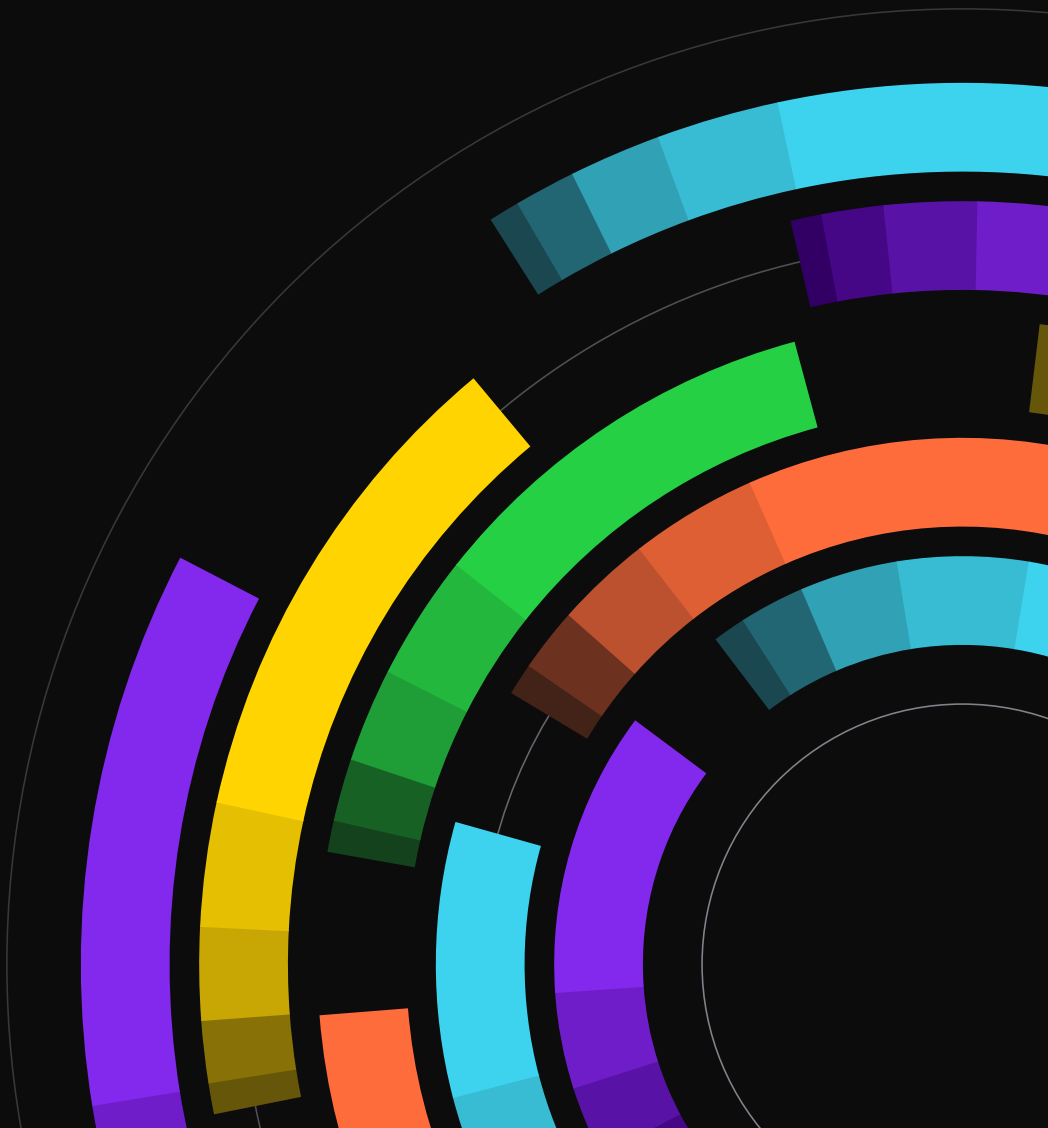
mariaDB

WHITE PAPER

MariaDB Cloud versus Amazon RDS for MySQL and MariaDB



CONTENTS

Table of contents

01	**Executive summary**	4
02	**Architectural foundations: community constraints versus enterprise stability**	5
	The inherent risks of Community Server deployments	5
	Lifecycle management and the technical debt of forced migrations	6
	The severe financial penalty of AWS Extended Support	6
03	**Redefining high availability: from passive standby to active-active continuity**	8
	The vulnerability and latency of DNS-based failover	8
	Transparent high availability via MariaDB MaxScale	8
	Advanced diagnostic tooling: Workload Capture and Replay	10
	Cross-cloud disaster recovery and SLA superiority	10
04	**Performance, scalability, and the serverless paradigm**	12
	Eliminating the single-writer chokepoint	12
	Elastic compute and true serverless architecture	12
	Provisioned compute autoscaling: head-to-head with RDS	13
	True serverless architecture and scale-to-zero	13
	Empirical price-performance benchmarks	14
05	**The unified data stack: HTAP and the AI-ready future**	15
	The architectural tax of AWS service sprawl	15
	Unified HTAP with MariaDB Exa	16
	Agentic AI, vector search, and the MCP server integration	16
06	**Total cost of ownership (TCO) and operational complexity**	18
	The deep inefficiency of the passive standby tax	18
	Elevating operations: Fractional DBA and Collective Intelligence	19
	Deployment flexibility: Bring Your Own Cloud (BYOC)	19
07	**Migration pathways**	21
08	**Conclusion**	22
	Sources and references	23

MariaDB Cloud	vs.	RDS
Active-active nodes	•	Passive standby nodes
Up to 99.995% SLA	•	99.95% SLA
Up to 8-year lifecycle	•	Frequent forced upgrades
True serverless scaling	•	Always-on provisioning
Cross-cloud DR	•	Single-cloud dependency
Integrated capabilities	•	Add-on heavy architecture

01

Executive summary

The enterprise database ecosystem has reached a critical inflection point in 2026. Organizations are grappling with the escalating computational demands of generative artificial intelligence, the necessity for high-throughput hybrid transactional and analytical processing (HTAP), and the unforgiving business requirement for continuous availability with zero downtime. As cloud architects and infrastructure leaders evaluate their data foundations, the traditional managed database model—characterized by legacy "lift and shift" provisioning, passive standby redundancy, and rigid lifecycle management—is increasingly proving inadequate for modern, cloud-native application architectures. A rigorous technical and economic analysis of Amazon Relational Database Service (RDS) against MariaDB Cloud reveals fundamental divergences in how these platforms approach core stability, scalability, continuous availability, and the holistic total cost of ownership (TCO).

Amazon RDS has historically served as a standard, ubiquitous utility for provisioning relational databases within the broader public cloud ecosystem. However, the underlying operational paradigm of AWS RDS for MySQL and MariaDB introduces distinct architectural and financial constraints. Because the service relies strictly on community editions of the database engines, organizations are subjected to aggressive end-of-life (EOL) cycles, reactive tiered support models, and legacy Domain Name System (DNS) failover mechanisms that inherently risk transaction loss and application downtime during infrastructural shifts[1]. Furthermore, the lack of native compute autoscaling and the reliance on disparate, fragmented AWS services for real-time analytics and artificial intelligence impose a heavy "architectural tax" on engineering teams, creating brittle data pipelines that require constant maintenance.

