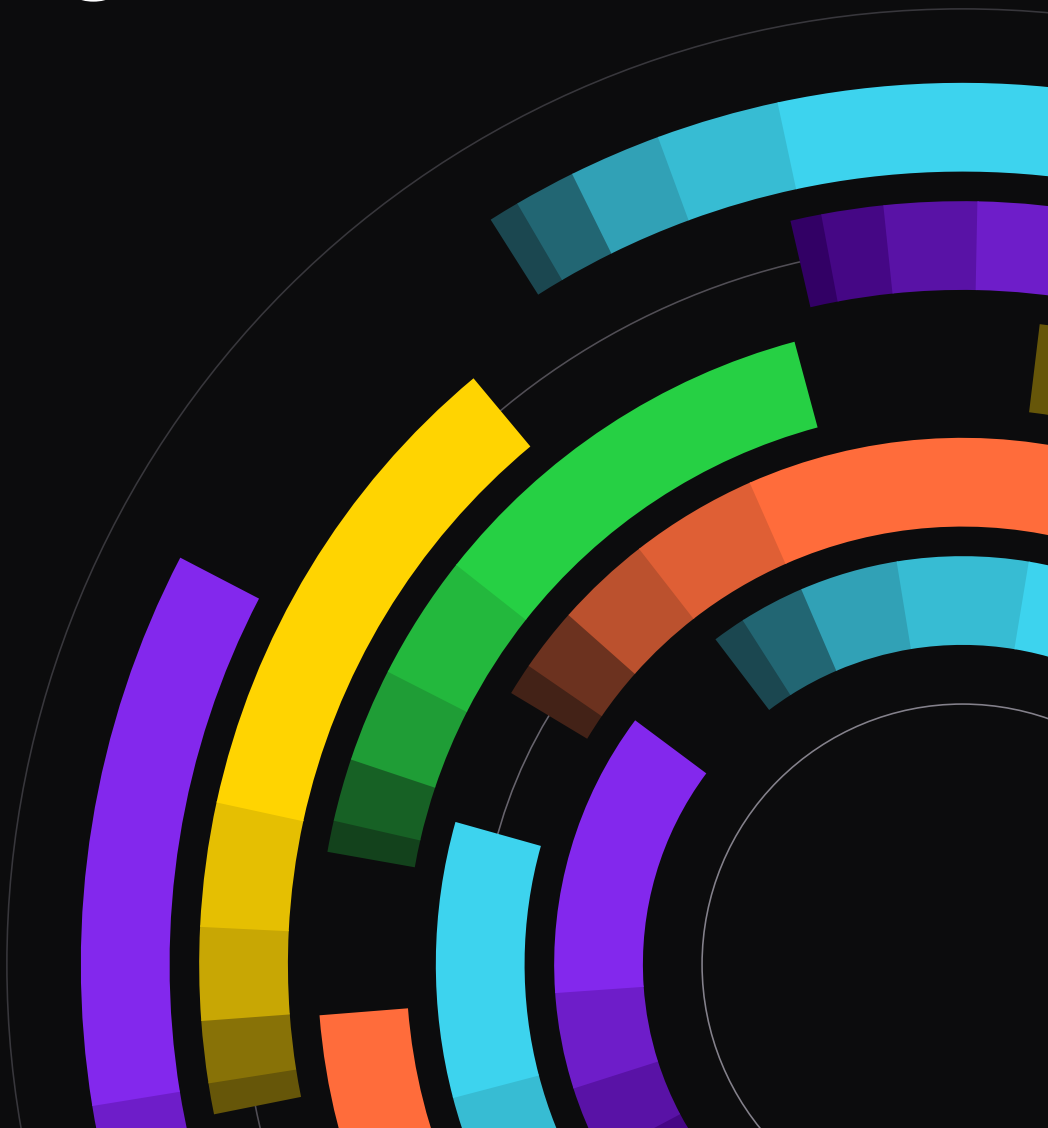
mariaDB

WHITE PAPER

MariaDB Cloud: The case for migration



CONTENTS

Table of contents

01 **Executive summary** 3

- Headline findings 3
- Two parts to the argument 4
- Headline qualitative claims 4
- The combined claim 4

02 **Part 1: Total cost of ownership** 5

- Methodology and assumptions 6
- Deliberate scope exclusions 6
- The RetailCo illustrative case study fleet configuration 7
- The cost stack line by line 9
- Why these numbers are conservative 11

03 **Part 2: Application experience and security** 13

- MariaDB Cloud reference architecture 13
- Pillar 1: Zero-downtime failover via MaxScale 15
- Pillar 2: White-glove dedicated support 17
- Pillar 3: Multi-cloud portability 18
- Pillar 4: Security posture via BYOA 20

04 **Conclusion** 25

- The TCO case 25
- The application experience and security case 25
- The combined argument 26
- Next step: TCO and architecture validation workshop 26

05 **Appendix A: Detailed cost stack derivation** 27

06 **Appendix B: Glossary** 30

01

Executive summary

Enterprises rarely run a single MariaDB database in production. A typical mid-market or large enterprise customer operates a fleet – often a dozen or more production environments split across business units, customer tiers, or geographies, plus matching UAT environments and a longer tail of development environments. When the unit of comparison shifts from a single database to a fleet, the case for moving from AWS RDS for MariaDB to MariaDB Cloud strengthens substantially: not only do the costs compound, but the operational and security limitations of running databases inside an AWS-managed perimeter compound alongside them.

This paper makes the case for migration along two dimensions, both quantified to the extent the methodology permits and presented with auditable underlying inputs. The first is total cost of ownership: at list prices on both sides, the comparison favors MariaDB Cloud by approximately 47% on annual TCO for the illustrative case study used throughout this paper. The second is application experience and security: across four operational pillars – zero-downtime failover, white-glove dedicated support, multi-cloud portability, and the security posture enabled by MariaDB Cloud's Bring-Your-Own-Account (BYOA) deployment model – MariaDB Cloud delivers structural advantages that AWS RDS cannot match by design.

Headline findings

The illustrative case study used throughout this paper is RetailCo, a hypothetical $10 billion retail e-commerce operator running a 60-environment MariaDB fleet plus a consolidated analytics pipeline. RetailCo is composite – it does not represent any specific customer – and the figures below are illustrative of the magnitude of savings available at this fleet shape, not a forecast of the customer's actual saving. Detailed configuration and methodology are presented in Part 1.

METRIC	ILLUSTRATIVE VALUE (RETAILCO)
Environment fleet	15 production + 15 UAT + 30 development + 1 analytics
Annual TCO on AWS RDS for MariaDB	$2,898,185
Annual TCO on MariaDB Cloud	$1,537,920
Annual saving	$1,360,265 (47%)
Three-year cumulative net saving	$3,980,795
Migration cost (one-time)	$100,000
Payback period	Under 1 month (~0.9 months)

Two parts to the argument

This paper is structured in two parts. Part 1 develops the total cost of ownership case in detail: methodology, the RetailCo case study configuration, the line-by-line cost stack, and the conservatism of the underlying inputs. Part 2 develops the case for application experience and security: the MariaDB Cloud reference architecture with annotated failover sequence, and four operational pillars covering zero-downtime failover, white-glove dedicated support, multi-cloud portability, and the security perimeter implications of BYOA.

