TECHNICAL WHITE PAPER

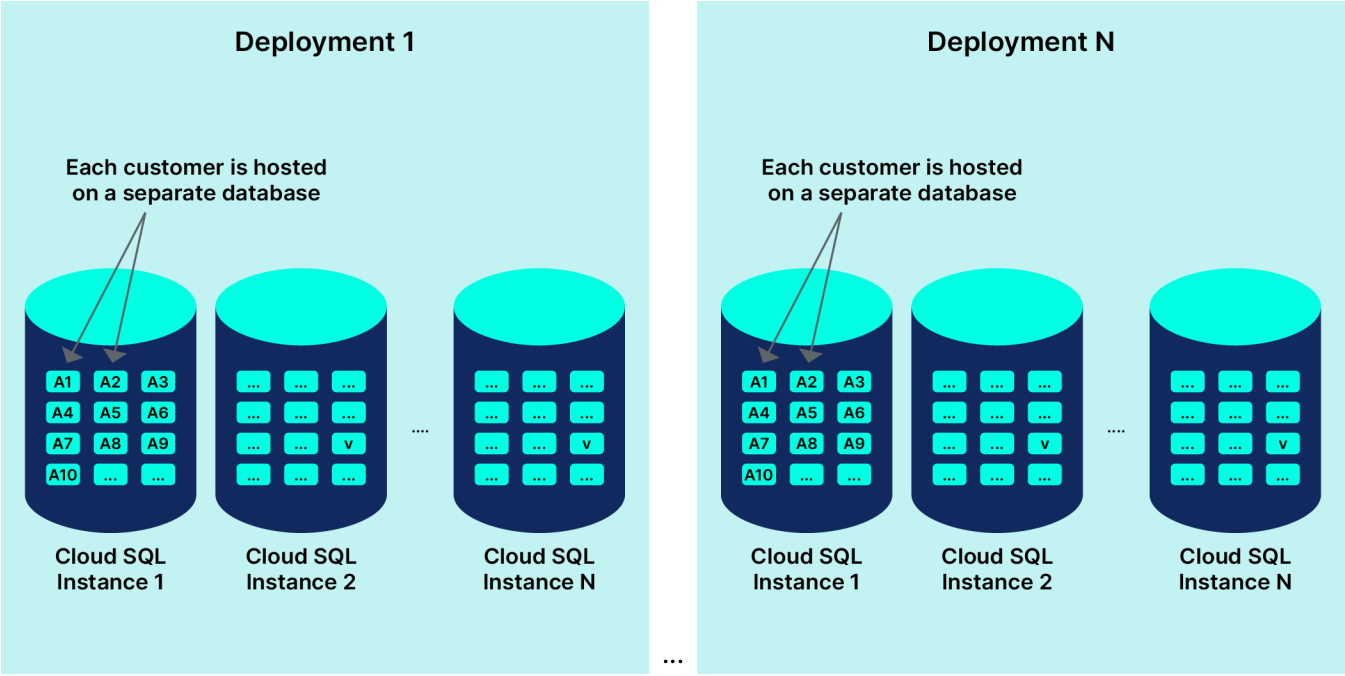
Insights from Engineering Robustness in Rubrik's Large Cloud SQL Fleet