Conversely, MariaDB Cloud represents a purposefully engineered, unified data foundation built upon the battle-hardened MariaDB Enterprise Server[2]. The platform fundamentally alters the Database-as-a-Service (DBaaS) paradigm by dismantling static, intelligent proxy with unparalleled scaling, routing and availability characteristics via MariaDB MaxScale, and true serverless elasticity[3]. With the integration of the MariaDB Enterprise Platform 2026, the service natively unifies high-velocity transactional processing, extreme-scale analytics (facilitated by MariaDB Exa), and agentic artificial intelligence capabilities—including built-in vector search, Model Context Protocol (MCP) servers, and retrieval-augmented generation (RAG) pipelines[2,4].

This comprehensive research report deconstructs the operational, architectural, and financial dimensions of both platforms. By exhaustively examining performance benchmarks, high availability mechanics, artificial intelligence integration, and long-term TCO dynamics, the evidence indicates that migrating from legacy AWS RDS architectures to MariaDB Cloud neutralizes hidden technical debt, eliminates the financial drain of the "idle standby tax," and accelerates the deployment of intelligent, agentic applications.

02

Architectural foundations: community constraints versus enterprise stability

The foundational software running beneath a managed database service dictates the ultimate limits of its security perimeter, its operational stability, and its long-term predictability. The strategic divergence between Amazon RDS and MariaDB Cloud begins at the kernel level of the database engine itself, shaping every subsequent operational capability.

The inherent risks of Community Server deployments

Amazon RDS for MariaDB and MySQL strictly utilizes the Community Editions of these database engines[1]. While open-source community releases serve as excellent platforms for rapid development, localized testing, and non-critical applications, deploying them as the backbone for mission-critical enterprise workloads introduces inherent operational vulnerabilities. Community versions prioritize the rapid introduction of new, experimental features, which inevitably introduces edge-case bugs, unoptimized code paths, and performance regressions. These builds have simply not been subjected to the exhaustive, enterprise-scale stress-testing required for Tier-1 financial, telecommunications, or healthcare systems. Running critical systems on non-hardened versions creates an unnecessary layer of business risk.

MariaDB Cloud, by stark contrast, is built exclusively upon MariaDB Enterprise Server[2]. This enterprise-grade engine undergoes rigorous, battle-hardened quality assurance processes to secure its stability prior to release, focusing on robustness and predictability. Furthermore, the Enterprise Server includes a vast array of critical features that are fundamentally absent from the standard Community Edition. These encompass advanced integration with Key Management Systems (KMS), the HashiCorp Vault plugin for secrets management, and a comprehensive Enterprise Audit plugin[5].

From a performance and tuning perspective, the Enterprise Server includes dynamic resizing of the InnoDB redo log, dynamic alteration of InnoDB purge threads, and an increased index limit (up to 128 indexes per table)[5]. It also features enhanced Oracle compatibility, including a Sybase SQL mode for extended aliases and robust PL/SQL support for stored procedures, enabling organizations to seamlessly lift-and-shift legacy Oracle workloads[6]. By utilizing a hardened engine equipped with these advanced features, organizations insulate themselves against the unpredictability of community-driven software lifecycles while maintaining strict regulatory compliance across global data sovereignty frameworks.

Lifecycle management and the technical debt of forced migrations

Perhaps the most disruptive operational challenge within the AWS RDS ecosystem is its highly aggressive lifecycle management policy. Amazon RDS operates on a compressed support window, typically enforcing an End-of-Life (EOL) cycle of approximately three years for standard support on major versions[7].

This accelerated EOL cycle creates a perpetual "migration treadmill" for enterprise engineering teams. Instead of allocating highly valuable engineering cycles to building revenue-generating application features, development teams are routinely forced to halt operations to conduct extensive regression testing, application refactoring, and database engine upgrades simply to maintain vendor support. If an organization fails to execute these mandatory, labor-intensive upgrades within the tight AWS window, they are automatically enrolled in "Extended Support," which functions as a punitive financial mechanism rather than a value-added enterprise service[8].

Additionally, even though AWS RDS allows deferring major version upgrade up to a point and then will force upgrade the database. More than likely, the apps will then fail, but the burden is on the customer to prevent that[7].

In direct contrast to the AWS operational model, MariaDB Enterprise Server provides an extended Long-Term Support (LTS) lifecycle of up to eight years[9]. For example, MariaDB Enterprise Server 10.6 has had its EOL extended through August 2029 versus on AWS RDS EOL Aug 2026[7,9]. This extended lifecycle provides enterprise architects and decision-makers with a stable, predictable, decade-long infrastructure roadmap. Organizations can schedule major version upgrades based on strategic business requirements and natural application evolution cycles, rather than arbitrary vendor deadlines, entirely eliminating the predatory technical debt associated with forced migrations. Furthermore, critical bug fixes, performance optimizations, and security patches are systematically backported to these older stable enterprise releases, ensuring continuous security without requiring disruptive major version upgrades[2].

The severe financial penalty of AWS Extended Support

The total cost of ownership implications of the AWS Extended Support model are severe and directly impact the bottom line. When an RDS instance running MySQL or MariaDB reaches its community EOL date, AWS applies an automatic surcharge measured per vCPU-hour[8].

This pricing model aggressively escalates over time, penalizing organizations that cannot easily migrate vast, complex database schemas. During the first two years of extended support (Year 1 and Year 2), AWS charges $0.100 per vCPU-hour. However, beginning in Year 3, this fee doubles to $0.200 per vCPU-hour[8]. When calculated across an enterprise fleet hosting dozens or hundreds of databases, these surcharges drastically inflate baseline infrastructure costs without providing any enhancements to performance, scalability, or feature availability.

AWS RDS INSTANCE CLASS	TOTAL VCPUS	BASE COMPUTE MONTHLY COST	YEAR 1 & 2 EXTENDED SUPPORT (MONTHLY SURCHARGE)	% SURCHARGE INCREASE	YEAR 3+ EXTENDED SUPPORT (MONTHLY SURCHARGE)	% SURCHARGE INCREASE
db.t3.medium	2	$0.136/hr ~ $99.28/month	$146.00	147%	$292.00	294%
db.m5.large	2	$0.342/hr ~ $249.66/month	$146.00	58%	$292.00	116%
db.r5.xlarge	4	$0.96/hr ~ $700.80/month	$292.00	41%	$584.00	83%
db.r5.2xlarge	8	$1.368/hr ~ $998.64/month	$584.00	58%	$1,168.00	116%

TABLE 1 Calculated AWS RDS Extended Support fees for deprecated MySQL/MariaDB instances, demonstrating the compounded monthly financial penalty of the 3-year EOL cycle. Calculations assume 730 hours per month[8].

As Table 1 demonstrates, an organization running a moderate db.r5.2xlarge instance will incur an unexpected annual penalty of over $14,000 by Year 3 of extended support, entirely independent of the baseline compute and storage costs. MariaDB Cloud's 8-year support lifecycle fundamentally eradicates this financial risk, offering TCO predictability that is impossible to achieve under the AWS paradigm[9].

03

Redefining high availability: from passive standby to active-active continuity

High availability (HA) and disaster recovery (DR) architectures have evolved significantly to meet the demands of global, continuous commerce, yet many cloud providers remain tethered to legacy failover mechanisms. The architectural differences between Amazon RDS and MariaDB Cloud dictate exactly how an application behaves during infrastructure failures, network failure, and hardware degradation.

