqlsh utility's *util.checkForServerUpgrade()* function. This tool proactively scans the database and identifies potential upgrade blockers before any changes are made.

Step-by-Step Usage:

1. Install mysqlsh on the source server.
2. Connect to the existing MySQL 8.0 instance.
3. Run the utility and specify the target version: *util.checkForServerUpgrade({version: '8.4.0'})*.
4. Review the generated report carefully and address every flagged issue.

The utility performs a wide range of checks, including:

- **reservedKeywords:** Detects database object names that conflict with the new reserved words in 8.4.
- **defaultAuthenticationPlugin:** Identifies user accounts that still rely on the now-disabled mysql_native_password plugin.
- **removedFunctions:** Flags functions used in stored routines that have been removed in the target version.
- **sysVarsNewDefaults:** Reports system variables with changed defaults, which prompts the DBA to review and potentially adjust their configuration file.
- **spatialIndex:** Flags spatial indexes that need to be dropped and re-created after the upgrade to prevent corruption.

C. Establishing a Performance Baseline

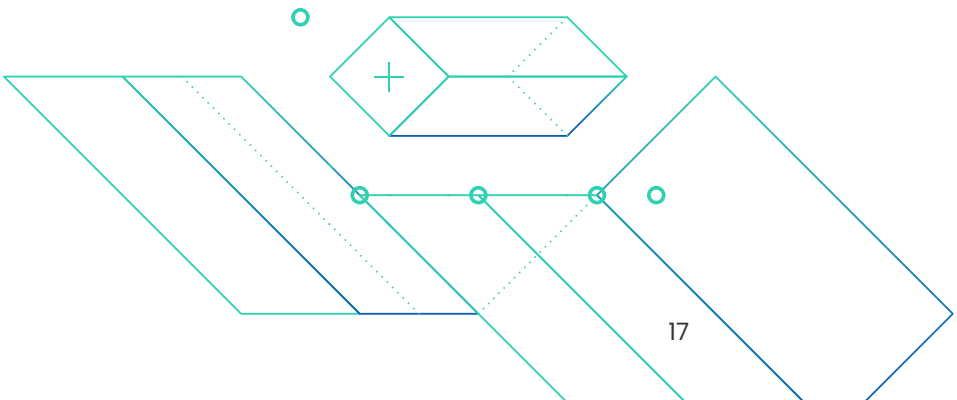
Prior to any changes, a performance baseline of the current MySQL 8.0 environment must be established.4 This involves running a representative application workload on the production system and collecting metrics such as query latency, throughput (QPS), and resource utilization (CPU, I/O).4 This benchmark will serve as a critical point of comparison to validate performance improvements or identify regressions after the upgrade.

D. The Backup Strategy

A full, verifiable backup is a non-negotiable prerequisite for any major version upgrade. Before initiating the migration, the team must confirm that a recent backup exists and has been successfully restored in a test environment. This provides a safety net in the event of unforeseen issues, allowing for a swift rollback to the original state. Percona XtraBackup is a popular tool for this purpose, offering hot backups without disrupting server performance.

VI. Phase 2: Detailed Migration Methodologies

There are three primary methods for migrating from MySQL 8.0 to MySQL 8.4, each with its own advantages and disadvantages based on the specific operational requirements.



A. Method: In-Place Binary Upgrade

This method involves replacing the existing MySQL 8.0 packages or binaries with the new MySQL 8.4 packages on the same server, without moving the data.

- **Ideal Use Case:**
  This method is suitable for small to medium-sized databases where the underlying hardware and operating system are not changing. It offers the lowest perceived complexity and shortest downtime, as the automatic upgrade of system tables is handled upon server restart.

- **Step-by-Step Procedure (Linux-centric):**

  1. **Preparation:** Complete all pre-migration checks from Phase 1.

  2. **Shutdown MySQL 8.0:** Stop the server to ensure a clean upgrade process. For systemd-managed servers, the command would be sudo systemctl stop mysqld. Configure the server to perform a slow shutdown by setting innodb_fast_shutdown=0 before shutting down.

  3. **Replace Binaries:** Install the MySQL 8.4 binaries. On most Linux distributions, this involves updating your package repository configuration and running an upgrade command.

  4. **Restart MySQL 8.4:** Start the new server. The MySQL server will automatically perform the necessary system table upgrades upon startup. The server may be unavailable during this automatic upgrade process.