Manjunath Chinni

Table of Contents

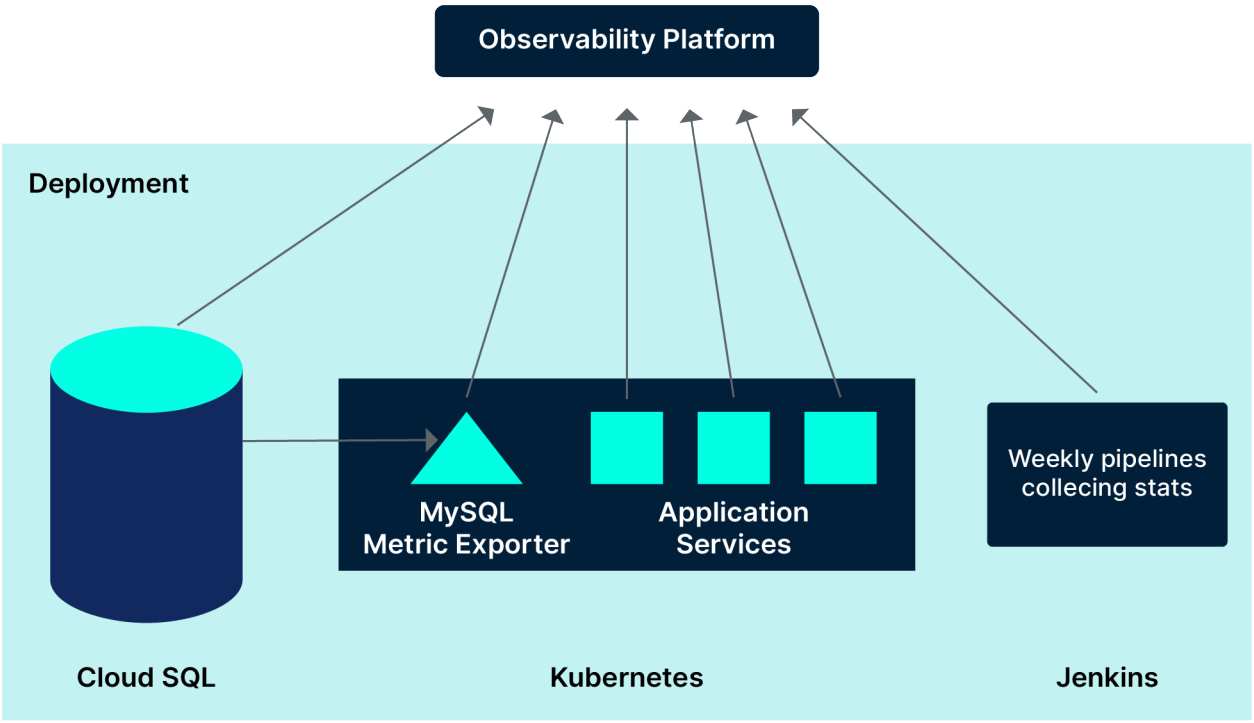
INTRODUCTION

At Rubrik, we rely on a multi-tenant architecture to store customer metadata in a large fleet of Cloud SQL database instances. With numerous production deployments globally, each supporting multiple customer accounts, maintaining high availability, performance, and robustness across this infrastructure is critical. Managing a large fleet of Cloud SQL instances and ensuring they remain resilient and performant has been a journey filled with valuable learnings. In this blog, I'll walk you through the strategies, challenges, and solutions that enable us to manage our database fleet effectively.



MONITORING: THE FOUNDATION OF HIGH AVAILABILITY

METRICS MATTER



The journey to high availability starts with robust monitoring—after all, you cannot improve what you do not observe. At Rubrik, we take a multi-layered approach to monitoring that goes beyond native tools to capture the depth and detail we need for proactive management.

We leverage **GCP's native Cloud SQL metrics**, which offer foundational insights, and extend this with a **custom metric exporter** that runs as a Kubernetes service on Google Kubernetes Engine (GKE). This exporter captures additional insights that aren't natively available in Cloud SQL, filling critical visibility gaps.

To capture granular details at the query level, we also built a **custom wrapper around the SQL driver** used by our services, which emits detailed metrics on every query. This provides us with actionable insights into query performance and helps us identify and address inefficient queries before they become problematic.

For even more observability, we've enabled **MySQL Performance Schema** across our fleet, giving us access to fine-grained details on MySQL internals, including memory usage internals, per-database load, and lock contention. The metric exporter taps into this schema to collect critical information that allows us to monitor and optimize at both a high level and in-depth.

Additionally, we run **weekly pipelines** that aggregate statistics on table size growth across the entire fleet, helping us keep an eye on data bloat, potential indexing issues, and other space-related concerns. These various sources of metrics collectively provide us with a 360-degree view of our production database instances, laying the foundation for the stability, performance, and availability that our customers expect.

AUTOMATIC DEBUG INFORMATION COLLECTION

When an alert fires, every second counts. To minimize response time and ensure effective troubleshooting, we've implemented an **automatic debug information collection pipeline**. This pipeline is triggered whenever an alert is raised, instantly gathering essential debug data from the affected Cloud SQL instance.