The two parts are designed to be read together but stand alone. A finance reader can stop at the end of Part 1 and have a defensible TCO model. A security or engineering reader can skim Part 1 and engage Part 2 in depth without losing context. The conclusion ties both back together for a buyer who needs the combined argument.

Headline qualitative claims

Beyond the dollars, MariaDB Cloud delivers four structural advantages over AWS RDS for MariaDB that are detailed in Part 2:

- **Zero-downtime failover via MaxScale:** instant connection routing keeps applications online during node failure or planned maintenance, compared to AWS RDS Multi-AZ failover, which takes 90–120 seconds and causes application-side disruption.
- **White-glove dedicated support via the fractional DBA service model:** direct access to MariaDB engineers who built the engine, with proactive performance reviews, capacity planning, and architectural guidance – not the generic database-agnostic Enterprise Support tiers offered by AWS.
- **Multicloud portability across Amazon Web Services (AWS), Microsoft Azure (Azure), and Google Cloud Platform (GCP):** enables cross-cloud disaster recovery and reduces single-hyperscaler lock-in, in contrast to AWS RDS, which by definition runs only on AWS.
- **Security posture via BYOA on AWS:** the MariaDB Cloud BYOA deployment model places database compute and storage inside the customer's own AWS account, governed by the customer's Access Management (IAM), Key Management Service (KMS), and audit pipelines. This contrasts sharply with the AWS RDS architecture, in which the actual database host runs in an AWS-managed account opaque to the customer's security tooling. AWS BYOA reaches general availability on June 30, 2026.

The combined claim

MariaDB Cloud is materially less expensive, structurally more available, operationally better supported, and architecturally more sovereign than AWS RDS for MariaDB.

These advantages are not narrowly distributed across a few specialized use cases – they apply to the typical fleet-scale enterprise deployment that constitutes the majority of MariaDB workloads on AWS today. The cost advantage compounds with environment count. The application-experience advantage compounds with availability requirements. The security advantage compounds with regulatory scope. For a customer evaluating their cloud database posture in 2026, the case for migration is more than a cost story.

02

Part 1: Total cost of ownership

Part 1 quantifies the cost difference between AWS RDS for MariaDB and MariaDB Cloud at fleet scale. The analysis is illustrated through the RetailCo case study – a hypothetical $10 billion retail e-commerce operator running 15 production, 15 UAT, and 30 development MariaDB environments, plus a consolidated analytics pipeline that replaces the typical AWS RDS → AWS DMS → AWS S3 → AWS Redshift stack with a single managed product. RetailCo is composite – it does not represent any specific customer – and the figures shown are illustrative of the magnitude of saving available at this fleet shape, not a forecast of the customer's actual saving.

At list prices on both sides, RetailCo's annual TCO is approximately $2.9 million on AWS RDS and $1.54 million on MariaDB Cloud, yielding annual savings of $1.36 million per year (47%) and a payback period of under one month on a typical $100,000 migration. The savings come from five compounding categories, ranked by absolute dollar contribution:

- **Auto-scaling on production and UAT environments – $587,150 (43%):** AWS RDS for MariaDB cannot scale compute vertically without downtime (there is no Compute Auto-scaling), so customers peak-provision 24×7. MariaDB Cloud can be configured to auto-scale compute without downtime, with billing tracking actual size minute-by-minute. Modeled at peak load = 2× normal. The Cluster Compute can be configured to automatically scale horizontally (add more replicas) or scale vertically based on sustained CPU usage or concurrent sessions.
- **Active-active architecture eliminating idle Multi-AZ standby – $282,055 (21%):** AWS Multi-AZ provisions a synchronous standby that does not serve read traffic. MariaDB Cloud's active-active architecture deploys a fully active second node, removing the idle vCPU spend.
- **Bundled platform stack – $250,158 (18%):** Connection proxy, observability, monitoring, and managed analytics that are separately billed on AWS are bundled in MariaDB Cloud's scale-fabric pricing.
- **Avoided forced major-version upgrade events – $150,000 (11%):** MariaDB Enterprise Server provides 8 years of supported life. AWS RDS runs Community Server's 3-year LTS window, forcing a fleet-wide upgrade approximately every three years.
- **Analytics pipeline collapse – $90,900 (7%):** MariaDB Exa replaces a multi-service Redshift pipeline with a single managed product fed by internal change-data-capture (CDC), eliminating orchestration components and labor.

These figures are deliberately conservative. List prices are used on both sides, DBA labor savings are excluded entirely from the model, and the recently announced BYOA deployment option – which would let customers apply existing AWS commitments and savings plans to the database compute – is not credited in the headline figure.

The remainder of Part 1 develops these claims with full methodology, fleet configuration, the complete cost stack, and four explicit reasons the numbers are conservative.

Methodology and assumptions

The cost model in this paper uses a list-price-to-list-price comparison methodology with a three-year analysis horizon. This section documents the methodology choices, the pricing inputs, and the deliberate scope exclusions, so the customer can replicate the analysis with their own inputs.

Composite customer. A single illustrative case study – RetailCo – is used throughout. RetailCo is a hypothetical aggregate of typical industry profiles in the retail and e-commerce sector at the $5–15 billion annual revenue scale, not any specific identifiable enterprise. The composite was constructed to reflect a fleet shape and workload pattern characteristic of mid-to-large enterprise customers running production MariaDB workloads on AWS today: multiple production database tiers serving customer-facing transactional workloads, a parallel UAT footprint, a longer tail of development environments, and an analytics layer fed from production via change data capture.

RetailCo is illustrative – it is intended to show the magnitude of saving available at this fleet shape, not to forecast any specific customer's actual saving. A customer evaluating their own situation should expect their actual figures to differ from RetailCo's.

Time horizon and discounting. Three-year analysis horizon. Net present value time-discounting is omitted to keep the comparison auditable in nominal-dollar terms. Readers who wish to apply a discount rate of 8–12% can do so by deflating the Year 2 and Year 3 cash flows; the conclusion does not change materially.

Pricing inputs. AWS pricing is sourced from publicly listed AWS RDS for MariaDB pricing in US East (N. Virginia) at the publication date, with Multi-AZ pricing computed as 2× Single-AZ rates per AWS's documented model. MariaDB Cloud pricing uses the publicly listed Foundation Tier rates with the 50% scale-fabric surcharge applied to active-active configurations. Storage on both sides is priced at $0.23 per GB per month – AWS gp3 Multi-AZ storage matches MariaDB Cloud bundled storage at this rate. AWS RDS Proxy is priced at $0.015 per vCPU-hour against the active vCPU count; MariaDB Cloud's intelligent proxy (MaxScale) is bundled in the scale-fabric surcharge.

Deliberate scope exclusions

Two categories of cost difference are deliberately excluded from the model so that the headline figures remain conservative and customer-replicable:

- **DBA labor savings are excluded by design.** MariaDB Cloud's fractional DBA service can replace meaningful in-house operational headcount in many customer environments, but labor savings vary too widely by organization, geography, and team structure to model defensibly in a generic comparison. The savings shown here are achieved purely through platform pricing and architectural differences. Any DBA labor reduction the customer realizes is additional upside beyond what is quantified.
- **BYOA cost capture is excluded by design.** MariaDB Cloud's Bring-Your-Own-Account (BYOA) deployment model – generally available on AWS as of June 30, 2026 – allows customers to deploy database compute and storage directly into their own AWS account, where it can draw down existing Reserved Instance commitments, Savings Plans, EDP discounts, and negotiated pricing. The headline figures in this paper do not credit any of this cost capture. For customers with significant existing AWS commitments, BYOA is a real source of additional saving on top of the figures presented here.

