MyWebinar #54

Upgrading to MySQL 8.4 LTS

A Complete Walkthrough — Planning, Execution, Validation & Cutover

8.0 ⇄ 8.4 LTS

End of Life — Long-Term Support

Praveen GR
Mydbops

Friday, 28 August 2026
Live Webinar

Mydbops
Scaling Databases

YOUR SPEAKER

Praveen GR

Mydbops

Works with production MySQL estates across on-prem and cloud, focusing on upgrades, replication topologies and high availability.

Focus for this session

Real-world 8.0 → 8.4 LTS upgrade paths -what breaks, what to test, and how to cut over with near-zero downtime.

About Mydbops

ISO-certified managed database services and consulting for MySQL, MariaDB, MongoDB, PostgreSQL and TiDB.

ABSTRACT

What this session covers

End-to-end walkthrough

A complete path from MySQL 8.0 to MySQL 8.4 LTS, built from real production upgrade work — not a feature tour.

Four phases in depth

Planning, execution, validation and cutover — with the decision points and gates that matter at each stage.

On-prem and cloud

Applicable to self-managed deployments and to AWS RDS for MySQL and Aurora MySQL-Compatible environments.

Take-home goal: leave with a checklist you can run against your own estate — not just an understanding of what changed in 8.4.

AGENDA

What we will walk through

01 Why upgrade now

02 MySQL's new release model

03 Pre-upgrade assessment

04 QA validation phase

05 Minimal-downtime strategies

06 Replication-based cutover

07 Rollback realities

08 Post-upgrade validation & takeaways

01

SECTION ONE

Why upgrade now

MySQL 8.0 has reached End of Life, and staying put is no longer free — technically or financially.

SECTION 01 · URGENCY

Why plan the upgrade now

MySQL 8.0 has reached End of Life

The Community edition no longer receives public bug fixes or security patches.

AWS RDS 8.0 standard support has ended

Instances still running 8.0 now fall under Extended Support automatically.

Extended Support is billed on top

A surcharge is added to your instance cost for as long as you stay on 8.0.

Delay compounds the cost

Security exposure, compliance risk and future migration complexity all grow the longer you wait.

AWS Extended Support: what it means for your bill

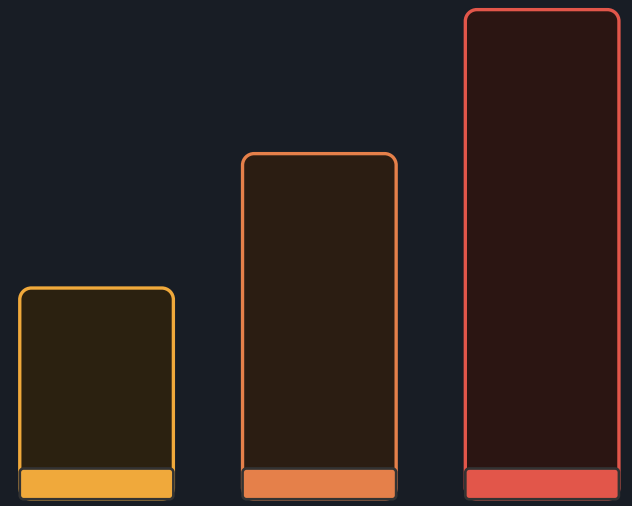
A per-vCPU, per-hour surcharge
Extended Support keeps 8.0 running — at an added hourly rate that increases year over year.

Designed as a forcing function
Staying on 8.0 becomes progressively more expensive than upgrading. That is the intent.

Applies to RDS and Aurora alike
Both RDS for MySQL and Aurora MySQL-Compatible instances are in scope.

Not just a DBA concern
Budget owners should see this line item as a deadline, not a recurring cost of doing business.

Cost of staying on 8.0



Illustrative — the surcharge rate steps up each year of Extended Support.

SECTION 02 · CONTEXT

Understanding MySQL's new release model

Innovation releases

Short-lived · frequent · new features first

8.1 8.2 8.3

Skip these in production

LTS releases

Long-lived · stable · production track

8.0 ⇄ 8.4 LTS

Target LTS-to-LTS

Since MySQL 8.1, Oracle ships on two tracks

Innovation releases move fast and expire fast; LTS releases are the ones you run in production.

8.4 is the current LTS line

It is the direct successor to 8.0 for anyone who values stability over early features.

SECTION 02 · TARGET

Why 8.4 specifically

Longer support window

An LTS line is supported for years, not months — unlike the Innovation releases it sits alongside.

Fewer behavioural surprises

Once you are on 8.4, patch versions change far less between releases than on the Innovation track.