- **Post-Upgrade Actions:**
  While the automatic upgrade handles most tasks, it is a best practice to run the mysql_upgrade command explicitly to finalize the process and check user tables for incompatibilities. The server must then be stopped and restarted for these changes to take full effect.

B. Method: New environment with cut over

Upgrading with a new environment involves provisioning a duplicate environment with the same number of servers, with the same hardware specs and the same operating system as the current production nodes.

On the newly provided hardware, the target MySQL version will be installed. The new environment will be set up, and the production data will be recovered. Remember that you can use pt-config-diff to verify MySQL configurations.

Replication from the current source to the newly built environment will be established. At cutover time, all writes on the current source will be halted, and the application traffic will need to be redirected to the new source. The cutover can be done using a Virtual IP address or by manually redirecting the application itself. Once writes are being received on the new environment, you are in a fail-forward situation, and the old environment can be torn down.

The new environment strategy has the following pros and cons:

- Additional infrastructure cost since a new environment must be built.
- Ability to upgrade both the OS and the DBMS at the same time.
- Allows upgrade of hardware easily.
- Requires only a single cutover window.

B.1 Logical Migration with MySQL Shell

This is the recommended method for major version upgrades, especially when migrating to new hardware, a different operating system, or a cloud-based environment.It involves exporting the data from the source and importing it into a new, clean target instance.

- **Ideal Use Case:**
  This approach is highly flexible, scalable, and provides the highest level of safety.
- **The Dump Process: util.dumpInstance()**
  - The mysqlsh utility is the modern, multithreaded alternative to the legacy mysqldump tool. It can export an entire instance with parallelism, creating separate, chunked files for each table that can be loaded concurrently.
  - Example command for a parallel dump: *mysqlsh --uri=user:password@host:port--sql --execute "util.dumpInstance('/path/to/dump', {threads: 8, compression: 'zstd'})"*.
  - Key options like threads and compression are vital for performance. Using zstd is recommended for superior compression ratios and faster speeds during both the dump and load processes.
- **The Load Process: util.loadDump()**
  - The *util.loadDump()* utility is designed to import the files created by the dump utility. It is also multithreaded and can load tables and chunks of tables in parallel, significantly reducing the total load time. The threads option, which defaults to 4, controls the level of parallelism.
  - Example command for a parallel load: *mysqlsh root@localhost:3306 -- util load-dump /path/to/backup*.
  - *util.loadDump()* can also handle decompression automatically and can resume interrupted loads, making it robust and reliable.

B2. Logical Migration with MyDumper and MyLoader

MyDumper and MyLoader are high-performance, open source alternatives to mysqldump and mysql that are specifically engineered for parallel data export and import.

- **Ideal Use Case:**
  This is a strong choice for migrating extremely large databases (1TB+) where minimizing downtime is the top priority.

- **The MyDumper Process:**

  - MyDumper's parallelism is controlled with the --threads (-t) option, and a guideline is to use one thread per CPU core. It can also split large tables into smaller, manageable chunks using --chunk-filesize (-F). The recommended compression method is zstd for optimal performance.

  - Example command: mydumper --host=localhost --user=root --password=mypassword --outputdir=/path/to/backup --threads=8 --chunk-filesize=1G --compress=zstd.

- **The MyLoader Process:**

  - MyLoader handles the parallel import, with options for fine-tuned performance. The --threads (-t) option controls concurrency, and a best practice is to allocate one thread for every two vCPUs. The --queries-per-transaction (-q) option allows for balancing transaction size and commit frequency, which can be critical for performance. MyLoader also includes the --optimize-keys option, which loads data without secondary indexes and then builds them afterward, a technique that can significantly accelerate the import process.

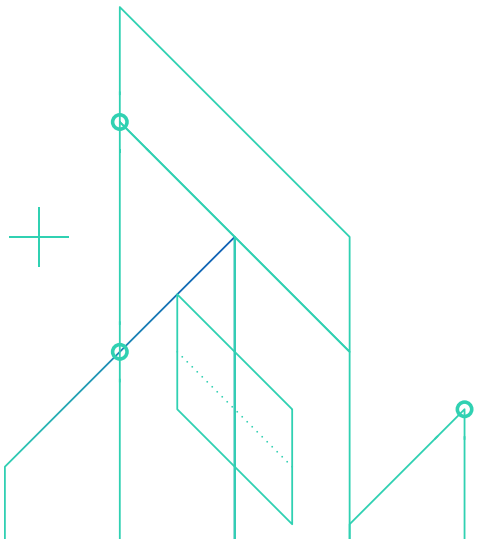
  - Example command: myloader --threads=4 --directory=/path/to/backup --database=mydatabase --queries-per-transaction=1500 --optimize-keys.