This automatic process significantly accelerates **root cause analysis (RCA)** and reduces the **mean time to resolution (MTTR)** to minutes from several hours and even days when waiting for issue recurrence. It allows our team to focus on diagnosing and resolving the issue rather than spending time manually collecting logs and diagnostic data. By having the relevant information upfront, we can jump straight into incident analysis, ultimately improving the reliability and resilience of our database fleet.

Incorporating this kind of automated data collection not only optimizes our response to incidents but also feeds into a continuous improvement loop. With a more complete dataset on hand, we can better analyze incident patterns, proactively refine alerts, and implement preventive measures to reduce recurrence.

ALERTS: PROACTIVE MONITORING

In our pursuit of high availability, **alerting** plays a pivotal role in enabling proactive monitoring and early detection of issues across our database fleet. We've configured a wide range of targeted alerts, each designed to monitor key performance indicators and operational health metrics across the databases. These alerts serve as our early-warning system, empowering our team to address potential problems before they escalate.

Some of the critical alerts we've established include:

Uptime / Availability
Resource Utilization on CPU, Memory, Storage, Connections
InnoDB History List Length (HLL)
InnoDB Row Locking
InnoDB Redo Log Usage Percentage
MySQL Partition Management

In addition to these primary alerts, we have several alerts dedicated solely to **proactive debug information collection**. These special-purpose alerts allow us to gather critical data as soon as anomalies are detected, ensuring we capture a snapshot of the database's state during early-stage incidents. By having these alerts in place, we can streamline future root cause analyses (RCA) and mitigate issues before they develop into critical failures.

By strategically configuring and fine-tuning these alerts, we maintain a proactive stance on database health, helping us catch and resolve brewing issues before they impact performance or availability.

WEEKLY REVIEW OF METRICS

A key part of our monitoring strategy is the **weekly review of top metrics**. Each week, our team gathers to examine a curated metrics dashboard that reflects the health and performance of the entire production fleet. This regular review serves multiple purposes:

1. **Fostering Familiarity:** By consistently engaging with production metrics, our engineers gain an intimate understanding of normal patterns and behavior across our database instances. This familiarity is crucial for quickly identifying deviations during incidents, ultimately reducing response times.

2. **Boosting Confidence:** Regular exposure to real-time data builds our engineers' confidence in handling issues under pressure. When incidents occur, they can rely on their experience and insights from these reviews to make informed decisions faster.

3. **Detecting Trends and Early Warnings:** Weekly reviews often reveal subtle shifts in metrics that may not yet trigger existing alerts. Identifying these trends early allows us to address potential issues proactively, whether by adjusting configurations, tuning queries, or adding new alert conditions to cover gaps in our monitoring strategy.

These weekly sessions are more than just a health check—they're an opportunity to continuously improve our monitoring setup and sharpen our team's incident response capabilities. By regularly refining our approach based on data and observations, we're able to stay one step ahead of potential issues and ensure the ongoing reliability of our production fleet.

CONSISTENCY OF CONFIGURATION: THE BACKBONE OF STABILITY

In a multi-tenant environment where customer needs vary widely, **configuration consistency** becomes essential for maintaining predictable performance across our fleet of CloudSQL instances. To ensure this, our team has established **standardized policies for every critical server configuration**. This approach prevents discrepancies that could lead to unpredictable behavior, costly outages, or performance degradation.

Our guiding principle is **"Look at the forest, not just a tree."** Rather than making isolated fixes, we evaluate each server parameter in the context of the entire fleet. This holistic mindset allows us to fine-tune configurations for the collective performance of hundreds of databases, proactively eliminating potential issues before they impact any one instance.

