

Running MariaDB When Nobody Wants to Be the DBA: *From Manual DBA Work to Safe Automation*



Roman Agabekov
Founder, **Releem**

Who am I?

- Founder of **Releem**
- **MariaDB / MySQL**
 - Developer
 - Support engineer
 - Reliability engineer
 - DBA
- **15+** years in **database** consulting

Agenda

1. Who are those teams
2. What databases do they use
3. The core problem
4. How we approached this problem
5. What changed in practice

When Database Management **Isn't** a Problem?

When you have a **DBA**

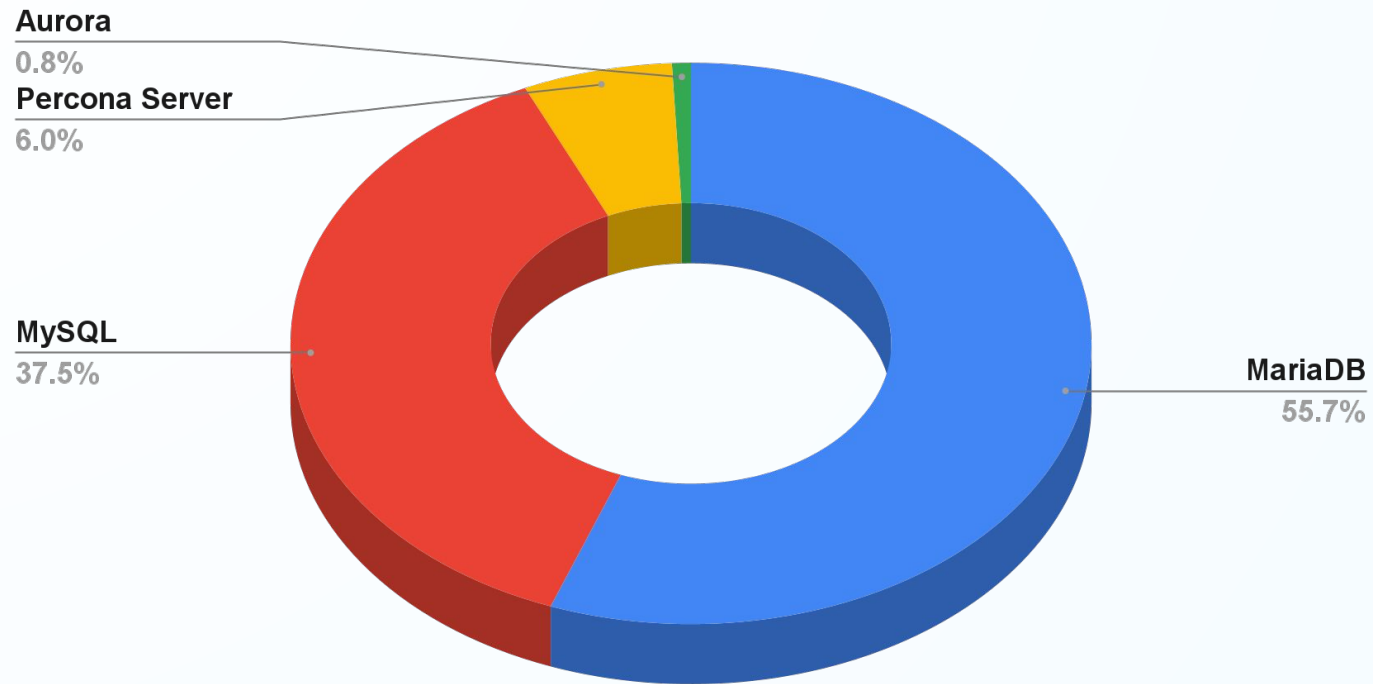


Who are those teams?

- **Shared hosting providers:**
 - no dedicated DBA,
 - infra engineer “also” owns DB
- **SaaS teams:**
 - no dedicated DBA,
 - developers own DB part-time

Based on Releem production statistics

What database do they use?