The vulnerability and latency of DNS-based failover

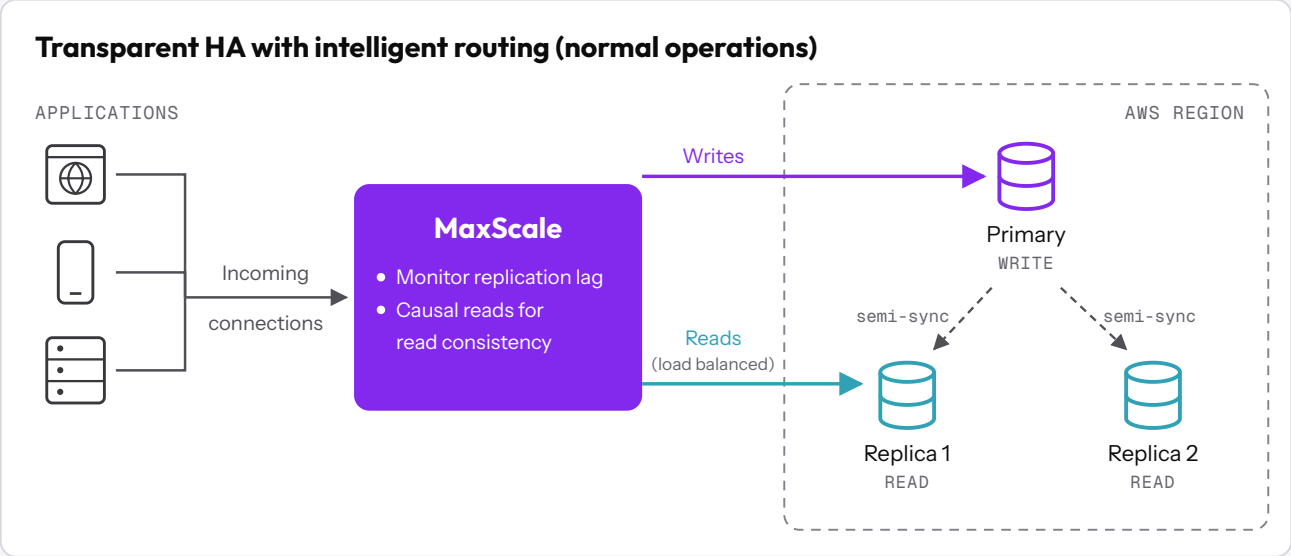
Amazon RDS achieves its standard high availability through a Multi-AZ (Availability Zone) deployment model, relying on a primary database instance synchronously replicating to a passive, "hot standby" instance in a secondary physical AZ[10]. While this fulfills basic infrastructural redundancy requirements, the failover mechanism itself is inherently flawed for modern, low-latency applications that require continuous state preservation.

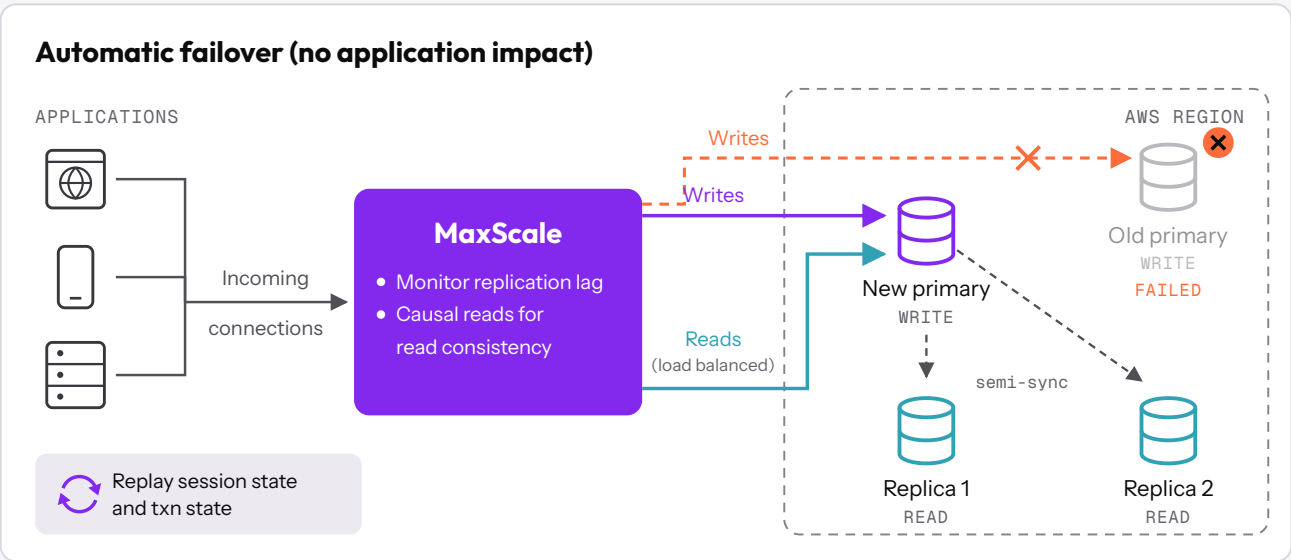
When a primary node fails in the RDS environment, the system relies on DNS (Domain Name System) propagation to redirect traffic to the newly promoted standby node[10]. Because DNS caching is notoriously inconsistent across different operating systems, application frameworks, and network layers, applications frequently attempt to communicate with the dead primary node long after the failover has been triggered by the control plane. Furthermore, because the standby node is entirely passive, it must undergo a time-consuming database crash recovery process before it can begin accepting read and write traffic.

This combination of slow DNS rewiring and mandatory crash recovery typically results in a hard application outage lasting between 60 to 120 seconds[10]. More critically, all in-flight transactions, prepared statements, and session states are abruptly dropped. The application logic must be heavily engineered with complex retry loops and error-handling routines to survive this shift, or the end-user will simply experience a system error and a dropped connection. For write-heavy applications that are frequently writing to the database, this will mean data loss.

Transparent high availability via MariaDB MaxScale

MariaDB Cloud abandons the legacy DNS failover model entirely. Instead, it utilizes an advanced, intelligent database proxy, router, and load balancer known as MariaDB MaxScale, which sits transparently between the application layer and the underlying database cluster[3].





MaxScale enables true "Transparent HA" natively within our primary deployment topology: Primary-Replica. When a primary node failure is detected, MaxScale automatically executes the failover and promotes a healthy replica instantaneously in sub-seconds[11]. Because MaxScale continuously monitors node health and completely abstracts the database infrastructure from the application layer, this topology shift requires zero manual intervention or application-side configuration changes.

For workloads requiring an even higher tier of fault tolerance, the underlying MariaDB Enterprise Cluster can utilize fully synchronous replication powered by Galera, taking high availability a step further[12]. In this configuration, all replicas are constantly active and mathematically synchronized. Because data is written to all nodes simultaneously, there is no passive standby node, entirely eliminating replication lag and the need for a database crash recovery process during a failover event.

Crucially, regardless of your deployment topology, MaxScale makes the entire infrastructural shift transparent to the application through highly advanced connection handling mechanisms[11]:

- **Connection migration:** The proxy holds the TCP connection open with the client application while seamlessly rewiring the backend connection to a newly promoted primary node.
- **Session restore:** MaxScale automatically restores the specific session context—such as user-defined variables, character sets, and specific database context—onto the new primary.
- **Transaction replay:** If a failure occurs precisely in the middle of an active transaction, MaxScale possesses the deep SQL parsing intelligence to automatically replay the interrupted transaction on the new primary without returning an error to the client.

The result is that the application—and by extension, the end user—remains completely unaware that a catastrophic database failure has occurred. The application perceives nothing more than a momentary spike in query latency, rather than a hard disconnect or a lost transaction. While AWS offers RDS Proxy as a separate, billable add-on service to mitigate some connection dropping, it remains a cumbersome integration that lacks the deep transaction replay intelligence and seamless native ecosystem integration of MaxScale[13].