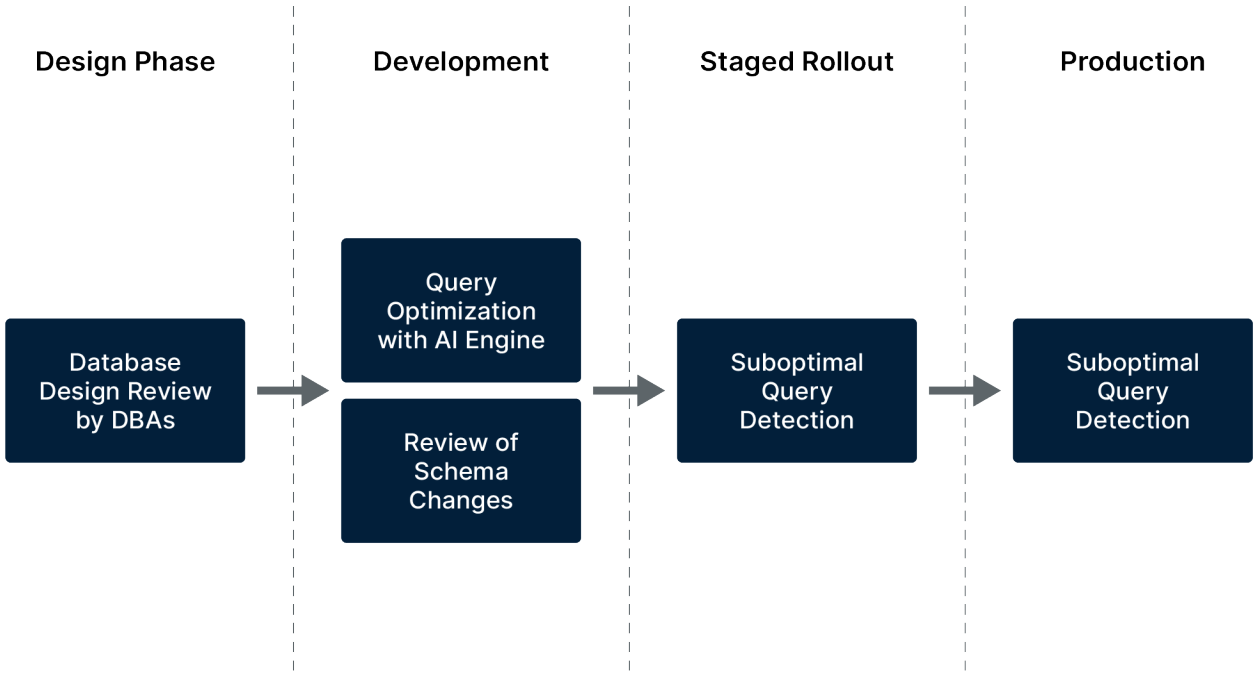
For example, each CloudSQL instance can have numerous tables per customer, which could quickly exhaust available table definition cache. By increasing the table definition cache size across the fleet, we avoid bottlenecks without needing to manage individual instances. Similarly, we size servers based on anticipated load to achieve an optimal balance of cost and performance.

Some of the **key parameters we've standardized include**:

Number of Connections
InnoDB Redo Log File Size (innodb_log_file_size)
InnoDB Purge Variables (innodb_max_purge_lag, innodb_max_purge_lag_delay, innodb_purge_batch_size)
MySQL BinLog Configuration (max_binlog_size)
Temporary Log File Size (innodb_online_alter_log_max_size)
Table Cache Sizes (table_open_cache, table_definition_cache)

By codifying and enforcing these configurations, we achieve uniform performance and reliability across our global deployments, enabling our engineers to manage the fleet efficiently.

EARLY AND PROACTIVE OPTIMIZATION FOR OPTIMAL QUERY PERFORMANCE



BOOSTING DEVELOPER PRODUCTIVITY WITH EARLY OPTIMIZATIONS

SHIFT-LEFT: OPTIMIZING DURING DEVELOPMENT

At Rubrik, we believe in catching potential database issues early in the development process—an approach known as the **shift-left mindset**. This proactive strategy enables our Database Administrators (DBAs) to **review table designs and query structures during the development phase** rather than after deployment, where fixes can be costly and disruptive.

To support this, our **engineering-wide design document template** includes a dedicated section for database design. Here, developers are required to detail aspects such as the data model, expected workload, data growth projections, cleanup criteria, performance requirements, and security considerations. This practice ensures that database architecture is thoughtfully integrated from the outset and aligned with long-term operational needs.