This table provides a concise comparative analysis of the three migration methods.

Method	Ideal Use Case	Pros	Cons	Key Command/ Utility
In-Place Binary	Small to medium DB, same hardware/ OS.	Minimal downtime, simple for minor upgrades.	Less flexible, higher risk if pre-checks are skipped.	systemctl start mysqld
Logical with mysqlsh	New hardware/ OS, cloud migrations.	Highly flexible, robust, official Oracle tool, multithreaded.	Slower than in-place, requires additional storage for dump files.	util. dumpInstance(), util.loadDump()
Logical with MyDumper	Extremely large databases (10 TB+).	Fastest logical method, fine-grained control, robust.	Requires a separate tool installation, higher complexity.	mydumper, myloader

VII. Phase 2.5: Upgrading Percona XtraDB Cluster (PXC) from 8.0 to 8.4

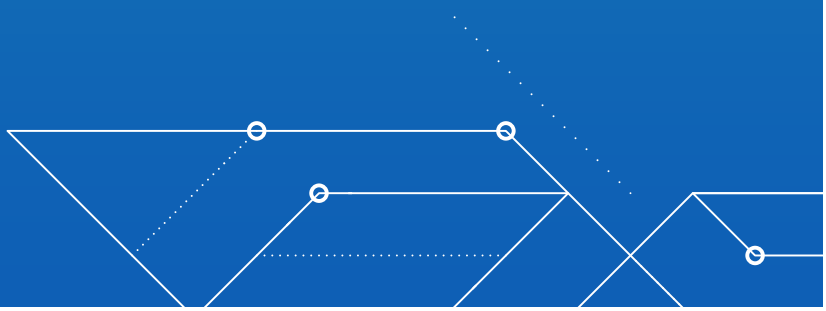
Percona XtraDB Cluster (PXC) migration is a specialized form of a rolling upgrade, designed for high-availability topologies. The process is similar to a minor version upgrade, but with a few critical differences to ensure the cluster remains stable and consistent during the transition. A rolling upgrade is the standard method for PXC, where you update one node at a time while the others handle the workload.



A. The Rolling Upgrade Process

To perform a rolling upgrade of a PXC cluster, follow these steps on each node, one at a time, until the entire cluster is running the new version.

1. **Synchronize Nodes:** Ensure all nodes in the cluster are synchronized and healthy before starting the upgrade on the first node.
2. **Stop the Service:** Shut down one of the 8.0 nodes. A common best practice is to start with a secondary node to minimize impact.
3. **Remove Old Packages:** Uninstall the existing PXC 8.0 packages. It is crucial not to delete the data directory during this step, as it contains the cluster's data.
4. **Install New Packages:** Install the Percona XtraDB Cluster 8.4 packages on the node.
5. **Start the Service:** Start the mysqld service with the new packages. The cluster will handle the data directory upgrade automatically. The node will either upgrade in-place during startup or perform a State Snapshot Transfer (SST) from another node to get a consistent copy of the data.
6. **Repeat:** Once the first node is successfully running on 8.4 and has rejoined the cluster, repeat the process for the next node. Continue this procedure for all nodes until the entire cluster is on PXC 8.4.



B. State Snapshot Transfer (SST) Improvements

Percona XtraDB Cluster 8.4 introduces a significant improvement to the SST process by implementing the Clone plugin. The Clone SST method leverages MySQL's native cloning capabilities to transfer data from a donor node to a joiner node. This method is faster and more resource-efficient than traditional SST methods like xtrabackup or rsync. This makes adding or re-syncing a node to a large cluster a much quicker and less intrusive operation, which is a key benefit for the migration process.

However, Clone cannot be used to perform an in-place upgrade between major versions. Instance cannot be cloned from a different MySQL server series. For example, you cannot clone between MySQL/Percona Server 8.0 and MySQL 8.4, but you can clone within a series, such as MySQL 8.4.1 and MySQL 8.4.13. This means that the Clone SST method cannot be used during a major version upgrade.

C. Key Challenges in a Mixed-Version Cluster

While PXC 8.0 and PXC 8.4 nodes can coexist in a mixed-version cluster, there are specific DDL (Data Definition Language) operations that can cause inconsistencies. The Galera protocol itself is compatible, but certain DDL enhancements in 8.4 can fail on an 8.0 node, leading to its eviction from the cluster.