MariaDB dominates in teams without dedicated DBAs

Based on Releem production statistics

How small teams manage databases today

- Database work happens between feature work and incidents
- Issues are noticed late, often via user complaints
- Plenty of advice exists, but trust is unclear and experiments are costly and time-consuming

Fixing the wrong thing feels riskier than doing nothing

What tasks developers solve

PERFORMANCE

Slow queries
Database server configuration
Schema
OS settings

RESOURCE USAGE

CPU/RAM/Storage usage
Cost of instance

RELIABILITY

Deadlocks
Long-running queries

SECURITY

Database permissions
OS settings

How teams solve this today

- Monitoring / observability / APM tools
- Specialized tools for configuration tuning, query optimization, security....
- Scripts for ad-hoc checks
- Blogs, Documentation
- Ask LLMs for guidance

Challenge #1

Monitoring / observability tools

- built for DBAs
- Exposes raw signals, not decisions
- Requires deep context to interpret
- Hard to answer: “What should I do next?”

Challenge #2

Specialized tools for configuration tuning, query optimization, security...

- Fragmented
- Manual actions required

Challenge #3

- Manual execution
- Even safe changes require time and attention

The Core Problem

Without DBA experience:

Teams struggle to decide what to do next and what is safe to change.

Decisions and implementation are manual.

Experimentation is time-consuming and risky.

How we approached this problem

A **database advisor** that helps teams **decide what to focus on,**
with **optional automation** after review.



How advisor works

1. Analyze metrics, queries, db/os/instance settings, schema
2. Identify issues across performance, resources, reliability, security
3. Propose actions (config, query, schema,...)
4. Explains why this action matters
5. Helps teams apply and validate the change

Where automation makes sense

Instance Layer

Resource CPU/RAM recommendations
Storage recommendations

OS Layer

OS settings recommendations

Database Service Layer

Configuration tuning
Query optimization and index suggestions
Security recommendations
Schema recommendations

Workload data collection

- Memory utilization, CPU usage, and disk I/O statistics
- Database server configuration and status information
- Tables and schema names
- Table definitions, index structures, and index usage statistics
- Aggregated query statistics with normalized SQL (placeholders only), with query examples and execution plans (EXPLAIN)

From data to action

Detection

1. Workload data collection
2. Pattern detection
(rules, stats, heuristics, ML)

From data to action

Detection

1. Workload data collection
2. Pattern detection
(rules, stats, heuristics, ML)

Decision

1. Action hypotheses
(rules, stats, heuristics, optional LLM)
2. Impact estimation
(stats, heuristics, ML)
3. Expert system
(rules)
4. Validated actions (recommendations)
5. Explanation (rules, LLM)