Additionally, we've implemented the DB Delegate program, where volunteers from various feature teams are trained to perform schema change reviews. These delegates act as additional eyes and ears for the DBAs, ensuring that schema changes adhere to best practices and operational requirements before they are implemented. This distributed approach to schema reviews not only strengthens our shift-left strategy but also fosters cross-functional collaboration, empowering feature teams to take a more active role in database performance and scalability.

AI-Driven Query Optimization in Code Reviews

We've integrated automation into our code review process in which **newly added queries are automatically fed into an AI-driven optimization engine** that provides suggestions to improve performance, indexing, and structure. By receiving real-time feedback during code review, developers can refine queries before they reach production, **reducing the risk of suboptimal query patterns** from impacting performance at scale. This shift-left approach empowers our teams to maintain high performance as the system grows and evolves.

Ensuring Smooth Operations Post-Rollout

MANAGING SLOW QUERIES IN PRODUCTION

Proactive query optimization doesn't stop with development. To ensure our databases remain efficient under real-world loads, we **analyze slow queries daily** using automated tools, which systematically classify and prioritize the most impactful slow queries, presenting them to the relevant teams for review and optimization.

This continuous review process enables us to **identify and address potential performance bottlenecks** before they affect customers. By optimizing query execution paths, adjusting indexes, and eliminating inefficiencies, we keep our databases running smoothly, ultimately contributing to a fast and seamless customer experience.

Together, these early and proactive optimization strategies create a resilient foundation for scalable performance, allowing us to meet the growing demands of our multi-tenant environment without compromising on speed or reliability.

DESIGN CHOICES MATTER

OPTIMAL METADATA CLEANUP

Efficient metadata cleanup is essential for maintaining database performance in a multi-tenant environment. At Rubrik, we leverage **MySQL partition tables** to simplify this process, converting traditionally heavy DML operations (like bulk deletes) into streamlined DDL operations. By using commands such as `ALTER TABLE .. PARTITION`, we can **drop partitions instead of deleting rows**, which significantly reduces server load. A single `ALTER TABLE` command can efficiently drop millions of rows, making cleanup operations faster and less resource-intensive.

To further reduce the impact of these operations, we schedule DDL-based cleanups during **off-peak hours**, ensuring that they do not disrupt customer-facing services.

Our team also developed a **partition management framework** that empowers feature teams to easily implement partitioning by simply specifying two parameters: interval (daily, weekly, monthly, or yearly) and retention period (e.g., 7 days, 6 months, or 2 years). This framework simplifies the adoption of partition tables across different features and ensures that large data tables are efficiently managed throughout their lifecycle.

CONNECTION MANAGEMENT AND QUERY CACHING

Managing connections and optimizing query flow are critical in high-traffic, multi-tenant environments. We rely on **ProxySQL** to multiplex connections, which helps reduce the overhead associated with frequent connection opening and closing. ProxySQL effectively **reduces connection churn** on the database server, allowing it to handle higher loads with greater stability.

In addition, ProxySQL's **query caching capabilities** significantly reduce the number of repetitive queries sent to the database. This reduces the workload on CloudSQL instances by caching frequently accessed query results, thereby improving response times and allowing the databases to dedicate resources to more critical or unique queries.

WORKLOAD ISOLATION: REDUCING NOISE FOR OPTIMAL PERFORMANCE

To tackle the "noisy neighbor" problem, where high-intensity workloads impact other operations on the same instance, we strategically **separated different types of workloads into dedicated instances**. Specifically, we moved our **write-heavy job framework workload** onto its own set of instances, isolating it from the **customer metadata workload**. This design choice allowed us to prevent the intensive, often bursty job framework operations from interfering with the performance of metadata-related transactions, which require consistent, low-latency access.

By isolating these workloads, we eliminated resource contention and significantly **improved the stability and predictability** of both environments. This separation also enables more targeted scaling and optimization for each workload type, ensuring that each receives the resources and configurations best suited to its unique requirements.

AUTOMATED SCALING

AUTO-SCALING AND INSTANCE SPLITTING

To maintain optimal performance and manage workload spikes effectively, we have implemented a sophisticated **auto-scaling** mechanism across our CloudSQL fleet. Our monitoring systems continuously observe metrics like CPU, memory usage, and query throughput, allowing us to proactively **adjust instance sizes based on demand**. When workload increases to a sustained, justifiable level, instances are automatically upscaled to handle the additional load, ensuring seamless performance even during traffic surges.