A prime example is the support for **foreign keys referencing non-unique or partial keys**. In MySQL 8.4, this operation is allowed, but in 8.0, it is not. If an administrator runs a

CREATE TABLE statement with such a foreign key on an 8.4 node, it will succeed. However, because DDL statements are replicated as a transactional operation (TOI) and must execute identically on all nodes, the statement will fail on the 8.0 node. The cluster's consistency voting protocol will detect this difference and evict the 8.0 node.

To mitigate this risk, it is highly recommended to perform all DDL statements on the oldest version node (the 8.0 node) in the cluster during the rolling upgrade. This ensures that only DDL that is compatible with all versions in the cluster is executed. Another change to be aware of is that PXC 8.4 uses a **keyring component** instead of the keyring plugin, though the documentation notes that a donor node using the plugin can still work with an 8.4 node using the component, ensuring compatibility during the rolling upgrade.

VIII. Phase 2.6: Replication Topology Upgrades and Challenges

Upgrading a single MySQL server is a well-defined process, but migrating a production replication topology requires a more nuanced approach to ensure high availability and data consistency. A rolling upgrade is the only viable strategy to minimize downtime.

A. The Rolling Upgrade Strategy

The official documentation and best practices from companies like Percona recommend a specific order for upgrading servers in a replication topology to ensure continuous operation. The process is as follows:

1. Upgrade Replicas First:

Begin the upgrade with the replicas, starting with the ones that are farthest away from the source in a multi-layered topology. This allows the newly upgraded replicas to catch up with the older version of the source server without causing an outage.

2. Monitor Replication:

During this phase, it is critical to monitor the replication status to ensure that the upgraded replicas are correctly pulling and applying changes from the older source server.

3. Perform a Switchover:

Once all replicas are on the new version (8.4), a planned switchover is performed. Stop all client updates to the original source, wait for at least one replica to apply all pending changes, and then promote one of the upgraded replicas to become the new source.

4. Upgrade the Old Source:

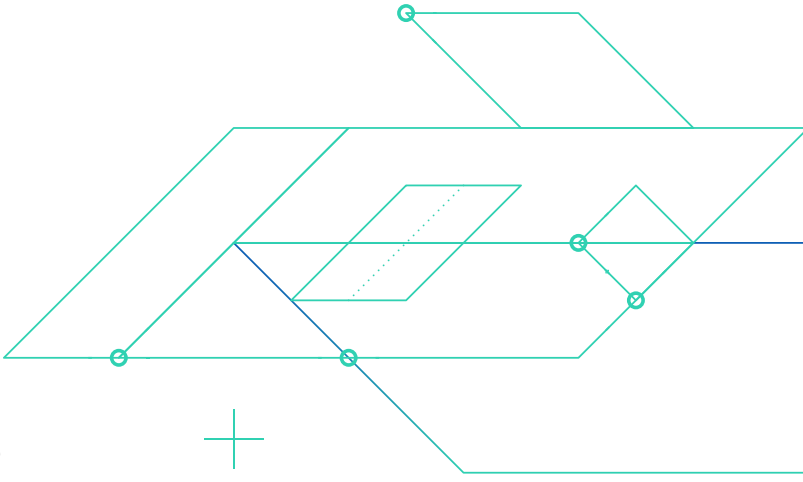
The original source server is now an isolated 8.0 instance. It can then be upgraded following the single-server procedure and reintroduced into the new replication topology as a replica of the new 8.4 source.

This method is applicable to standard replication and is also the recommended approach for upgrading Percona XtraDB Cluster and MySQL InnoDB Cluster, ensuring minimal disruption.

B. Replication-Specific Breaking Changes

The upgrade from MySQL 8.0 to 8.4 is not seamless for replication due to several breaking changes.