Smaller jump than 5.7 → 8.0

Both are LTS series and the surface area is narrower than the last major migration many teams remember.

The catch: features deprecated across several 8.0 minor versions are now fully removed in 8.4. A smaller jump is not a free one — everything you were warned about has finally been taken away.

THE PLAN

High-level upgrade roadmap

PLAN & VALIDATE | BUILD & SYNC | SWITCH & STABILISE

01
Assessment
Upgrade checker and blocker remediation

02
QA validation
Upgrade, testing and formal sign-off

03
Production build
Parallel or blue-green environment

04
Replication sync
Sync data and validate consistency

05
Cutover
Switch traffic to the new 8.4 endpoint

06
Stabilise
Post-upgrade monitoring and clean-up

Each phase gates the next. Nothing moves forward until the previous phase has signed off — and phases 1 and 2 loop internally until their reports come back clean.

02

SECTION TWO

Pre-upgrade assessment

Find every incompatibility on paper before a single binary is replaced.

The upgrade checker - your first gate

```
mysqlsh >
util.checkForServerUpgrade()
```

Mandatory first step. Nothing else in this deck should start before the report is clean.

The report is prioritised

Errors	Warnings	Notices
Must be fixed	Review each	Informational

What it does

Scans your server for incompatibilities and produces a prioritised report of errors, warnings and notices.

What it compares

Your current database against the target MySQL version, surfacing potential upgrade issues before you commit.

Why it matters

It catches breaking changes before they cause a failed upgrade or a broken production server — not something you want to discover mid-cutover.

SECTION 02 · PHASE 1

How the upgrade checker works

```
util.checkForServerUpgrade('user@host:3306',
                           {"targetVersion":"8.4.0"})
```

targetVersion

The MySQL version to check against — set it explicitly to 8.4.x rather than relying on a default.

include / exclude

Run or skip specific checks when you need a targeted re-run instead of a full scan.

Non-interactive CLI

The same check runs headless, so it can be wired into CI and automation pipelines.

Best practice: run against a copy of production data, resolve all errors, review every warning, then re-run until the report is clean.

Common blockers moving 8.0 → 8.4

Removed variables	innodb_log_file_size and innodb_log_files_in_group are replaced by innodb_redo_log_capacity; every slave_* variable is renamed replica_*.
Authentication	mysql_native_password is now disabled by default — old drivers and application accounts will fail to connect.
Privileges	The deprecated SET_USER_ID privilege is removed, replaced by SET_ANY_DEFINER and ALLOW_NONEXISTENT_DEFINER.
Foreign keys	8.4 requires a unique key on the referenced columns in the parent table — schemas that got away with it before will not.
Data types	AUTO_INCREMENT on FLOAT and DOUBLE columns is completely removed and now raises an error.

Replication-specific blockers

MASTER SLAVE ⇄ SOURCE REPLICA

Terminology change runs through SQL syntax and status variables alike

Old terminology is gone, not deprecated

MASTER/SLAVE in SQL syntax and status variables has been replaced by SOURCE/REPLICA throughout.

Deprecated replication syntax is removed entirely

Several replication features flagged in earlier versions no longer parse at all in 8.4.

Your automation is in scope

Home-grown replication tooling, monitoring scripts and CI pipelines referencing the old syntax will break — not warn.

SET_USER_ID removal bites during the upgrade itself

It can break cross-version replication — exactly the topology you rely on while migrating.

SECTION 02 · REMEDIATION

Unblocking the upgrade

01

Triage the output

Errors first, warnings second. Do not batch them together.

02

Update configuration

Replace removed variables with their new equivalents before the 8.4 binary is installed.

03

Update code & scripts

Application, ORM and DBA tooling that references deprecated syntax.

04

Re-run the checker

After every remediation round, until the report comes back clean.

This is a loop, not a checklist. Expect several rounds — each fix can surface the next one underneath it.

MySQL

03

SECTION THREE

QA validation phase

Do the upgrade for real — somewhere it cannot hurt you — and prove it works under load.

Upgrading in QA

Mirror production topology

Perform the actual upgrade first in a QA or staging environment that matches production structure — replicas, proxies and all.

Use a production-like data snapshot

Synthetic or minimal datasets hide the problems you are trying to find. Volume and data shape both matter.

This is where the rest surfaces

Blockers that no static check can catch appear here — behaviour, timing and interaction issues rather than syntax.

The QA environment is the last place a failure is cheap.

Everything after this point costs a change window, a rollback plan, or an incident.

Load testing

Test under production-like traffic

Match real traffic patterns and concurrency, not a smoke test. Peak behaviour is what you are validating.