Advanced diagnostic tooling: Workload Capture and Replay

Beyond failover mechanics, MaxScale introduces powerful enterprise diagnostic capabilities unavailable in the standard RDS ecosystem. The Workload Capture and Replay (WCAR) filter allows database administrators to capture live client traffic from a production MaxScale instance and store it in a replayable format[14].

This allows organizations to generate highly accurate, reproducible client traffic without engineering custom application-specific load generators. Administrators can use these captured workloads to repeatedly measure the effects of configuration changes, test query performance against new database engine versions, and debug complex SQL anomalies in an exact mirror of production reality. However, as a fully managed component of MariaDB Cloud, these configurations are autonomously optimized by the provider[2].

Cross-cloud disaster recovery and SLA superiority

The geographic constraints of database availability also present a significant point of divergence. Amazon RDS high availability is generally strictly confined to the borders of a single cloud provider, locking the disaster recovery topology within AWS Availability Zones[10]. Achieving true cross-cloud resilience requires extensive, custom-engineered, manual replication pipelines that are highly fragile and difficult to maintain.

MariaDB Cloud natively supports cross-cloud and cross-region disaster recovery[2]. Organizations can maintain active replicas across entirely different cloud providers (e.g., AWS, Microsoft Azure, Google Cloud) or hybrid on-premises environments. If an entire hyperscaler region goes dark—an event that has occurred multiple times in recent history—MariaDB Cloud can seamlessly redirect traffic to an alternate provider, ensuring true data sovereignty and absolute business continuity[2]. For instance, during the 2026 Middle East conflict, physical drone strikes on AWS data centers in the UAE and Bahrain knocked out regional cloud infrastructure and triggered a cascading global outage that even took down major AI platforms[15].

This architectural superiority is mathematically reflected in the Service Level Agreements (SLAs) offered by both platforms.

METRIC	AWS RDS MULTI-AZ	MARIADB CLOUD (POWER TIER)
Target availability SLA	99.95%[16]	99.995%[17]
Max permitted monthly downtime	21 minutes, 44 seconds	2 minutes, 10 seconds
Core failover mechanism	DNS rewiring & database crash recovery[10]	MaxScale transparent proxy[11]
Transaction/session replay	No (application drops connection)[13]	Yes (automatic replay)[11]
Cross-cloud DR with native support	No (locked to AWS infrastructure)	Yes (AWS, GCP, Azure)[2]

TABLE 2 High availability architecture and SLA comparison, demonstrating the mathematical difference between AWS RDS and MariaDB Cloud.

By shifting from 99.95% to 99.995% availability, MariaDB Cloud reduces allowable monthly downtime from nearly 22 minutes to just over 2 minutes, fulfilling the strict "always-on" requirements of modern digital enterprises while preventing the catastrophic application failures inherent in the RDS DNS failover model[16,17].

04

Performance, scalability, and the serverless paradigm

As application workloads scale globally, the underlying database must flex efficiently to handle both high-concurrency read traffic and sudden, unpredictable spikes in computational demand. Legacy relational architectures force a rigid provisioning model that ultimately strangles performance.

Eliminating the single-writer chokepoint

Amazon RDS relies fundamentally on a traditional single-master architecture[10]. In this model, all write operations (INSERT, UPDATE, DELETE) are bottlenecked through a single primary node. While RDS allows for the creation of read replicas to offload read traffic, these replicas rely heavily on asynchronous logical replication. Under heavy transactional load, asynchronous replication inevitably introduces replica lag, meaning applications querying the read replicas will frequently receive stale, highly inconsistent data.

MariaDB Cloud dismantles this structural bottleneck through its multi-node, active-active clustering environment[12]. The clustering model utilizes fully synchronous replication, guaranteeing absolute zero data loss and flawless data consistency across the entire cluster. Because the cluster operates in a true active-active configuration, each node natively processes read and write operations, entirely bypassing the single-writer limitation.

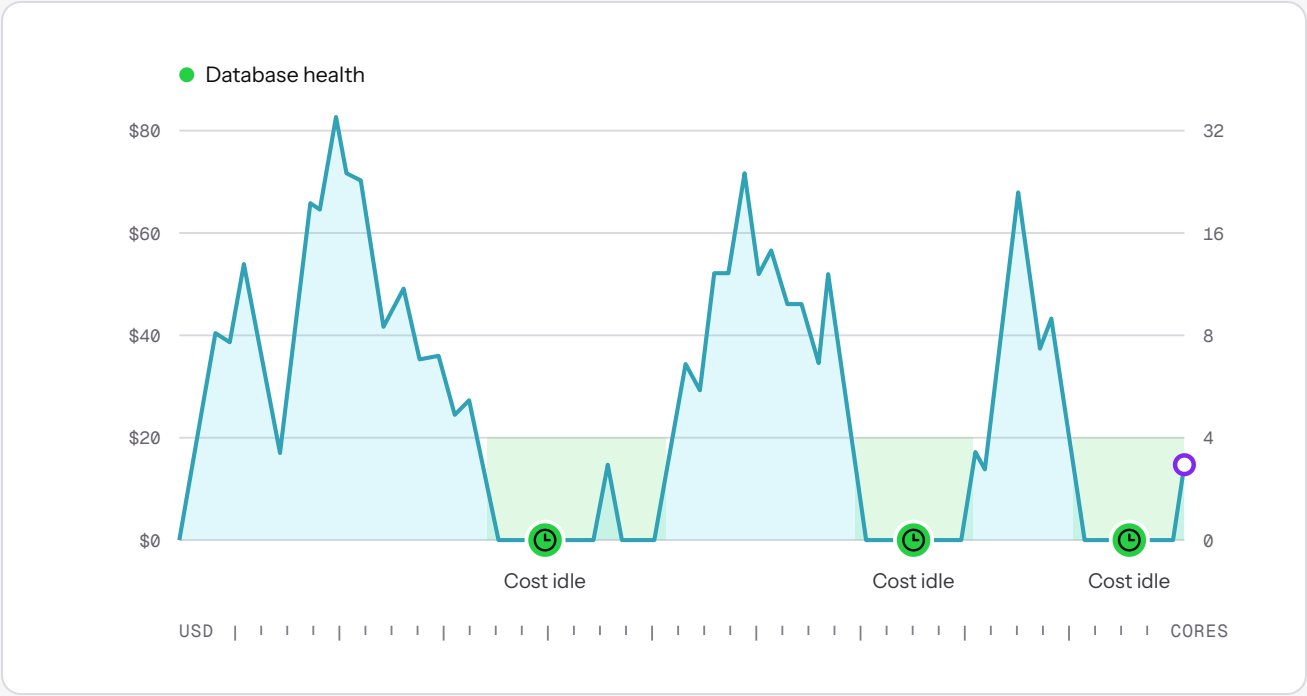
Furthermore, MariaDB MaxScale operates as an intelligent Read/Write Split Router[18]. It automatically dissects incoming database traffic, routing resource-heavy analytical read queries to specific replicas while directing write queries appropriately, ensuring high-concurrency reads never starve the primary transactional nodes of critical compute resources.

Elastic compute and true serverless architecture

In AWS RDS, resource scaling is fundamentally asymmetrical. While RDS provides mechanisms for automatic storage scaling, the critical components of compute (CPU and RAM) remain static, manual resources. If an application experiences a sudden, viral traffic spike, the static compute ceiling of the RDS instance will be quickly exhausted, leading to CPU saturation, dropped connections, and eventual database crashes. This lack of instantaneous compute autoscaling forces database administrators into a paradigm of constant over-provisioning—paying exorbitant hourly rates for peak compute capacity 24/7 as an "insurance policy" against unpredictable traffic spikes.