- **Terminology Removal:**
  The most significant and impactful change is the complete removal of the MASTER and SLAVE terminology from SQL syntax and status variables. This is a breaking change for any scripts, tools, or applications that rely on the old syntax.
  - START SLAVE becomes START REPLICA
  - SHOW SLAVE STATUS becomes SHOW REPLICA STATUS
  - CHANGE MASTER TO becomes CHANGE REPLICATION SOURCE TO
- **Authentication for Replication Users:**
  In MySQL 8.4, the deprecated mysql_native_password authentication plugin is disabled by default. This can cause replication to fail if the replication user on the source was created using this authentication method. The replication user must be recreated or altered to use the more secure caching_sha2_password plugin, ensuring connectivity after the upgrade.
- **Function and Variable Removals:**
  The WAIT_UNTIL_SQL_THREAD_AFTER_GTIDS() function, which was deprecated in 8.0, has been removed and must be replaced with WAIT_FOR_EXECUTED_GTID_SET().14 Similarly, the expire_logs_days system variable has been removed in favor of binlog_expire_logs_seconds.



Downgrade

You cannot perform a direct in-place rollback from MySQL 8.4 to 8.0.

The only supported method for downgrading between major versions like this is to restore a backup taken before the upgrade was performed.

This is because MySQL 8.4 introduces significant changes to the data dictionary and internal file formats that are not backward-compatible with MySQL 8.0. Trying to start a MySQL 8.0 server with data files from an 8.4 installation will result in errors and will not work.

Supported Downgrade Methods

Restore from a pre-upgrade backup: This is the safest and most reliable method. Before attempting any major version upgrade, you should always create a full logical or physical backup of your data. If the upgrade fails or you need to revert, you can simply restore this backup to a clean MySQL 8.0 installation.

Logical Dump and Load: If you don't have a backup, you can use a logical dump and load process. This involves exporting your data from the MySQL 8.4 instance using a tool like mysqldump or mysqlpump and then importing it into a new, clean MySQL 8.0 instance. This method avoids the physical file format incompatibility, but it's a slower process and may encounter issues if the 8.4 database uses features or syntax not supported by 8.0.

IX. Phase 3: Post-Migration Validation and Optimization

The upgrade is not complete once the migration method is finished. A series of validation and optimization steps must be performed to ensure the new environment is stable and performant.

A. Operational Validation Checklist

1. Server Status: Verify that the new MySQL 8.4 server is running correctly and that the version is as expected.

2. Error Log Review: Scrutinize the MySQL error log for any warnings or errors that occurred during the upgrade or restart process.

3. System and Data Dictionary Upgrade: Confirm that the mysql_upgrade process completed successfully.

4. Connectivity: Test that all applications and clients can successfully connect to the new server using the correct authentication methods and updated replication terminology.

B. Performance Tuning and Optimization

Do not simply copy the old my.cnf file to the new server. As established, MySQL 8.4's new defaults are optimized for modern hardware and workloads. The previous configuration may now be suboptimal or even detrimental. The new values for variables like *innodb_io_capacity* and *innodb_log_buffer_size* are a direct response to the capabilities of modern hardware. The DBA should reevaluate their configuration, comparing it against the new defaults and making adjustments accordingly to maximize performance. Percona's pt-variable-advisor tool, which has been updated for MySQL 8.4, can help with this process.

C. Application-Level Verification

The application must be thoroughly tested in the new environment. Re-run the performance benchmarks established in Phase 1 and compare the new metrics against the baseline. This will quantify the performance impact of the upgrade and highlight any regressions that require further investigation.

D. Troubleshooting Common Post-Upgrade Issues

Even with meticulous planning, issues can arise. The table below lists the most common post-upgrade problems and their mitigation strategies.

Change	Problem	Mitigation Strategy
mysql_native_password default disabled	Application connections fail with authentication errors.	Re-enable the plugin in the my.cnf file or, for a more permanent solution, update user accounts to caching_sha2_password using ALTER USER.
MASTER/SLAVE terminology removed	Custom replication scripts or commands fail.	Update all operational scripts and application code to use the new REPLICA and SOURCE terminology.
AUTO_INCREMENT on FLOAT/DOUBLE removed	Upgrade fails with an error on a specific table.	Convert the data type of the affected column to a numeric integer type before the migration begins.

A comprehensive approach to an upgrade requires a full-stack perspective. The util.checkForServerUpgrade() tool is a crucial part of this because it identifies cross-layer dependencies before they become a problem. A successful migration is not just a database-level event; it is a collaborative effort involving database administrators, application developers, and systems engineers to ensure every layer of the stack is compatible and optimized for the new environment.