Auto-scaling waste reduction methodology. Auto-scaling savings are included as a quantified line item, modeled with a single simple assumption: peak load is approximately 2× normal load. AWS RDS for MariaDB cannot scale compute vertically without downtime, so the customer must peak-provision 24×7 – paying for 2× the average required capacity, continuously. MariaDB Cloud auto-scales without downtime, billing for actual size minute-by-minute. The effective compute saving on auto-scaled environments is therefore approximately 50% on the affected compute lines.

Auto-scaling savings apply to production and UAT environments only; development environments are sized as fixed single-node deployments on both sides.

Illustrative line items. Two line items in the model are computed from per-environment unit economics rather than vendor-published list prices: CloudWatch monitoring and observability, and the cost of major-version upgrade events. Both are walked through in detail in the methodology appendix so that readers running their own analysis can substitute their own observed numbers. The figures used in the headline tables represent typical observed values for fleet-scale RDS deployments, not floor or ceiling estimates.

The RetailCo illustrative case study fleet configuration

RetailCo is a hypothetical $10 billion retail e-commerce operator. The case study is illustrative – not a real customer – and the figures shown are intended to demonstrate the magnitude of saving available at this fleet shape, not to forecast any specific customer's actual saving. The fleet configuration described below was constructed to be representative of customer-facing transactional retail at scale: multiple production database tiers serving the storefront, order management, inventory, and fulfillment systems, with parallel UAT environments matching production at a smaller instance footprint, and a development tail supporting feature work, integration testing, and data engineering.

Annual revenue and operational profile.

ATTRIBUTE	VALUE
Annual revenue	`$10 billion`
Industry	Retail e-commerce, omnichannel
Geographic footprint	Multiple regions, single primary AWS region
Database engine standardization	MariaDB across the transactional fleet
Application architecture	Microservices on EC2, with database-backed transactional workloads
Workload profile	Daily peaks aligned to retail business hours, weekend traffic dips, seasonal spikes

Fleet configuration environments. RetailCo operates 60 MariaDB environments across production, UAT, and development tiers, plus a consolidated analytics pipeline fed from production via change data capture. The breakdown below shows the matched-active-vCPU configuration on both AWS RDS and MariaDB Cloud – that

is, the AWS sizing and MariaDB Cloud sizing chosen so that both sides deliver the same active vCPU capacity to the application. The AWS Multi-AZ standby is paid for but does not serve traffic; the MariaDB Cloud second node is fully active.

TIER	COUNT	AWS RDS SIZING	MARIADB CLOUD SIZING
Production	15	db.r6g.8xlarge Multi-AZ	`2 × Sky-16x128 active-active`
UAT	15	db.r6g.4xlarge Multi-AZ	`2 × Sky-8x64 active-active`
Development	30	db.t4g.large Single-AZ	Sky-2x4 single-node
Analytics	1	Redshift + DMS + S3	MariaDB Exa, 32 vCPU + CDC

The production tier serves storefront, order management, inventory, customer profile, and fulfillment workloads. Each production environment averages roughly 2 TB of data with growth of approximately 30% per year. UAT environments mirror production at half the instance class for pre-production testing. Development environments support feature work, integration testing, and data engineering, with usage typically concentrated in business hours and absent during nights and weekends.

The analytics environment in the AWS configuration reflects the typical pattern for retail customers: production data is replicated via DMS into S3, then loaded into Redshift for business intelligence and reporting workloads. The MariaDB Cloud configuration replaces this pipeline with MariaDB Exa, a managed columnar analytics product fed directly from production via internal CDC, eliminating the DMS, S3 staging, and Redshift coordination layers.

Workload duty cycle. Production and UAT environments are assumed to operate on a 24×7 schedule, with peak load approximately 2× normal load. Peak load occurs during retail business hours and seasonal events; normal (off-peak) load occurs during nights, weekends, and between seasonal peaks. On AWS RDS, this duty cycle requires peak-provisioning every environment 24×7 because vertical compute scaling cannot occur without downtime. On MariaDB Cloud, vertical auto-scaling tracks actual size minute-by-minute, so customers pay for peak load only when traffic is at peak.

Development environments are assumed to operate on a business-hours-only schedule but are sized as fixed single-node deployments on both sides, since the savings from compute auto-scaling on development workloads are typically modest. The auto-scaling line item in the cost stack therefore applies only to the production and UAT tiers.

Why this case study is representative. The fleet shape and workload profile described above is characteristic of mid-to-large enterprise retailers running customer-facing transactional MariaDB on AWS. Smaller retailers (under $1 billion revenue, sub-20 environment fleet) will see proportionally smaller absolute savings in dollar terms but similar percentage savings; larger retailers (over $25 billion revenue, 100+ environment fleet) will see proportionally larger absolute savings and similar or higher percentage savings, because the bundling and architectural-efficiency gains scale with environment count.

The cost stack line by line

This section presents the annual cost decomposition for the RetailCo case study at list prices on both sides. Each line is shown for both AWS RDS for MariaDB and MariaDB Cloud, with the absolute saving and the percentage saving on that line. The total at the bottom is the figure carried forward into the headline metrics.

The lines are presented in the same order as the cost stack used by the cost-center finance team, not in order of dollar size.

Cost stack annual.

LINE ITEM	AWS RDS	MARIADB CLOUD	SAVING	% SAVING
Compute (15 prod + 15 UAT + 30 dev, 24×7 peak provisioning)	$1,455,375	$1,203,420	$251,955	17%
Storage (gp3 / bundled, including dev tier)	$625,140	$629,280	-$4,140	-1%
Connection proxy (RDS Proxy / MaxScale bundled)	$94,608	included	$94,608	100%
Standard support (10% of compute)	$217,512	$183,272	$34,240	16%
CloudWatch + Database Insights observability	$155,550	included	$155,550	100%
Major-version upgrade event (amortized)	$150,000	$0	$150,000	100%
Analytics pipeline (Redshift+DMS+S3 vs. Exa)	$200,000	$109,100	$90,900	45%
Auto-scaling adjustment (prod + UAT, peak = 2× normal)	$0	-$587,150	$587,150	–
ANNUAL TOTAL	$2,898,185	$1,537,920	$1,360,265	47%

Cost stack re-ordered by absolute saving. The same line items, re-ordered to make the contribution of each saving lever explicit. This view is the most useful for a finance reader trying to understand which architectural and pricing differences are driving the overall saving.