Compare metrics pre- and post-upgrade

Baseline before you upgrade so the comparison is evidence rather than opinion.

Watch for behavioural regressions

Optimiser and InnoDB behaviour changes between versions are the usual cause of a query that suddenly got slower.

Metrics to capture on both sides

- Query latency
- Throughput
- Connection handling
- Replication lag under load

Compatibility testing

Database objects

Application queries, stored procedures, triggers and views — everything that carries SQL you did not write today.

ORMs and drivers

Especially around the authentication plugin change. An old connector can pass every SQL test and still fail to connect.

Third-party tooling

Monitoring agents, backup tools and migration or CI scripts that reference the old replication syntax.

Compatibility is not only about your own code. Most upgrade surprises arrive through something a vendor or a script wrote years ago.

Resolving concerns before sign-off

Every issue surfaced during load or compatibility testing must be resolved before proceeding. No exceptions.

Triage each finding by type

Performance regression, query incompatibility, driver or authentication failure, or an application-level break, they need different owners.

Fix, re-test, re-validate

Against the same test scenarios that found the problem, until results come back clean.

Do not carry open concerns forward

Anything unresolved at QA sign-off becomes a production incident later, with an audience.

SECTION 03 · GATE

QA sign-off checklist

Formal gate before production planning begins

- Upgrade checker report is clean
- Load test results within acceptable thresholds
- Compatibility test suite passed, all concerns resolved
- Rollback and backup plan documented and reviewed
- Stakeholder approval recorded — App, DBA and SRE

Five boxes. All of them.

A partial sign-off is not a sign-off — it is a decision to accept risk without naming it. Record who approved and when.

MySQL

04

SECTION FOUR

Deployment & cutover

Build the new environment alongside the old, sync it, and switch with near-zero downtime.

SECTION 04 · STRATEGY

Minimising downtime: two approaches

Goal: a near-zero downtime cutover to production. Both routes below get you there — they differ in how much you orchestrate yourself.

Parallel environment

Build the new 8.4 environment yourself, alongside the existing 8.0 one, and sync it with replication.

Full control · more manual orchestration

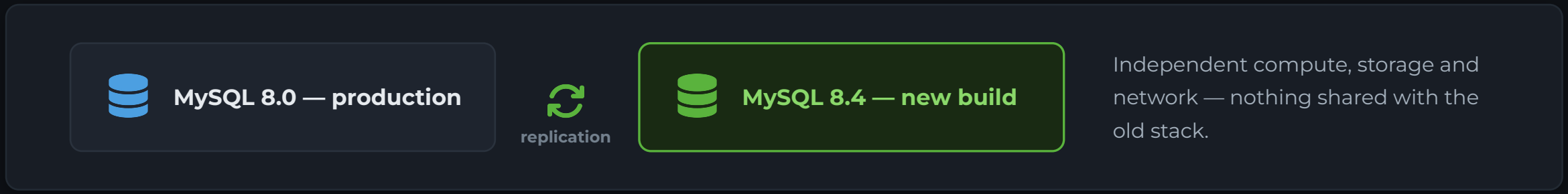
Blue-Green deployment

Use the cloud-native managed workflow — AWS RDS and Aurora Blue/Green Deployments handle the sync for you.

Managed · fewer moving parts you own

SECTION 04 · OPTION A

Parallel environment approach



Stand up a fully independent 8.4 environment

Compute, storage and network provisioned separately, running alongside the existing 8.0 environment.

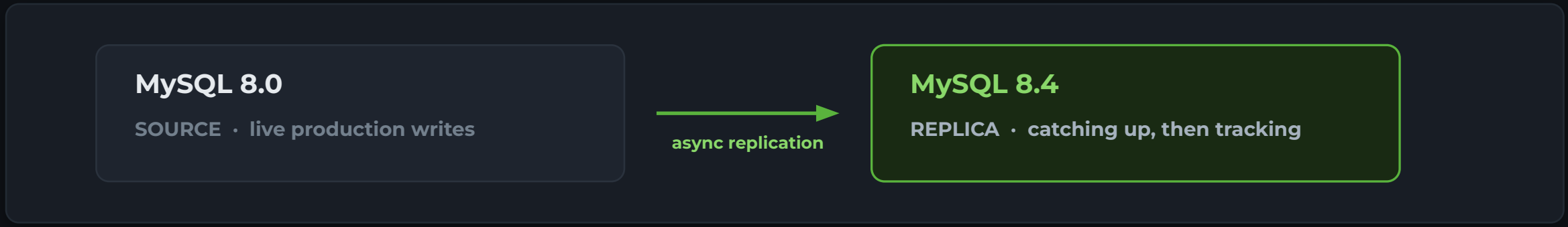
Sync the data via replication

The new environment tracks production continuously until you are ready to switch — covered in the next slide.