X. Appendices: Essential Reference Material

Table A: MySQL 8.0 vs. MySQL 8.4 Feature and Configuration Differences

Category	MySQL 8.0	MySQL 8.4
Authentication Default	caching_sha2_password enabled by default; mysql_native_password enabled by default.	caching_sha2_password enabled by default; mysql_native_password disabled by default.6 The default_authentication_plugin variable is removed.
Replication Terminology	MASTER/SLAVE terminology used in SQL syntax and variables.	All MASTER/SLAVE terminology is removed and replaced with REPLICA/SOURCE.
New Reserved Words	N/A	MANUAL, PARALLEL, QUALIFY, TABLESAMPLE are new reserved words.
Data Type Restrictions	AUTO_INCREMENT is allowed on FLOAT and DOUBLE columns.	AUTO_INCREMENT is no longer allowed on FLOAT and DOUBLE columns.
InnoDB io_capacity	Default is 200.	Default is 10000.
InnoDB log_buffer_siz e	Default is 16 MiB.	Default is 64 MiB.
InnoDB adaptive_hash_index	Default is ON.	Default is OFF.
Replication Functions	WAIT_UNTIL_SQL_THREAD_AFTER_GTIDS() is available.	WAIT_UNTIL_SQL_THREAD_AFTER_GTIDS() is removed; use WAIT_FOR_EXECUTED_GTID_SET() instead.
Binlog Variables	expire_logs_days is available.	expire_logs_days is removed; use binlog_expire_logs_seconds instead.
Memcached Variables	daemon_memcached and innodb_api variables are available.	All daemon_memcached and innodb_api variables are removed.

Table B: Key Commands and Options for mysqlsh and mydumper for Large-Scale Migrations

Tool	Dump/Export Command	Key Options and Purpose	Load/Import Command	Key Options and Purpose
MySQL Shell	util.dumpInstance(...)	threads: 8 for parallelism,	compression: 'zstd' for file size reduction.	
MySQL Shell	util.loadDump(...)	threads: 8 for parallelism,	excludeUsers: [...] to avoid user creation.	
MyDumper	mydumper...	--threads=8 for parallelism,	--chunk-filesize=1G to split large tables,	--compress=zstd for optimal compression.
myloader...	myloader...	--threads=4 for parallel import,	--queries-per-transaction=1500 to optimize commits,	--optimize-keys for faster index creation.

References

1. Percona documentation: https://docs.percona.com/percona-server/8.4/upgrade.html

2. Database Migrations - Percona https://www.percona.com/resources/database-migrations

3. Percona Toolkit — An all-around solution to all of your MySQL troubles | by Shadowfax https://shadowfax-in.medium.com/percona-toolkit-an-all-around-solution-to-all-of-your-mysql-troubles-c33610ab3c70

4. Percona XtraBackup 8.4 Documentation https://docs.percona.com/percona-xtrabackup/8.4/index.html

5. MySQL 8.4 Reference Manual: 3.3 Upgrade Best Practices - MySQL https://dev.mysql.com/doc/refman/8.4/en/upgrade-best-practices.html

6. Upgrade from 8.0 to 8.4 overview - Percona Server for MySQL https://docs.percona.com/percona-server/8.4/upgrade.html

7. 8 Changes in MySQL 8.4.0 (2024-04-30, LTS Release) - Oracle Help Center https://docs.oracle.com/cd/E17952_01/mysql-8.4-relnotes-en/news-8-4-0.html

8. 1.4 What Is New in MySQL 8.4 since MySQL 8.0 https://dev.mysql.com/doc/refman/8.4/en/mysql-nutshell.html

9. Changes in MySQL 8.4.4 (2025-01-21, LTS Release) https://dev.mysql.com/doc/relnotes/mysql/8.4/en/news-8-4-4.html

10. Upgrade the database major version in-place | Cloud SQL for MySQL https://cloud.google.com/sql/docs/mysql/upgrade-major-db-version-inplace

11. Major Version Upgrade - Oracle Help Center https://docs.oracle.com/en-us/iaas/mysql-database/doc/major-version-upgrade.html

12. 1.5 Server and Status Variables and Options Added, Deprecated, or Removed in MySQL 8.4 since 8.0

https://dev.mysql.com/doc/refman/8.3/en/added-deprecated-removed.html

13. MySQL Shell 8.4 : 11.1 Upgrade Checker Utility https://dev.mysql.com/doc/mysql-shell/8.2/en/mysql-shell-utilities-upgrade.html

14. Upgrading MySQL 8.0 BugFix to MySQL 8.4 LTS - Oracle Help Center https://docs.oracle.com/en/database/mysql/heatwave-aws/hw-aws-upgrading-mysql-8.0- 8.4.html