SAVING DRIVER	ANNUAL SAVING	% OF TOTAL SAVING
Auto-scaling on production and UAT (active vCPU billed minute-by-minute, peak = 2× normal)	$587,150	43%
Active-active architecture (no idle Multi-AZ standby) + small support delta	$282,055	21%
Bundled platform stack – proxy, observability, monitoring	$250,158	18%
Avoided forced major-version upgrade events (8-year vs. 3-year support)	$150,000	11%
Analytics pipeline collapse (Exa replaces RDS+DMS+Redshift stack)	$90,900	7%
ANNUAL TOTAL SAVING	$1,360,265	100%

Three observations are worth flagging from this view:

- **Auto-scaling alone is the largest single line, at 43% of total savings.** AWS RDS for MariaDB cannot scale compute vertically without downtime – a structural limitation of the service, not a pricing artifact. Customers facing this limitation peak-provision 24×7 and pay for the over-provisioned headroom continuously. MariaDB Cloud's auto-scaling without downtime is the single most important architectural difference quantified in this paper.
- **Architecture and bundling together account for roughly 39% of total savings.** Active-active architecture eliminates the idle Multi-AZ standby, and bundled platform components (MaxScale proxy, observability) replace separately billed AWS line items. These savings exist independent of workload duty cycle – a customer with a flat 24×7 utilization profile would still capture them.
- **The avoided major-version upgrade event is real money even though it occurs every three years.** Annualized at one-third of the upgrade cost, this contributes 11% of total saving. Customers should think of this less as a one-time cost and more as a structural avoided expense – the upgrade events keep coming.

Three-year cumulative summary. Multiplied across a three-year analysis horizon, net of the one-time migration cost:

YEAR	ANNUAL SAVING	LESS MIGRATION COST	CUMULATIVE NET
Year 1	$1,360,265	-$100,000	$1,260,265
Year 2	$1,360,265	–	$2,620,530
Year 3	$1,360,265	–	$3,980,795

Three-year cumulative net saving: approximately $3.98 million. Computed as: 3 × $1,360,265 annual saving, less $100,000 one-time migration cost. Year 1 payback occurs at month 0.9 – the recurring monthly saving (~$113,000) exceeds the migration cost within the first calendar month after cutover.

Why these numbers are conservative

The figures presented are deliberately conservative. Four methodology choices and explicit scope exclusions bias the analysis toward AWS RDS rather than against it. The customer's real saving on a like-for-like fleet migration is likely to be larger, sometimes substantially larger, than the figures shown.

Reason 1 – List prices on both sides; no commit discounts applied. Neither AWS Reserved Instances, AWS Savings Plans, nor MariaDB Cloud commit discounts are applied in the headline figures. Both vendors offer commit-based discounts of approximately 30–60% depending on term length and commitment level, and customers should negotiate directly with their vendor representatives. The percentage saving reported in this paper holds approximately when discounts are applied symmetrically on both sides – a customer with a 40% AWS RI discount and a 40% MariaDB Cloud commit discount will still see roughly 47% saving on annual TCO.

This is a deliberate methodology choice. List-price comparison is the only comparison that is reproducible without access to vendor-specific commercial terms. Adding hypothetical discount layers introduces non-auditable assumptions and ages poorly as commercial terms evolve.

Reason 2 – DBA labor savings excluded entirely. MariaDB Cloud's fractional DBA service can replace meaningful in-house operational headcount in many customer environments. The fractional DBA model provides direct access to engineers who built the MariaDB engine, with proactive performance reviews, capacity planning, migration guidance, and architectural review included as part of the service.

None of this is credited as a saving in the cost model. The reason is purely methodological: labor savings vary too widely by organization, geography, and team structure to model defensibly. A customer whose DBA team currently spends meaningful time on RDS upgrade preparation, observability tuning, RDS Proxy configuration, parameter group management, or troubleshooting Multi-AZ failover events will recover those cycles when migrated to MariaDB Cloud. The dollar value of those recovered cycles is real but not quantified here.

Reason 3 – Strategic optionality not modeled. Several MariaDB Cloud capabilities have real economic value that is specific to customer circumstances and therefore not modeled in the headline figures:

- **Multi-cloud portability** across AWS, Azure, and GCP enables cross-cloud disaster recovery architectures and reduces single-hyperscaler lock-in. The economic value of reduced lock-in (improved negotiation leverage on hyperscaler renewal cycles, ability to relocate workloads in response to regulatory or pricing changes) is real but customer-specific.
- **Serverless-tier savings** on development environments – MariaDB Cloud's serverless option scales compute to zero during off-hours, eliminating idle development environment spend during nights and weekends. This is not modeled in the headline figures because the development environments are sized as fixed single-node deployments on both sides.
- **Cross-region disaster recovery cost differentials** when customers actually deploy disaster recovery (DR) – the AWS data transfer charges for cross-region replication can be material at fleet scale. MariaDB Cloud's bundled cross-region replication option carries different economics.

Reason 4 – BYOA cost capture not modeled in headline figures. This is a new factor that did not exist when prior versions of this analysis were published. MariaDB Cloud's Bring-Your-Own-Account (BYOA) deployment model is generally available on AWS as of June 30, 2026 (Azure GA preceded; GCP follows). Under BYOA,

database compute and storage are deployed directly into the customer's own AWS account rather than into MariaDB's managed cloud account.

With BYOA on AWS, the underlying database compute and storage hit the customer's AWS bill rather than being bundled into the MariaDB Cloud monthly charge. This means the customer's existing AWS commercial commitments – Reserved Instances, Savings Plans, Enterprise Discount Program (EDP) tiers, and any negotiated infrastructure pricing – apply to the database infrastructure as well. For a customer with 30–50% effective AWS commit discount on their compute spend, this represents a material additional saving.

Critically, the architectural sources of saving developed throughout this paper still apply under BYOA. Active-active architecture still eliminates the idle Multi-AZ standby. Auto-scaling without downtime still removes the need to peak-provision 24×7. The platform stack is still bundled. The major-version upgrade event is still avoided. BYOA changes who pays for the underlying infrastructure (the customer's AWS account, drawing down their commitments), not how much underlying infrastructure is required to deliver the same performance. The result is a combined saving that compounds the architectural advantages with the customer's existing AWS commercial position.

The headline figures in this paper do not credit any of this BYOA cost capture. For an AWS-committed customer evaluating BYOA, the true total saving is approximately the figures shown plus the customer's effective AWS discount rate applied to the underlying compute and storage that move into BYOA. This is real money – typically several hundred thousand dollars per year for a fleet of RetailCo's scale, depending on the customer's commit discount level.

The 47% saving figure is a floor, not a ceiling. List prices on both sides, DBA labor savings excluded, strategic optionality unmodeled, and BYOA cost capture omitted. The customer's actual realized saving on a like-for-like fleet migration will frequently be larger – and for AWS-committed customers adopting BYOA, materially larger.

03

Part 2: Application experience and security

Part 2 makes the case that goes beyond the dollar saving developed in Part 1. MariaDB Cloud delivers structural advantages over AWS RDS for MariaDB across four operational pillars that a typical CFO model does not capture: zero-downtime failover, white-glove dedicated support, multi-cloud portability, and security posture via the BYOA deployment model. These are not differentiators that a procurement-led RFP exercise tends to surface – they are properties of the underlying architecture and of the operational service wrapped around it.