For instances that have grown too large due to increased data or a high volume of tenants, we employ **instance splitting**. This strategy involves redistributing data and services across multiple instances, reducing the burden on any single server. By splitting oversized instances into smaller, manageable units, we can maintain high performance while simplifying resource allocation and troubleshooting.

STAYING AHEAD OF THE UPGRADE CURVE

In the fast-evolving landscape of database technology, staying ahead of the upgrade curve is crucial for maximizing performance and leveraging the latest innovations. In early 2023, we undertook a comprehensive upgrade of our entire CloudSQL fleet from MySQL 5.7 to MySQL 8.0. This strategic move was driven by our commitment to continuous improvement and operational excellence.

Upgrading to MySQL 8.0 unlocked a host of powerful features and performance enhancements that have had a transformative impact on our database operations. Notably, the introduction of **instant column addition** significantly reduces the time required for schema changes.

Additionally, MySQL 8.0 offers **increased throughput**, which translates to faster data processing and improved overall system performance. As we continue to scale our operations, these enhancements enable us to handle higher volumes of transactions seamlessly while maintaining high levels of availability and reliability.

INCIDENT HANDLING: LEARNING FROM CHALLENGES

At the heart of our operational philosophy is a commitment to learning from every database-related incident. Each incident undergoes a comprehensive investigation aimed at identifying root causes and eliminating them whenever possible. Our approach is not just reactive; it's about fostering a culture of continuous improvement and resilience.

In cases where fully eliminating a root cause isn't feasible, we focus on developing **automated mitigation strategies**. These measures act as safeguards, helping to manage incidents more effectively and ensuring that our systems remain robust in the face of challenges. For instance, runaway growth of the **InnoDB history list length (HLL)** can be mitigated by **automatically throttling update rates**. By setting parameters like `innodb_max_purge_lag` and `innodb_max_purge_lag_delay`, we enable purge threads to catch up, preventing HLL from building up excessively.

Furthermore, each incident provides valuable insights that drive enhancements to our monitoring systems. We continuously refine our approach by creating new alerts and introducing additional metrics, ensuring that we have the visibility needed to detect and respond to potential issues before they escalate. This iterative process strengthens our operational framework and helps us stay one step ahead of future challenges.

Ultimately, these incidents contribute to the overall resilience of our systems, enabling us to minimize downtime and maintain high levels of service availability. By treating each challenge as an opportunity for growth, we not only bolster our current operations but also pave the way for a more stable and efficient future.

CLOSING THOUGHTS

At Rubrik, high availability transcends the mere avoidance of downtime; it embodies our commitment to a culture of continuous improvement. Through meticulous monitoring, ongoing optimization, and comprehensive incident handling, we maintain a CloudSQL fleet that is robust, scalable, and well-equipped to meet the demands of our global customer base. By sharing our experiences and insights, we aspire to inspire others to adopt similar practices, empowering them on their journey toward achieving high availability.

Looking ahead, our focus includes further workload separation, implementing controls to restrict per-query and per-database resource consumption, and building developer frameworks to streamline operations such as data cleanup, pagination, backfill, and bulk updates, ensuring efficiency and performance as we scale.

ACKNOWLEDGMENTS

This journey would not have been possible without the relentless efforts of our incredible platform database, SRE team, and DB delegates. A heartfelt thanks to platform database team members Rajorshi, Travis, Gabriel, Rahul, Yashwanth, Sudip, Anmol, Gurneet, Hardik, and SRE team members Prabudas, Suraj, Mihir whose expertise and dedication have been instrumental in scaling Rubrik's database fleet to support our ever-growing ARR. Your commitment to excellence and innovation is what drives us forward. Thank you for being the backbone of our success!

rubrik

Global HQ
3495 Deer Creek Road
Palo Alto, CA 94304
United States

1-844-4RUBRIK
inquiries@rubrik.com
www.rubrik.com

rwp-insights-from-engineering-robustness-in-rubriks-large-cloud-sql-fleet / 20241127