Provisioned compute autoscaling: head-to-head with RDS

MariaDB Cloud resolves this inefficiency for provisioned databases by offering dynamic horizontal and vertical compute autoscaling[2]. Rather than locking workloads into a rigid instance size like AWS RDS, MariaDB Cloud intelligently monitors application pulse and scales compute resources in sub-seconds to accommodate workload surges automatically. Crucially, this scaling occurs seamlessly on provisioned instances without requiring disruptive reboots or dropping client connections. The platform actively preserves and hydrates the buffer cache during vertical scaling transitions, ensuring query performance remains consistently fast and avoiding the severe latency spikes typically caused by cold cache states.



True serverless architecture and scale-to-zero

Beyond provisioned infrastructure, MariaDB Cloud delivers a true serverless database architecture built for unpredictable, variable, or intermittent workloads[2]. For development, staging, or testing environments that experience long periods of inactivity, the serverless offering can dynamically scale to zero.

The lifecycle operates through a highly responsive, automated loop:

1. **Sub-Second Activation:** When an application sends a query to a database currently scaled to zero, MariaDB Cloud provisions the compute infrastructure in seconds, instantly scaling from 0 to 1.
2. **Elastic Elasticity:** As traffic floods in, the architecture seamlessly scales higher (adding more compute slices or instances) to handle the concurrent load without manual intervention.
3. **Automated Scale-Down:** Once the workload subsides and a period of low or zero traffic is detected, the platform automatically scales the infrastructure back down to 1, and eventually back to 0.

As a result, organizations only pay for the exact compute resources consumed during active query execution, completely eliminating the massive financial waste of keeping idle database instances running 24/7.

Empirical price-performance benchmarks

Rigorous, independent benchmarking validates the architectural advantages of MariaDB over traditional MySQL architectures deployed on AWS.

In comprehensive tests utilizing the TPC-C benchmark (simulating highly complex order-entry OLTP workloads) via Carnegie Mellon Database Group's BenchBase tool, MariaDB Cloud demonstrated overwhelming superiority against AWS Aurora (MySQL)[19]. The benchmark analyzed both platforms deployed in the AWS us-east-1 region within the exact same Availability Zone to eliminate network jitter anomalies.

The results indicated that MariaDB consistently delivered up to 3x higher throughput and 2.5x lower latency across all workload scales compared to AWS Aurora. This massive performance advantage is heavily attributed to the deep engine optimizations inherent in MariaDB 11.4 compared to Aurora's utilization of the older MySQL 8.0.x engine[20].

When calculating the Cost per Million Transactions (CPMT)—a metric that measures how efficiently a platform processes workloads relative to its infrastructure costs (calculated as Cost per Second / Throughput per Second x 1,000,000)—MariaDB Cloud delivered an astounding 7.5x better price-performance than AWS Aurora[19].

BENCHMARK PERFORMANCE METRIC	MARIADB ADVANTAGE OVER AWS	PRIMARY DRIVING ARCHITECTURAL FACTORS
Throughput (transactions)	Up to 3x higher	Optimized storage engine (MariaDB 11.4 vs MySQL 8.0)[20]
Query latency	Up to 2.5x lower	Active-active clustering, transparent MaxScale routing[11,12]
Cost efficiency (CPMT)	Up to 7.5x better	Predictable I/O costs, avoidance of Aurora's separate IOPS billing[19]

TABLE 3 Summary of independent TPC-C benchmark results comparing MariaDB Cloud to AWS Aurora, demonstrating vast differences in raw performance and economic efficiency[19].

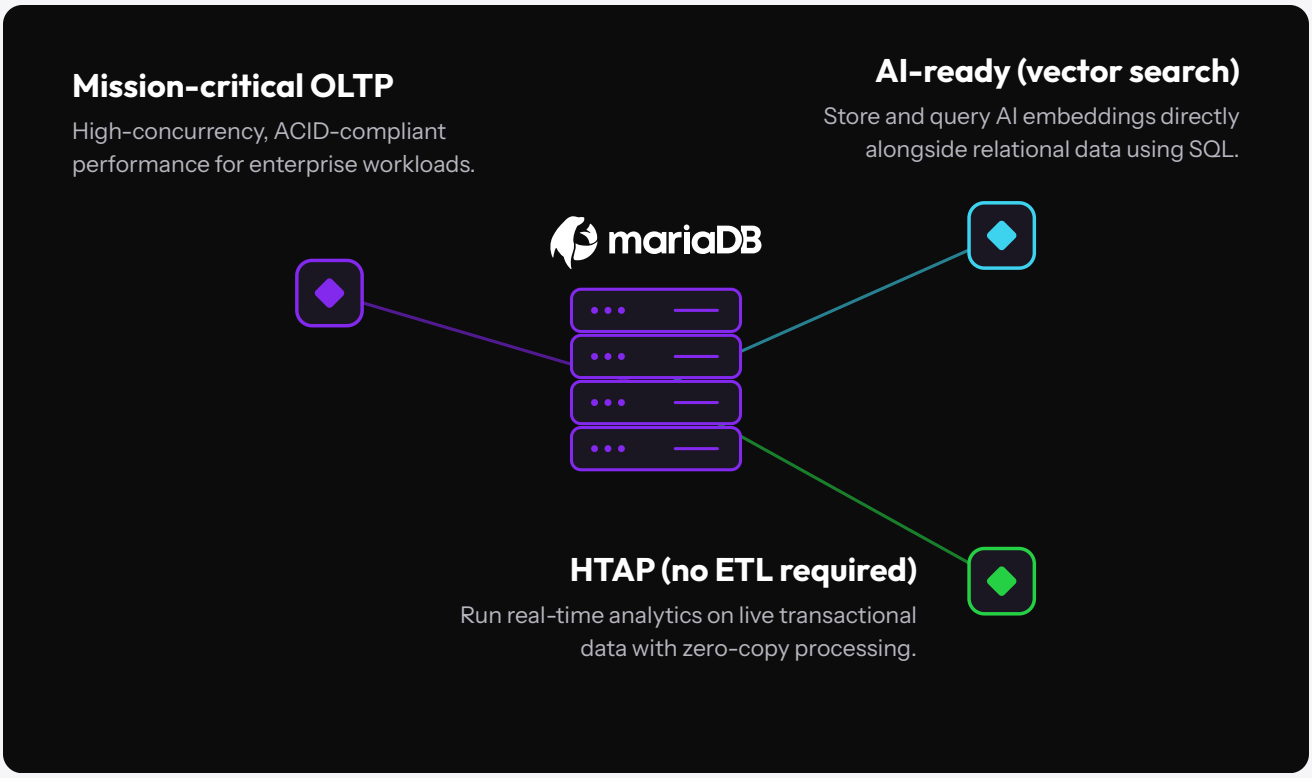
The benchmark revealed that AWS Aurora configurations were found to be 2 to 5 times more expensive simply in baseline configuration[19]. Furthermore, Aurora charges separately for I/O operations (IOPS), which escalate rapidly during high-demand workloads, whereas MariaDB Cloud's pricing is highly predictable and does not arbitrarily penalize high I/O volume[2].

Further independent benchmarks conducted by Mark Callaghan[21] and SQLpipe[22] confirmed these findings, revealing that MariaDB processed 38% more orders per minute than MySQL and was 61% more cost-effective than standard RDS MySQL. From a long-term stability perspective, over 14 releases spanning a decade, MariaDB demonstrated stable performance with near-zero CPU regression, whereas MySQL exhibited a severe performance degradation of nearly 30% between versions 5.6 and 8.0[21].