The four pillars are introduced briefly here and developed in detail in the sections that follow. Each pillar is presented at peer depth:

- **Pillar 1 – Zero-downtime failover via MaxScale:** MariaDB Cloud's active-active topology with MaxScale-mediated routing keeps applications online during node failure. AWS RDS Multi-AZ failover takes 90–120 seconds with application-side disruption – a real revenue impact for retail at peak hours and a real availability gap for any customer-facing application.
- **Pillar 2 – White-glove dedicated support:** MariaDB Cloud's fractional DBA service model provides direct access to engineers who built the MariaDB engine, with proactive performance review, capacity planning, migration support, and architectural guidance. AWS Enterprise Support is a generic, database-agnostic tier; the named support engineer assigned to a customer is not, in general, a database expert.
- **Pillar 3 – Multi-cloud portability:** MariaDB Cloud runs natively on AWS, Azure, and GCP with the same architecture and operational model. AWS RDS by definition runs only on AWS. The strategic implications – cross-cloud disaster recovery, hyperscaler negotiation leverage, regional data residency flexibility – are real and compound over time.
- **Pillar 4 – Security posture via BYOA:** MariaDB Cloud BYOA on AWS, generally available June 30, 2026, deploys database compute and storage directly into the customer's AWS account. The customer's IAM, KMS, audit, and compliance tooling extend to the database tier. AWS RDS by architectural design places the actual database host in an AWS-managed account that the customer cannot see, audit, or apply their security tooling to.

MariaDB Cloud reference architecture

This section presents the reference architecture that underpins the operational pillars. The architecture is intentionally described at a conceptual altitude – sufficient to understand the failover sequence and the role of each component, without reproducing implementation-level deployment details that vary across cloud providers and customer environments.

The diagram below shows the steady-state architecture with a numbered overlay (1 through 4) indicating what happens during a node failure.

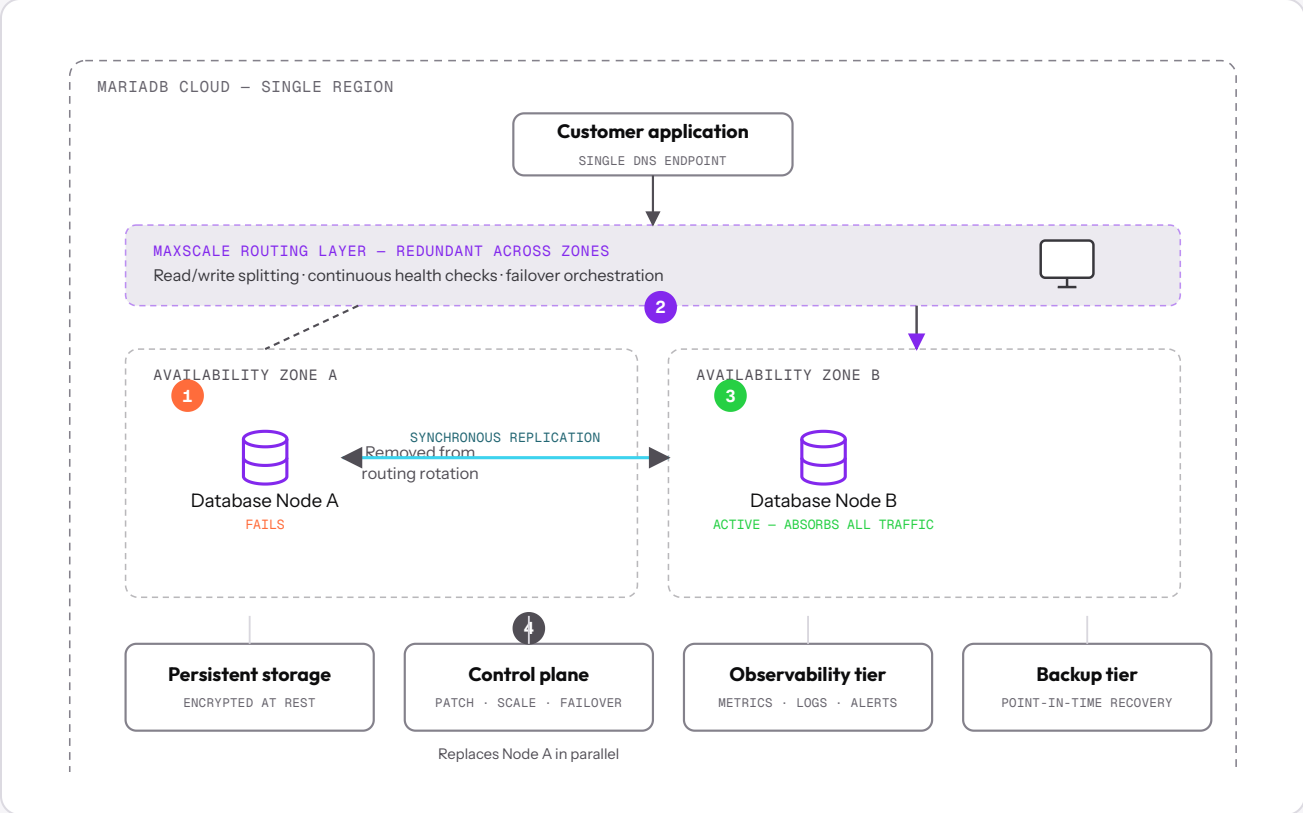


FIGURE 1 MariaDB Cloud reference architecture with annotated failover sequence.

Architecture components. The reference architecture consists of nine logical components, deployed across two availability zones in a single region:

- **Customer application:** the application servers that connect to the database via a single DNS endpoint. The application requires no awareness of the underlying topology – connection strings, security groups, and TLS configuration are independent of which database node is currently serving any given query.
- **MaxScale routing layer:** MariaDB's intelligent database proxy. MaxScale terminates client connections, performs read/write splitting (writes to the primary node, reads distributed across active nodes), runs continuous health checks against the database nodes, and orchestrates the rerouting of connections during a failure event. MaxScale is deployed redundantly across availability zones so that the routing layer itself is not a single point of failure.
- **Database Node A (Active):** a fully active MariaDB instance in availability zone A. Serves both read and write traffic, with the engine running continuously. State is kept current with Node B via synchronous replication.
- **Database Node B (Active):** a fully active MariaDB instance in availability zone B. Serves both read and write traffic. This is the architectural difference from AWS RDS Multi-AZ – in the AWS architecture, the second node is a passive standby that does not serve any traffic, only replicates state from the primary. In MariaDB Cloud's active-active architecture, both nodes are paid for and both nodes work.
- **Persistent storage layer:** encrypted at rest with per-AZ replication and block-level snapshots. The storage layer is logically shared across the two database nodes for replication efficiency, with physical durability

backed by multi-AZ replication of the underlying block storage.

- **Synchronous replication channel:** the channel by which Node A and Node B keep their state current with each other. The replication is synchronous – a write is acknowledged to the application only after it has been durably committed to both nodes, ensuring no data loss in the event of node failure.
- **Control plane:** the orchestration layer that manages the database lifecycle. Handles automated provisioning, patching and upgrades, failover orchestration, auto-scaling decisions, and backup scheduling. The control plane communicates with the database nodes and MaxScale instances via privileged APIs.
- **Observability tier:** metrics, dashboards, query and slow logs, audit logs, and alerting. Bundled in the MariaDB Cloud service rather than separately billed (the equivalent of AWS CloudWatch metrics, custom metrics, log ingestion, and Database Insights, all of which are separately metered on AWS).
- **Backup tier:** continuous, incremental backup with point-in-time recovery. Cross-region backup replication is available as a standard configuration option.

Why active-active matters. The single most important architectural point in this diagram is that both database nodes are active. This has both economic and operational consequences.