From data to action

Detection

1. Workload data collection
2. Pattern detection
(rules, stats, heuristics, ML)

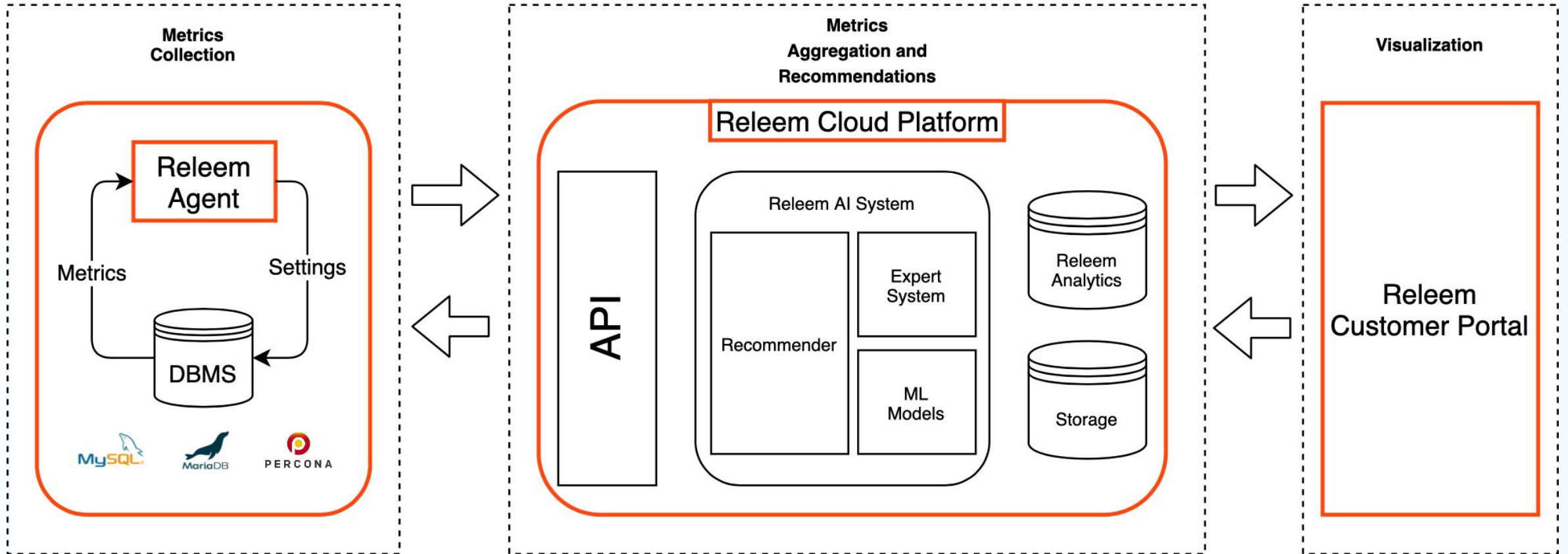
Decision

1. Action hypotheses
(rules, stats, heuristics, optional LLM)
2. Impact estimation
(stats, heuristics, ML)
3. Expert system
(rules)
4. Validated actions (recommendations)
5. Explanation (rules, LLM)

Execution

1. Execution
(manual or automated)
2. Feedback impact
(metrics)

System architecture



What teams see

Dashboard

One stop for all information needed

Server Info

| | | | | | |
|---------------|--------------|----------------|---------------|--------------|--------------|
| IP Address | OS version | DB Version | Instance Type | Agent Status | Memory Limit |
| 192.9.139.251 | Debian 11.11 | MariaDB 11.8.3 | On-premise | Connected | 16011 MB |

[Settings](#)

Releem Score

92%

Best Average Poor About Releem Score

Health Status

| | |
|-----------------|-------|
| System | (4/5) |
| MyISAM / InnoDB | (5/5) |
| Memory | (5/5) |
| Queries / Logs | (4/4) |

Health checks

Security Status

| | |
|-----------------------------|-----|
| Authentication & Access | 7/7 |
| System & Network Security | 2/3 |
| Data Integrity & Operations | 1/2 |

Security Checks

Recommended Configuration

Preparing configuration... coming soon

3 Unapplied recommendations

Current Configuration Impact

Configuration

System Metrics

| | | | |
|-------------|-----------|-----------|--------------|
| CPU 4 cores | RAM 16 GB | SWAP 0 GB | IOPS R/W |
| 5.75 % | 14.4 GB | 0 GB | 0.12 / 59.55 |

Last day

Database Metrics

| | | | |
|-------------------------------|----------------------------|----------------------------|--------------------------------|
| Latency ms | Queries per Second | Slow Queries | Aborted Clients |
| Day avg: 30.43 ↓ -7% last day | Day avg: 939 → 0% last day | Day total: 0 → 0% last day | Day total: 1203 ↓ -8% last day |

Queries & Schema

Query Analytics

Range: 2026.01.18 05:20 - 2026.01.18 05:30

| Queries | Count | Avg. Execution Time | Load On Total Time |
|---|-------|---------------------|--------------------|
| UPDATE `items` SET LAST_VALUE = ?, `last_timestamp` = ? WHERE `itemid` = ? | 88350 | 0.08 ms | |
| SELECT `itemid` FROM `items` WHERE `itemid` = ? FOR UPDATE | 88353 | 0.07 ms | |
| SELECT `s`.`sid`, `s`.`created`, `s`.`last_rid`, `s`.`hostid`, `s`.`dbms_version`, `s`... | 29588 | 0.18 ms | |
| SELECT ID, `tasks_type_id`, `timestamp_status` FROM `tasks` WHERE `sid` = ? AND STATUS != ? AND ST... | 29442 | 0.16 ms | |
| SELECT `uid`, `current_plan` FROM `users` WHERE `api_key` = ? | 29629 | 0.09 ms | |
| SELECT `il`.`id`, `i`.`itemid`, `i`.`last_value`, `i`.`last_timestamp`, `il`.`value_ty... | 14755 | 0.16 ms | |
| SELECT `i`.`itemid`, `il`.`name`, `i`.`history`, `il`.`value_type`, `il`.`preproc_type... | 14790 | 0.15 ms | |
| SELECT ID, `tasks_type_id` FROM `tasks` WHERE `sid` = ? AND STATUS = ? ORDER BY ID LIMIT ? | 14699 | 0.12 ms | |
| UPDATE `servers` SET `last_timestamp` = ? WHERE `sid` = ? | 14705 | 0.12 ms | |