05

The unified data stack: HTAP and the AI-ready future

The data demands of 2026 extend far beyond high-speed transactional processing. Modern applications require real-time analytics and deeply integrated, agentic artificial intelligence. The traditional approach to these requirements has been to build complex, multi-system data pipelines, a methodology that is fundamentally challenged by unified database architectures.



The architectural tax of AWS service sprawl

Building a comprehensive, modern data stack within the AWS ecosystem requires developers to act less as application builders and more as systems integrators. To achieve a blend of Online Transaction Processing (OLTP), Online Analytical Processing (OLAP), and artificial intelligence capabilities, an organization utilizing RDS must manually stitch together highly fragmented services.

A typical AWS architecture requires RDS for transactions, AWS Redshift for analytics, and Amazon Bedrock or OpenSearch for vector embeddings and AI workloads. Even with AWS's heavily marketed "Zero-ETL" integrations, the reality is that the data must physically move across the network between disparate systems, incurring severe network latency. Consequently, the analytics generated are never truly real time. Furthermore,

each isolated product silo maintains its own query dialect, distinct security governance policies, and complex connection management overhead. This "architectural tax" significantly increases the administrative burden on database engineers, inflates cloud spend, and creates highly fragile infrastructure architectures that break during scale.

Unified HTAP with MariaDB Exa

MariaDB Cloud eliminates the friction of service sprawl by consolidating the entire data lifecycle into a single, cohesive enterprise platform[2]. The platform facilitates true Hybrid Transactional and Analytical Processing (HTAP) without the need for fragile ETL (extract, transform, and load) pipelines or manual data movement.

This is achieved through the native integration of purpose-built columnar storage engines[23]. MariaDB ColumnStore includes an advanced query accelerator that enables high-performance, parallel OLAP queries directly on live operational data. More significantly, the introduction of MariaDB Exa—built in strategic partnership with Exasol—provides an embedded, high-performance in-memory analytics engine[24]. MariaDB Exa utilizes massively parallel processing (MPP) architecture designed for multi-terabyte, complex analytics, delivering query speeds up to 1,000 times faster than traditional InnoDB engines.

Because these engines exist within the exact same unified database perimeter, the application simply issues standard SQL to a single endpoint. MariaDB MaxScale utilizes smart routing to automatically dissect the query, seamlessly sending transactional workloads to InnoDB and massive analytical workloads to Exa or ColumnStore, and subsequently aggregating the results[18]. Developers receive instantaneous, real-time insights on live production data without managing the complexities of data movement.

Agentic AI, vector search, and the MCP server integration

The rapid rise of generative AI demands databases that can act as the authoritative, secure knowledge base for large language models (LLMs). Building AI applications on RDS is a heavy-lifting exercise, requiring data to be constantly extracted to external vector databases (like OpenSearch), which introduces severe risks of data proliferation and noncompliance.

MariaDB Enterprise Platform 2026 is expressly architected for the era of agentic AI[2,4]. The platform embeds vector search capabilities natively into the core database engine, allowing organizations to execute semantic searches alongside traditional relational data. It utilizes hybrid search, merging the keyword precision of full-text search with the semantic relevance of vector embedded search, employing Reciprocal Rank Fusion (RRF) to intelligently rank and unify the results for superior query accuracy.

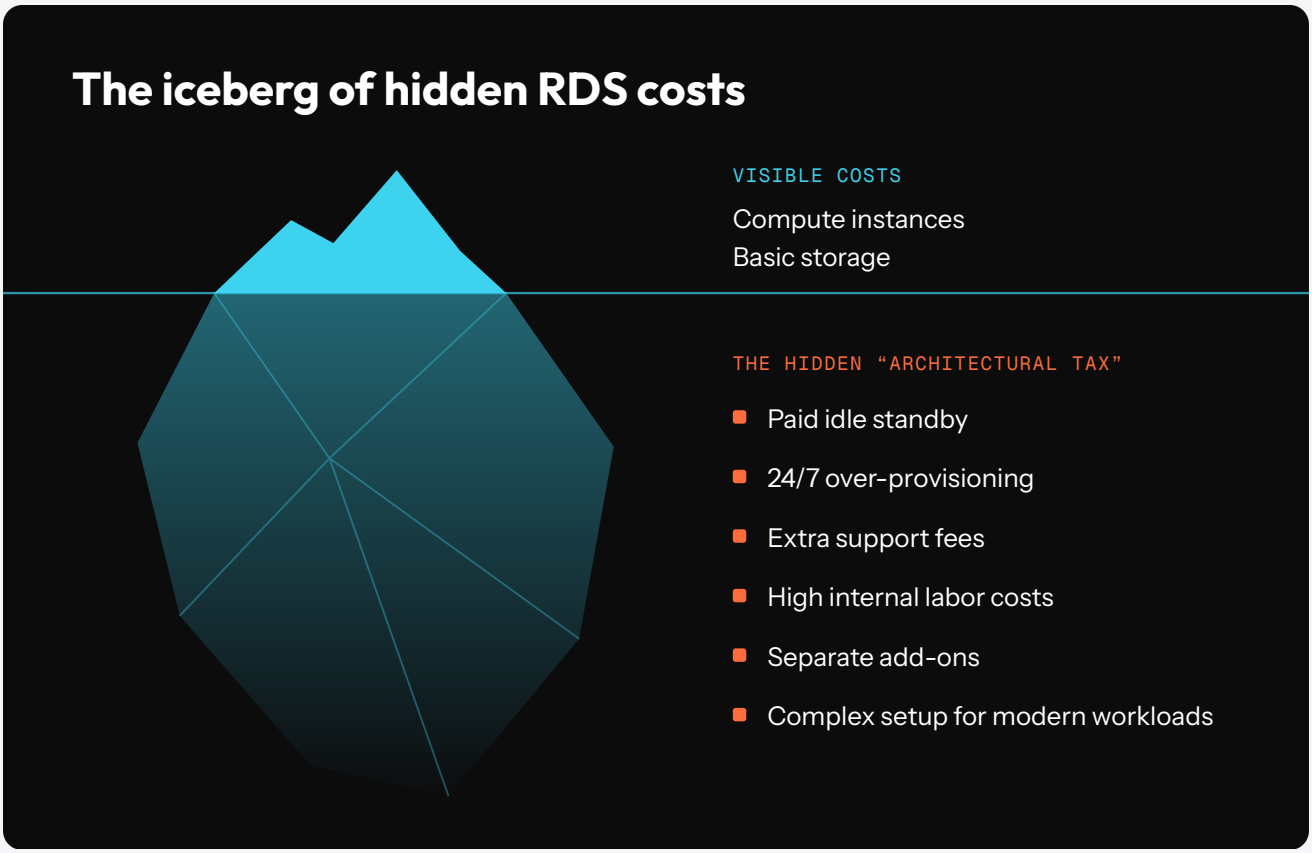
More impressively, MariaDB Cloud provides a native "RAG-in-a-box" pipeline[2]. Retrieval-augmented generation (RAG) traditionally requires highly complex orchestration to handle data chunking, embedding generation, and contextual retrieval. MariaDB AI RAG consolidates this entire workflow into a single, easy-to-use REST API. Deployed as a multi-container microservice stack alongside the database (utilizing components like FastAPI, Redis for message brokering, and Celery workers for processing), it automates document ingestion, smart content splitting, and high-quality response generation. Developers can securely ground their LLMs using their proprietary enterprise data stored directly in MariaDB, eliminating AI "hallucinations" without ever exposing sensitive corporate data outside the secure database boundary.