Economically, the customer is not paying for an idle standby. The Multi-AZ deployment in AWS RDS provides a synchronous standby instance whose vCPUs, memory, and storage are billed at full rate but which cannot serve read traffic – it exists purely to be promotable in the event of primary failure. The matched-active-vCPU sizing methodology used in Part 1's TCO analysis accounts for this: AWS sizing of (for example) db.r6g.8xlarge Multi-AZ delivers 32 active vCPUs at full rate, while MariaDB Cloud sizing of 2 × Sky-16×128 active-active delivers 32 active vCPUs across two nodes that both serve "read" traffic.

Operationally, the consequence is more important. Because Node B is already running and already with Node A's state, failover is not a process of starting up a standby – it is a process of removing Node A from the routing rotation. There is no warm-up time, no cache rebuild, no DNS propagation. The architecture turns failover from a multi-minute procedure into a sub-second routing change.

Pillar 1: Zero-downtime failover via MaxScale

Failover behavior is the single most important availability characteristic of a managed database service. It is what determines whether a node failure produces a brief connection-retry blip or a multi-minute customer-facing outage. AWS RDS Multi-AZ and MariaDB Cloud differ structurally in their failover model, and the difference is the gap between an availability incident and an availability event.

AWS RDS Multi-AZ failover. AWS RDS Multi-AZ failover takes 90 to 120 seconds in typical conditions, with application-side disruption throughout the window. The mechanism is as follows: the standby goes through database "crash recovery," then, Standby is promoted to primary, the database endpoint's DNS record is updated to point to the new primary, and the application's connection pool gradually reestablishes connections to the new primary as the DNS update propagates and as cached connections fail.

This is not a bug – it is the architectural consequence of an active-passive topology with DNS-mediated endpoint resolution. The standby cannot start serving traffic instantly because (a) it is not actively running queries, so its buffer pools and caches are cold, and (b) the application's clients cannot reach it until DNS

resolution flips. Each of these is a multi-second process, and they happen sequentially.

During the 90- to 120-second window, the application experiences:

- **Connection errors.** In-flight queries and existing connections to the failed primary fail. The application's connection pool sees a cascade of connection-refused or connection-reset errors as the existing connections die.
- **Failed retries.** Application-level retry logic typically retries against the cached endpoint, which still resolves to the failed primary until DNS propagation completes. These retries fail.
- **Customer-visible errors.** Whatever the application surfaces during a database outage – error pages, transaction failures, abandoned shopping carts – surfaces during the window.

AWS publishes a 99.95% Multi-AZ availability SLA, which works out to approximately 22 minutes of downtime per month. Multi-AZ failover events are billed against this SLA as expected behavior, not as outages.

MariaDB Cloud MaxScale failover. MariaDB Cloud's failover sequence completes in under one second from node failure to traffic restoration on the surviving node. The mechanism is fundamentally different:

- Node A becomes unavailable due to instance failure, network partition, or maintenance event.
- MaxScale's continuous health checks detect the failure within sub-second timeframes. MaxScale removes Node A from the routing rotation.
- All read and write traffic now flows to Node B, which was already active and serving traffic. No data loss – synchronous replication kept Node B's state current with Node A up to the moment of failure. Application connections re-resolve to Node B's MaxScale endpoint via the same TCP path; existing connections experience a single transient retry.
- Application sees a single transient connection retry, then resumes normal operation. No DNS change, no manual intervention, no operator paging required. The control plane begins replacing Node A in parallel with continued service from Node B.

The architectural reason this is possible is that Node B is not a standby being promoted – it is an active node already serving traffic. The failover is a routing change, not a state-machine transition. The customer's connection pool on the application side sees one failed query (the one in flight at the moment of failure), and the next query succeeds against Node B.

Business impact at retail peak. For a retail e-commerce operator like RetailCo, the difference between 90–120 second failover and sub-second failover is real money during peak hours.

A typical mid-market e-commerce site running through a peak hour processes orders at a rate that translates to approximately $5,000 to $15,000 of revenue per minute, depending on seasonality and promotional cadence. During a flash sale, on Black Friday, or during a viral product moment, the rate can be substantially higher. A 90–120 second outage during such a window costs $7,500 to $30,000 in immediate lost transaction revenue, plus the second-order effects: abandoned carts that don't return, customer trust damage, support ticket spikes, and elevated refund and chargeback rates from confused customers.

Multi-AZ failover events do happen. AWS commits to 99.95% Multi-AZ availability, which translates to roughly 22 minutes of allowed downtime per month. A retailer with 15 production environments experiences, in expectation, approximately 22 × 15 = 330 environment-minutes of allowed downtime per month – distributed across the fleet, but real. At even half the allowed rate, this is 11 minutes per month of failover-induced disruption fleet-wide. At RetailCo's revenue scale, that translates to several hundred thousand dollars of annual exposure that is not modeled in the Part 1 cost stack.

MariaDB Cloud's sub-second failover does not eliminate every form of database availability incident, but it eliminates the specific category of incident that AWS RDS Multi-AZ failover produces by design: the multi-minute application-side disruption during routine node failure or planned maintenance. The architectural difference compounds across the fleet.

Planned-maintenance windows. The same architectural difference applies to planned maintenance – patching, minor-version upgrades, instance type changes, and other operations that touch the database hosts. On AWS RDS, these operations either require a Multi-AZ failover (with the same 90–120 second customer-visible window) or extend the maintenance window beyond what the customer's change-management process can accommodate.

On MariaDB Cloud, the active-active topology lets the operations team patch one node at a time while the other continues to serve traffic, then route traffic to the patched node and patch the second. The customer-visible impact is the same as a node failure event: a single transient retry per connection, no application-level error. This eliminates the typical customer pattern of accumulating a backlog of postponed maintenance windows because the change-management cost of each window exceeds the operational cost of waiting.

Pillar 2: White-glove dedicated support

Database support is a category where the difference between a fully managed service from a database vendor and a fully managed service from a cloud provider is operationally meaningful. The two services look superficially similar – both promise 24×7 incident response, both promise expert assistance with database operations – but the substance of what is delivered differs in ways that show up in customer outcomes. The table below captures the substantive difference at peer depth.

DIMENSION	AWS ENTERPRISE SUPPORT	MARIADB CLOUD (INCL. FRACTIONAL DBA)
Support engineer specialization	AWS-service generalist	MariaDB engine specialist
Database expertise depth	Generic; tickets often escalate to RDS team	Direct engine knowledge; many engineers are codebase contributors
Engagement model	Reactive – customer opens ticket, support responds	Reactive support + proactive fractional DBA engagement
Performance review cadence	Not included	Included – quarterly or per agreed cadence
Capacity planning support	Not included	Included – sizing reviews, growth modeling
Migration and architecture guidance	Limited; AWS Professional Services is separate	Included; MariaDB engineers participate in architecture decisions
Pricing	`3-10% of monthly AWS spend`	Bundled in MariaDB Cloud price

The fractional DBA model is the highest-value piece of the offering. A named MariaDB engineer is assigned to the customer's account and dedicates a defined fraction of their time to ongoing MariaDB operations on a recurring basis – quarterly performance reviews, capacity planning, schema design consultation, migration support, and operational event participation. This is structurally different from reactive ticket-based support and from full-time hired DBAs (which can cost $200,000–$300,000 fully loaded for a single experienced engineer). It gives the customer ongoing access to senior database expertise at a price point meaningfully below an in-house hire.