15. 3.5 Changes in MySQL 8.4 https://dev.mysql.com/doc/en/upgrading-from-previous-series.html

16. MySQL 8.4 Support in Percona Toolkit 3.7.0 https://percona.community/blog/2025/01/06/mysql-8.4-support-in-percona-toolkit-3.7.0/

17. Upgrade the database major version by migrating data | Cloud SQL for MySQL https://cloud.google.com/sql/docs/mysql/upgrade-major-db-version-migrate

18. How to Safely Upgrade InnoDB Cluster From MySQL 8.0 to 8.4 - Percona https://www.percona.com/blog/how-to-safely-upgrade-innodb-cluster-from-mysql-8-0-to-8-4/

19. MySQL 8.4 Reference Manual :: 3.7 Upgrading MySQL ... - MySQL https://dev.mysql.com/doc/refman/8.4/en/upgrade-binary-package.html

20. MySQL 8.4 Reference Manual :: 2 Installing MySQL https://dev.mysql.com/doc/mysql-installation-excerpt/8.3/en/

21. https://dev.mysql.com/doc/refman/8.4/en/upgrade-in-place.html

22. 6.4.5 mysql_upgrade — Check and Upgrade MySQL Tables https://dev.mysql.com/doc/en/mysql-upgrade.html

23. MySQL Shell 8.4 :: 11.5 Instance Dump Utility, Schema ... - MySQL https://dev.mysql.com/doc/mysql-shell/8.3/en/mysql-shell-utilities-dump-instance-schema.html

24. Multithreaded Data Dumps With MySQL Shell - Oracle Blogs https://blogs.oracle.com/mysql/post/multithreaded-data-dumps-with-mysql-shell

25. MySQL Shell 8.0 :: 11.4 Parallel Table Import Utility https://dev.mysql.com/doc/mysql-shell/8.0/en/mysql-shell-utilities-parallel-table.html

26. Migrate very large databases to Amazon Aurora MySQL using … https://aws.amazon.com/blogs/database/migrate-very-large-databases-to-amazon-aurora-mysql-using-mydumper-and-myloader/

27. MySQL Shell 8.4 :: 11.6 Dump Loading Utility https://dev.mysql.com/doc/mysql-shell/8.3/en/mysql-shell-utilities-load-dump.html

28. Upgrade from 8.0 to 8.4 overview https://docs.percona.com/percona-server/8.4/upgrade.html

29. How to Safely Upgrade InnoDB Cluster From MySQL 8.0 to 8.4 https://www.percona.com/blog/how-to-safely-upgrade-innodb-cluster-from-mysql-8-0-to-8-4/

30. Upgrade Percona XtraDB Cluster https://docs.percona.com/percona-xtradb-cluster/8.4/upgrade-guide.html

31. docs.percona.com, https://docs.percona.com/percona-xtradb-cluster/8.4/upgrade-guide.html#:~:text=Shut down an 8.0 node,service%3A Start the node again.

32. Percona Distribution for MySQL 8.4.4 using Percona XtraDB Cluster https://docs.percona.com/percona-distribution-for-mysql/8.4/release-notes-pxc-8.4.4.htm

33. Percona XtraDB Cluster 8.4.4-4 https://docs.percona.com/percona-xtradb-cluster/8.4/release-notes/8.4.4-4.html

34. Versions compatibility - Percona Operator for MySQL https://docs.percona.com/percona-operator-for-mysql/pxc/versions.html

35. Releases · sysown/proxysql - GitHub https://github.com/sysown/proxysql/releases

36. MySQL 8 Password Management & Plugin Switching - Mydbops https://www.mydbops.com/blog/password-management-in-mysql-8

37. Building Production-Ready MySQL Master-Slave Replication with Docker and ProxySQL: A Complete Guide | by Poulastaa | Sep, 2025 | Medium https://medium.com/@poulastaa/building-production-ready-mysql-master-slave-replicatio n-with-docker-and-proxysql-a-complete-guide-70007be745ee

38. Stop Sending Traffic to Broken MySQL Replicas with ProxySQL - Mydbops https://www.mydbops.com/blog/prevent-proxysql-from-directing-traffic-to-broken-mysql-r eplica

39. Percona Monitoring and Management - Database Tools https://www.percona.com/software/database-tools/percona-monitoring-and-management

40. PMM - Percona Monitoring and Management https://percona.community/projects/pmm/

PERCONA®