Control comes at the price of orchestration

You own configuration, testing and timing end to end — more work than a managed blue/green, and more say in how it runs.

Async replication: old to new



New 8.4 environment becomes a replica of 8.0 production

Configure asynchronous replication with the existing production instance as the source.

This direction is required, not preferred

Old (8.0) must be the source and new (8.4) the replica — the reverse is not version-compatible.

Let it fully catch up, then keep tracking

The replica should be continuously following production writes well before you plan the cutover window.

SECTION 04 · OPTION B

Blue-Green deployment (AWS-native)

BLUE

Current production

MySQL 8.0 · serving live traffic

GREEN

Staging environment

MySQL 8.4 · synced automatically

≤ 5s

Typical switchover downtime for single-Region configurations

A managed workflow for 8.0 → 8.4 on RDS

Covers pre-upgrade checks, minimal-downtime cutover and a rollback safety net in one process.

Upgrade without touching production

When you are ready, promote the green environment to become production. Traffic follows the switchover.

AWS manages the sync for you

Replication between blue (current) and green (new version) is handled automatically throughout.

SECTION 04 · GATE

Validating replication before cutover

Monitor replication lag continuously

Note the terminology change — it is SHOW REPLICA STATUS in 8.4, and your old scripts will not know that.

```
mysql >
SHOW REPLICA STATUS\G
```

Run data consistency and checksum validation

Compare source and replica directly. Lag reaching zero is not the same as the data being identical.

Test the application against the replica read-only

Confirm compatibility on the real 8.4 instance before you promote it, while a mistake is still reversible.

⚠ Do not schedule the cutover window until all three checks pass consistently — not once, but across a full traffic cycle.

SECTION 04 · EXECUTION

Cutover planning & execution

01

Schedule the window

Define the cutover window with explicit rollback checkpoints agreed in advance.

02

Reach zero lag

Briefly freeze or quiesce writes — or let the managed blue/green switchover do it for you.

03

Redirect traffic

Point the application at the new 8.4 endpoint via DNS or a connection endpoint switch.

04

Verify immediately

Run application health checks the moment traffic lands, before anyone stands down.

The moment 8.4 accepts its first write, the next slide applies. Know your rollback position before you open the gate.

SECTION 04 · REALITY CHECK

Rollback realities

Direct rollback is not feasible once 8.4 has accepted writes.

MySQL replication requires the replica to be the same version or higher than its source. You cannot replicate from 8.4 back down to 8.0 — this is a structural limit of the replication protocol, not a setting you forgot to enable.

What rollback actually relies on

Pre-cutover backups

Snapshots taken immediately before switchover — close enough in time to be genuinely useful.

Old environment on standby

Keep 8.0 alive and read-only for a defined retention window rather than tearing it down on day one.

A tested restore plan

A rehearsed restore or re-platform procedure — not a live reverse-replication plan that cannot exist.

05

SECTION FIVE

After the cutover

The upgrade is not done when traffic moves — it is done when the numbers hold and the old stack is retired.

Post-upgrade validation & monitoring

Confirm the baseline still holds

Application error rates, query latency and connection counts should sit within the numbers you captured before the upgrade.

Watch for optimiser plan changes

A query that was fine yesterday can pick a different plan on 8.4. Slow-query monitoring earns its keep in the first days.

Re-point everything around the database

Scheduled jobs, backups and monitoring or alerting must all target the new environment — silently failing backups are the classic miss.

Decommission on a date, not on a feeling

Retire the old environment only after the agreed retention and rollback window has actually passed.

WRAPPING UP

Key takeaways

01	**This is urgent, not optional**	EOL and AWS Extended Support billing turn "sometime this year" into a dated decision.
02	**The upgrade checker is your first gate**	Resolve every Error before anything else starts. Re-run until the report is clean.
03	**Validate thoroughly in QA**	Load and compatibility testing, a formal sign-off, and every concern resolved — no exceptions.
04	**Blue-green or parallel + replication**	Either route gets you near-zero downtime. Pick by how much orchestration you want to own.
05	**Rollback is directional by design**	Plan backups and standby retention. Reverse replication from 8.4 to 8.0 is not an option.

Start with the checker this week. Everything else in this deck follows from what it tells you.

Questions?

Thank you for joining MyWebinar #54.

TReach out any time if you would like a second pair of eyes on your own 8.0 → 8.4 plan.

www.mydbops.com

www.mydbops.com/contact

linkedin.com/company/mydbops

Mydbops
Scaling Databases