The economic dividends are not credited to the headline TCO figures, but customer outcomes are observable: customer DBA teams typically recover 0.5 to 1.0 FTE-equivalent of capacity from RDS-specific operational tasks (RDS Proxy configuration, parameter tuning, upgrade preparation, Multi-AZ troubleshooting), database-tier incident volume drops measurably, and capacity planning accuracy improves. White-glove support matters most during major events (migrations, version upgrades, peak seasonal events), when in-house DBA capacity is constrained, and when the application is customer-facing and revenue-critical – all conditions characteristic of the customer profile this paper addresses.

Pillar 3: Multi-cloud portability

AWS RDS for MariaDB runs only on AWS. This is a definitional fact, not a limitation that AWS could relax. The service is part of the AWS service catalog, and the database hosts are EC2 instances in AWS-managed accounts. A customer running their MariaDB workloads on AWS RDS has, by adopting that service, made a decision to host that database tier on AWS for the foreseeable future.

MariaDB Cloud runs on AWS, on Microsoft Azure, and on Google Cloud Platform with the same operational model, the same architecture (active-active with MaxScale), and the same control plane experience. A customer can deploy production environments on one cloud and disaster recovery replicas on another, can move

workloads between clouds in response to regulatory or commercial changes, and can avoid the kind of single-vendor concentration risk that becomes more visible during commercial renewal cycles.

What multi-cloud portability actually delivers. Multi-cloud is one of the most over-promised concepts in enterprise software. The variant that MariaDB Cloud delivers is concrete and bounded – three specific capabilities that are each economically and operationally meaningful, rather than a vague promise of "runs anywhere":

Cross-cloud disaster recovery architectures. A customer can deploy production MariaDB Cloud on (for example) AWS in their primary region, and a disaster recovery replica on Azure in a secondary region. The replication, monitoring, and failover orchestration is handled by the same MariaDB Cloud control plane as if both deployments were on the same cloud. In the event of an AWS-region-wide failure, traffic can fail over to the Azure replica with the same architectural model that intra-cloud failover uses.

The economic value of cross-cloud DR depends on the customer's risk tolerance and regulatory requirements. For customers in highly regulated industries (financial services, healthcare, public sector) or with explicit regulatory cross-cloud DR mandates (some EU regulators, certain APAC jurisdictions), the value is direct compliance value. For other customers, the value is in the resilience to single-cloud-provider failure modes – region-wide outages, control-plane incidents, or commercial disputes that affect service availability – that have no good answer within a single-cloud strategy.

Hyperscaler negotiation leverage. Customer renewal negotiations with hyperscalers are structurally asymmetric. The hyperscaler knows what the customer's workloads cost to migrate elsewhere – typically a multi-quarter project at meaningful cost – and prices renewal terms accordingly. A customer whose database tier is portable across clouds has measurably more leverage in those negotiations: the implicit cost of relocating workloads is lower, and the credibility of the threat to relocate is higher.

This is not a hypothetical effect. Customers with credible multi-cloud capability report renewal terms that are 5–15% better than otherwise-similar customers without that capability, even when no actual relocation occurs. The mere capability to relocate, in negotiations with sophisticated counterparties, has commercial value.

Regional and regulatory data residency flexibility. Some customer geographies and customer industries impose data residency constraints that are tighter than the regional footprint of any single hyperscaler. The clearest example is sovereign cloud requirements in some EU member states, but similar constraints exist in regulated industries in APAC, the Middle East, and parts of Latin America. A customer whose database runtime is portable across hyperscalers can place workloads in the region or sovereign cloud that satisfies the residency requirement, rather than choosing between non-compliance and a large lift to relocate workloads.

Why this is structural, not commercial. The multi-cloud capability described above is not the result of MariaDB Cloud running parallel implementations on each hyperscaler. It is a consequence of MariaDB Cloud being built on top of MariaDB Enterprise Server – the same engine on every cloud – wrapped in the same MaxScale routing layer and the same control plane. The architecture is identical across clouds because the underlying open-source engine is identical.

AWS RDS for MariaDB cannot deliver multi-cloud portability because it is, by service definition, an AWS service. AWS could in theory build a multi-cloud version of RDS – but that would be a strategic decision contrary to AWS's product positioning, not a roadmap item. A customer evaluating database platforms in 2026 should treat

AWS RDS's single-cloud constraint as a permanent property of the service, not a temporary limitation.

The strategic compounding effect. The three capabilities described above compound over time in a specific way. Each year a customer continues on a single-cloud database tier, the migration cost of relocating that tier elsewhere grows: more data to move, more dependencies to refactor, more institutional knowledge embedded in the cloud-specific tooling. After three to five years on AWS RDS, the realistic option to move the database tier to another cloud has effectively expired – the migration cost exceeds the strategic value of the optionality.

Each year a customer runs MariaDB Cloud, the optionality is preserved. Cross-cloud DR architectures can be added later. Workloads can be moved later. The negotiation leverage is real on each renewal cycle, not just the first one. The strategic value of preserved optionality is, in present-value terms, meaningfully positive over a three-to-five-year horizon.

Pillar 4: Security posture via BYOA

AWS BYOA – General availability: June 30, 2026. MariaDB Cloud's Bring-Your-Own-Account (BYOA) deployment model is generally available on AWS as of June 30, 2026. Azure GA preceded this date; Google Cloud Platform support follows. This section is written in present tense, reflecting the GA capability as of the document publication date. The GA timing is the only piece of this section that ages.

The fourth pillar of the case for migration is security posture, and it depends on a relatively new capability in the MariaDB Cloud service – the BYOA deployment model – that did not exist at the time prior versions of the cost analysis were published. This section develops the architectural difference between AWS RDS and MariaDB Cloud BYOA, the security implications for the customer's data perimeter, and the trade-offs that an honest treatment of the model requires.

Two architectural models, side by side. The diagram below presents the two deployment models at peer depth, with equal visual weight. The left panel shows the AWS RDS architecture: the customer connects to an Elastic Network Interface (ENI) in their own VPC, but the actual database host runs in an AWS-managed account that the customer cannot see, audit, or apply their security tooling to. The right panel shows the MariaDB Cloud BYOA architecture: the database compute and storage run in the customer's own AWS account, with the MariaDB control plane orchestrating lifecycle operations via scoped IAM roles and short-lived credentials, never accessing customer data.