Furthermore, the platform features fully integrated Model Context Protocol (MCP) servers[2,25]. The MCP server acts as a standardized, secure bridge, allowing external AI agents (such as IDE copilots like Cursor, Windsurf, or autonomous bots) to natively interact with the MariaDB database[25]. The architecture provides core tools for standard SQL execution and vector operations, alongside conditional tools that expose the entire AI RAG pipeline for ingestion and generation. Coupled with built-in AI copilots that offer automated Text-to-SQL query generation and sophisticated database diagnostics, MariaDB Cloud transitions the database from a passive storage bin into an active, intelligent participant in AI-driven development.

06

Total cost of ownership (TCO) and operational complexity

Evaluating cloud database platforms strictly on the sticker price of compute instances ignores the vast operational costs associated with administrative toil, resource utilization inefficiencies, and the necessity of highly specialized internal labor. A holistic TCO analysis demonstrates massive capital efficiencies when migrating away from legacy RDS structures.



The deep inefficiency of the passive standby tax

The Amazon RDS Multi-AZ architecture fundamentally requires organizations to pay for hardware they are architecturally prevented from utilizing[10]. To maintain high availability, users must provision and pay full price for a secondary "hot standby" instance that exactly matches the primary instance in size, power, and cost. However, because of AWS's single-master design, this secondary instance is completely blocked from serving application traffic, providing absolutely zero performance benefit until a catastrophic failure occurs. This "passive standby tax" forces organizations to literally pay for "dead air" 24 hours a day.

MariaDB Cloud operates on a paradigm of 100% resource utilization[2]. Within a standard Primary-Replica MariaDB Cluster, the replica nodes do not sit idle. Instead, every provisioned node is active and participating in serving query requests by handling read workloads offloaded automatically by MaxScale[18]. Consequently, the hardware provisioned for high availability and redundancy is also actively contributing to overall cluster throughput and daily performance, rather than acting as an expensive, locked-away insurance policy.

Combined with the cost-cutting serverless scaling capabilities that eliminate the need for peak over-provisioning, organizations report realizing up to 65% total cost savings compared to equivalent deployments on Amazon RDS[2].

Elevating operations: Fractional DBA and Collective Intelligence

Managing an RDS environment remains an "admin-heavy" endeavor. Despite being marketed as a managed service, the persistent quirks of community database engines, the necessity of manual compute scaling during traffic spikes, and the fragmented nature of the AWS analytics ecosystem force companies to hire dedicated, highly paid internal database administrators (DBAs) merely to maintain basic infrastructure hygiene. When complex, engine-level bugs inevitably occur, RDS provides only tiered, generalist infrastructure support. These generalists can assist with console configuration, but they are technically powerless to patch deep-level kernel bugs, forcing clients to wait indefinitely for upstream community fixes.

MariaDB radically alters this operational support dynamic by providing direct access to the software engineers who actually architect and build the database kernel[26]. This guarantees that severe, core-level database anomalies are resolved rapidly with direct code-level fixes rather than temporary operational workarounds.

To augment internal teams, MariaDB provides highly proactive Managed Operations services[26]. This "Fractional DBA" model grants organizations access to a pool of global database experts for a fraction of the cost of a full-time internal hire. Unlike reactive AWS support, MariaDB's Managed Operations experts utilize "Collective Intelligence"—leveraging telemetry, diagnostics, security profiles, and optimization data synthesized from hundreds of global, mission-critical deployments—to proactively identify performance bottlenecks, execute deep health checks, and prevent outages long before they impact the business. This allows internal engineering teams to escape the RDS migration treadmill, stop babysitting fragile infrastructure, and reallocate their headcount toward building core product features.

Deployment flexibility: Bring Your Own Cloud (BYOC)

For enterprises operating under strict regulatory frameworks, non-negotiable data residency requirements, or complex financial commitments with hyperscalers, standard SaaS database offerings often present insurmountable compliance hurdles. MariaDB Cloud circumvents this barrier through its innovative "Bring Your Own Cloud" (BYOC) architecture[2].

BYOC allows organizations to deploy fully managed MariaDB databases directly within their own cloud tenant and Virtual Private Cloud (VPC). The data plane—encompassing all compute instances, storage volumes, networking routes, and backup artifacts—remains securely locked within the customer's own security perimeter, ensuring absolute data sovereignty and auditability. Concurrently, the MariaDB Cloud control plane manages the database lifecycle remotely, executing provisioning, automated failovers, and necessary patching via secure,

short-lived identities without ever accessing the underlying customer data. This model provides the ultimate synthesis for enterprise architecture: the frictionless operational ease of a fully managed DBaaS combined with the strict infrastructure control and financial benefits of utilizing existing cloud volume discounts, committed spend agreements, and enterprise security policies.

07

Migration pathways

The strategic imperative to migrate from AWS RDS to MariaDB Cloud is clear, yet executing massive database migrations carries inherent perceived risks regarding downtime and data fidelity.

Traditional migration utilities within the AWS ecosystem, such as the AWS Database Migration Service (DMS), are often positioned as seamless solutions but possess critical technical limitations[27]. AWS DMS is highly proficient at transferring raw table data (homogeneous migrations), but it is notoriously deficient at migrating secondary database objects. Triggers, complex stored procedures, intricate indexing structures, specific user privileges, and subtle schema variations are frequently lost or require highly complex secondary manual conversions using the AWS Schema Conversion Tool (SCT)[27].

To streamline this transition and guarantee fidelity, MariaDB provides purpose-built utilities, including the MariaDB Migration Assessment Tool[28]. This tool analyzes the source RDS or Oracle instance—evaluating PL/SQL packages, data types, identifiers, sequence behaviors, and custom SQL functions—to automatically recommend necessary architectural adaptations, drastically reducing the friction of heterogeneous or cross-platform migrations prior to the actual data movement.

08

Conclusion

The cloud database ecosystem is undergoing a fundamental and irreversible realignment in 2026. The baseline expectation for infrastructure has shifted from simple, static cloud-hosted storage to intelligent, proactive, and continuously available data platforms capable of powering next-generation applications.

Amazon RDS, while historically foundational to early cloud adoption, binds modern organizations to a legacy paradigm. Its reliance on community-tier engines, punitive forced-upgrade cycles, the crushing financial weight of Extended Support fees, asynchronous passive-standby architectures, and a highly fragmented analytical and AI toolset introduces severe technical debt and unmanageable operational overhead.

MariaDB Cloud represents the modern antithesis to these constraints. By delivering an actively managed, 8-year LTS Enterprise Server fortified with resilient clustering, transparent MaxScale proxy failovers, and true serverless scale-to-zero compute elasticity, it neutralizes the most pervasive risks of cloud database management while delivering a significantly lower overall Total Cost of Ownership (TCO) compared to AWS RDS. With the strategic integration of seamless HTAP via MariaDB Exa and embedded agentic AI pipelines—including native vector search, RAG-in-a-box APIs, and secure MCP server bridging—MariaDB Cloud does not merely store data; it actively powers the reasoning and execution of the next generation of real-time, intelligent applications. For enterprises prioritizing absolute availability, substantial TCO optimization, and aggressive AI innovation without the architectural tax of service sprawl, the strategic mandate is definitively shifting away from RDS and toward the unified, highly performant capabilities of MariaDB Cloud.