Processes & Locks

Deadlocks Process List

What teams see

Deadlock Insight

More information with context

Deadlock Insights

🕒 Time: 2025.09.28 03:00:26
📄 Database name: test_db
📄 Tables: semaphore

Insert Same PK Deadlock

Plain `INSERT` is not idempotent when multiple workers try to write the same key at once. Two writers collide on the same PRIMARY key and deadlock.

Raw data output

```
2025-09-28 00:00:26 0x7ffa6578e700
*** (1) TRANSACTION:
TRANSACTION 632093048, ACTIVE 2 sec inserting
mysql tables in use 1, locked 1
LOCK WAIT 6 lock struct(s), heap size 1128, 3 row lock(s)
MySQL thread id 465734, OS thread handle 140718900717312, query id
264699324 localhost [REDACTED] Update
INSERT INTO `semaphore` (name, value, expire) VALUES
('field_info:bundle:slide:slide', '39405829268d8505e39e897.39520935',
'1759006854.2453')
*** (1) WAITING FOR THIS LOCK TO BE GRANTED:
RECORD LOCKS space id 5383 page no 142 n bits 72 index PRIMARY of table
```

How to fix it

1. Change the statement to one of these patterns so the second writer doesn't need to "win" a brand-new insert: `INSERT IGNORE`, `REPLACE`, or `INSERT ... ON DUPLICATE KEY UPDATE` if the inserted row may already exist.
2. Reduce the locked range by adding filters or more specific indexes.
3. Consider `READ COMMITTED` isolation level to reduce the range lock scope.

[Learn more](#)

ⓧ Close

What teams see

Optimized Query with recommendations

Examine the measurable impact

Query Inspection | *Optimised*

Database Name: **releemdb**

Count: **4566116**

Load On Total Time: **793.9**, s ↓

Avg. Execution Time: **0.18** ms ↓

```
SELECT
  `created`,
  `sid`
FROM
  `servers`
WHERE
  `uid` = ?
  AND `hostname` = ?
  AND `is_deleted` = ?
```

Query Digest

EXPLAIN

OVERVIEW

This query has been optimised. Reduced CPU load from **17.4%** to **0.3%**. Reduced query execution time from **2.089** ms to **0.1738702** ms. Reduced total query execution time by 95.6%

Recommendations

RECOMMENDATIONS

Create an index on one or more of these columns: `uid`, `is_deleted`, `hostname` for table `servers`

The columns used in the WHERE clause for this table are part of existing indexes. However, they can't be leveraged because they do not make what's called an 'index prefix'. To understand what an index prefix is, consider the following: an index on columns (A,B,C) can be used for query conditions on A, A and B, A and B and C, but not on B and C or C alone. All columns must be consecutive and to the left of an index in order to be considered by the server. An alternative in this case is to create a specific index, including the columns invoked in the query, in a valid order. Below is the corresponding SQL statement.

```
CREATE INDEX `idx_uid_is_deleted_hostname` ON
`servers`(`uid`,`is_deleted`,`hostname`) ALGORITHM=INPLACE LOCK=NONE;
```

Include additional conditions to match the index prefix

To leverage index `idx_sid_uid` on table `servers`, add conditions to the query including these columns: `sid`

Delete

Compatibility

DB: MariaDB, MySQL, Percona Server

OS: Linux, Windows

Cloud integration: AWS RDS, GCP Cloud SQL

6 MILLIONS

query
recommendations

What we've seen
in production

2025

53,000

config tuning
suggestions

**348
BILLIONS**

SQL queries
collected

**808
MILLIONS**
API requests

714,000
schema
recommendations

571,000
deadlocks detected
and analyzed

What changed

Teams started manage databases differently:

- Continuously check recommendations and apply them
- Reduced manual config, query optimization and schema checks work
- Fewer emergency incidents
- Database work became clear, not stressful

Any Questions?



r.agabekov@releem.com



@agabekovroman <https://twitter.com/agabekovroman>



<https://www.linkedin.com/in/roman-agabekov/>



<https://releem.com/>