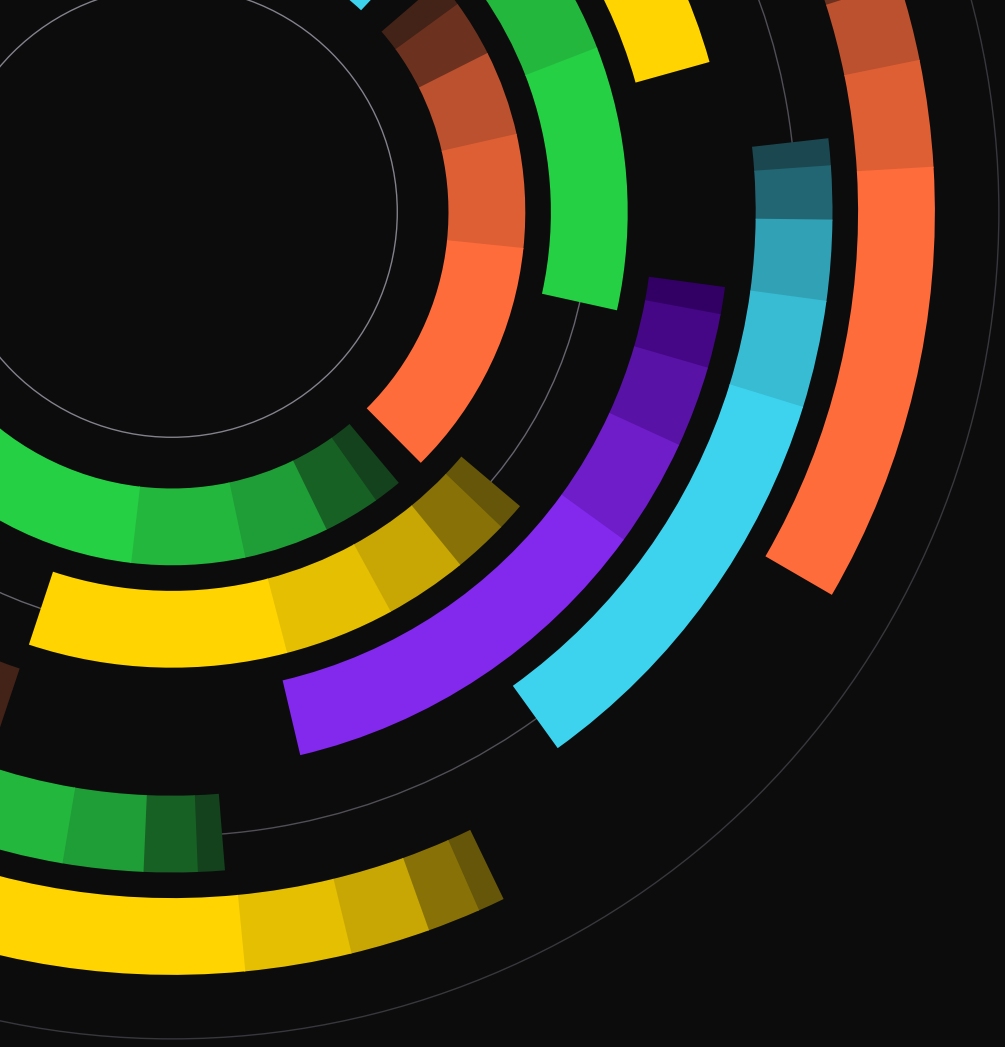
Get started with MariaDB Cloud for free

mariadb.com/cloud-get-started

SOURCES

Sources and references

1. Amazon RDS Features: Choice of Database Engines - https://aws.amazon.com/rds/features/
2. MariaDB Cloud Overview - https://mariadb.com/docs/mariadb-cloud/
3. MariaDB MaxScale Architectural Components & Deployment Guide - https://mariadb.com/docs/maxscale/
4. Unifying AI Agents and Enterprise Data: MariaDB Enterprise Platform 2026 Release Log - https://mariadb.com/resources/blog/announcing-the-release-of-mariadb-enterprise-platform-2026/
5. MariaDB Enterprise Server vs. MariaDB Community Server: Detailed Feature Comparison - https://mariadb.com/products/community-server-vs-enterprise-platform/
6. SQL Mode: ORACLE and PL/SQL Compatibility Reference - https://mariadb.com/docs/release-notes/community-server/about/compatibility-and-differences/sql_modeoracle
7. Amazon RDS Engine Versions and Release Life Cycle Policies - https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/extended-support.html
8. Amazon RDS Extended Support Pricing: Cost per vCPU-Hour Adjustments - https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/extended-support-charges.html
9. MariaDB Maintenance and Long-Term Support (LTS) Release Lifecycle Policy - https://mariadb.org/about/#mariadb-server-long-term-release-maintenance-periods
10. High Availability for Amazon RDS: Multi-AZ Deployments using Synchronous Mirroring - https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Concepts.MultiAZ.html
11. Configuring Transparent Failover and Session Replay Mechanics - https://mariadb.com/docs/maxscale/maxscale-use-cases/maxscale-overview
12. Multi-Master Replication for MariaDB Enterprise Cluster - https://mariadb.com/docs/galera-cluster
13. Managing Connections with Amazon RDS Proxy - https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/rds-proxy.html
14. The MaxScale Workload Capture and Replay (WCAR) Filter Diagnostic Manual - https://mariadb.com/docs/maxscale/reference/maxscale-filters/maxscale-workload-capture-and-replay
15. Amazon's Cloud Unit Says Drone Strikes Damaged UAE and Bahrain Facilities - https://www.reuters.com/world/middle-east/amazon-cloud-unit-flags-issues-bahrain-uae-data-centers-amid-iran-strikes-2026-03-02/
16. Amazon RDS Service Level Agreement (SLA) - https://aws.amazon.com/rds/sla/
17. MariaDB Cloud Power Tier Availability Service Level Agreement - https://mariadb.com/terms/
18. Read/Write Split Router (readwritesplit) Configuration in MaxScale - https://mariadb.com/docs/maxscale/reference/maxscale-routers/maxscale-readwritesplit
19. BenchBase: A Multi-DBMS SQL Benchmarking Framework via JDBC - https://github.com/cmu-db/benchbase
20. MariaDB 11.4 Storage Engine Optimizations and Query Optimizer Enhancements - https://mariadb.com/docs/release-notes/community-server/11.4/11.4.0
21. Sysbench on a medium server: MariaDB is faster than MySQL - http://smalldatum.blogspot.com/2024/07/sysbench-on-medium-server-mariadb-is.html
22. MariaDB Enterprise Features & Security - https://mariadb.com/docs/server/
23. MariaDB ColumnStore Columnar Architecture for Real-Time Analytical Workloads - https://mariadb.com/docs/analytics/#mariadb-columnstore
24. Exasol and MariaDB Announce Strategic Partnership - https://www.exasol.com/press/exasol-and-mariadb-announce-strategic-partnership/
25. Model Context Protocol (MCP) Specification: Standardizing Database Connections for Agentic AI Environments - https://modelcontextprotocol.org/docs/learn/architecture
26. MariaDB Remote DBA (RDBA) Services and Proactive Collective Intelligence Support Protocols - https://mariadb.com/services/remote-dba/
27. AWS Database Migration Service (DMS) Homogeneous vs. Heterogeneous Schema Limitations - https://docs.aws.amazon.com/dms/latest/userguide/Welcome.html
28. Automating Structural Refactoring via the MariaDB Migration Assessment Tool - https://mariadb.com/migrations/migration-assessment-tool/



mariaDB

Get started with MariaDB Cloud for free

mariadb.com