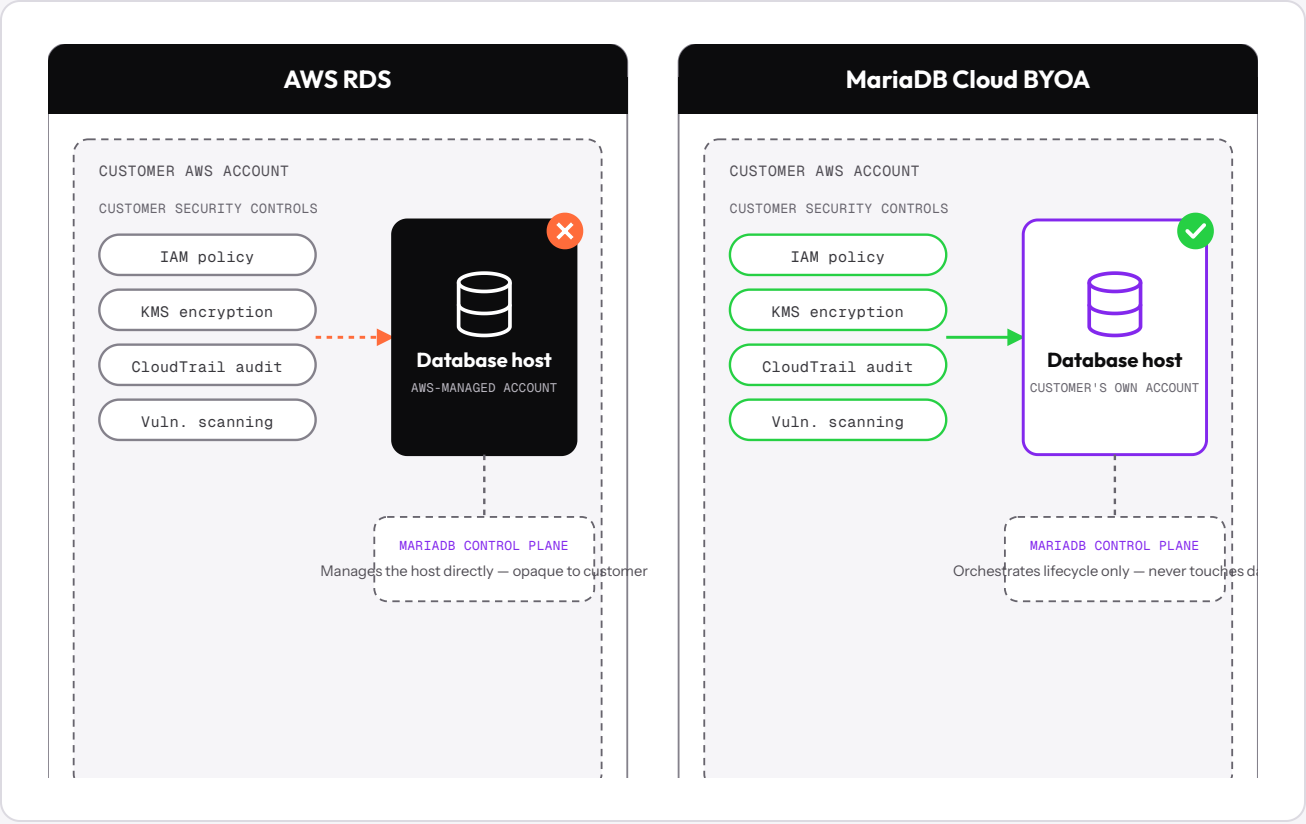


FIGURE 2 Architectural comparison of the customer security perimeter in AWS RDS vs. MariaDB Cloud BYOA on AWS.

The AWS RDS architecture. AWS RDS for MariaDB uses what is sometimes called the "ENI projection" architectural model. From a networking perspective, the database appears to live in the customer's VPC: the customer chooses subnets, security groups apply, the database has a private IP address from the customer's CIDR range, and clients in the customer's VPC connect to it over private networking. This is what most customers see when they look at the deployment in the AWS console.

The architectural reality is different. The Elastic Network Interface that has the customer's IP address is, in fact, only a network interface – there is no actual compute running in the customer's account. The ENI is a projection of network connectivity into the customer's VPC; the host that runs the actual MariaDB engine and stores the actual data is an EC2 instance running in an AWS-managed account that is not visible to the customer. The customer cannot see the host in their EC2 console, SSH into it, collect host-level logs, or install configuration management or vulnerability scanning agents on it. AWS has full operational access; the customer has none.

This architectural model is operationally elegant for AWS – it allows them to manage the underlying compute as a service while maintaining clean tenant boundaries. But it has security and compliance implications for the customer:

- The customer's IAM policies do not extend to the database compute. The customer cannot, for example, restrict which AWS principals can take actions on the host, because they cannot reference the host.
- The customer's audit and compliance tooling does not extend to the database compute. CloudTrail captures API calls against the RDS service, but does not capture host-level activity. Configuration management

tooling does not see the host.

- The customer's vulnerability scanning does not reach the host. Whatever scanning AWS performs on the host is opaque to the customer; the customer cannot validate that any specific scanning regime is in place.
- The customer's encryption key management is constrained. The customer can use a Customer-Managed Key (CMK) for storage encryption, but the actual encryption operations occur on a host the customer does not control. The KMS audit trail shows that the key was used; the trail does not show by what principal on what host.
- The customer's logging pipeline does not include the host. Database engine logs are surfaced to CloudWatch as configured, but host-level logs (operating system audit, file system audit, network audit) are not exposed to the customer at all.

None of this is a defect in AWS RDS. It is the architectural consequence of the service's design. A customer evaluating AWS RDS should be aware that they are accepting these constraints as a property of the service, not as a temporary limitation that AWS will remove.

The MariaDB Cloud BYOA architecture. BYOA inverts the model. The database compute and storage run in the customer's own AWS account, in the customer's VPC, with the customer's IAM policies, the customer's KMS keys, the customer's CloudTrail and audit pipelines, and the customer's configuration management and vulnerability scanning tooling all extending to the database tier exactly as they would to any other workload running in the customer's account. The MariaDB Cloud control plane reaches into the customer's account via a scoped IAM role with short-lived credentials, performs the lifecycle operations needed to manage the database (provisioning, patching, scaling, backup, monitoring, failover orchestration), and never accesses the customer's data.

The data perimeter implications are direct and important:

- **Data physically resides in the customer's account.** Database compute and storage are billable line items on the customer's AWS bill, visible in the customer's AWS console, and subject to the customer's account-level controls.
- **Customer IAM policies extend to the database tier.** The customer can restrict which AWS principals (users, roles, services) can perform which actions on the database hosts. The MariaDB Cloud control plane operates with a scoped IAM role that is documented and that the customer can audit and adjust within the bounds required for the service to function.
- **Customer KMS keys protect the data, with full lifecycle control.** Encryption keys are owned by the customer's account, can be rotated, audited, and revoked by the customer's processes, and are used by encryption operations on hosts that the customer's IAM controls. Key compromise scenarios are entirely within the customer's blast radius.
- **Customer audit and compliance tooling reaches the database tier.** CloudTrail captures all API activity against the database hosts. Configuration management tooling can enforce baseline configuration on the hosts. Vulnerability scanning, AWS Inspector, GuardDuty, Security Hub – all of these extend to the database tier as they would to any other workload.
- **Customer logging pipelines include host-level activity.** Operating system audit logs, file system audit, network flow logs – all are produced on hosts in the customer's account and flow to the customer's logging

infrastructure on whatever cadence the customer configures.

Why this matters for compliance and security posture. For a customer in a regulated industry, the architectural difference between AWS RDS and BYOA is not a nuance – it is the difference between "we can demonstrate to our auditor that the database tier is governed by our security controls" and "we depend on AWS's controls, which we cannot directly audit."

Specific scenarios where this difference is material:

- **Data sovereignty under regulatory regimes.** GDPR Article 32 and similar regulations in other jurisdictions require demonstrable customer control over data processing, including infrastructure-level controls. AWS RDS's architecture places the data on infrastructure outside the customer's direct control; BYOA places it inside. The compliance posture difference is direct.
- **Industry-specific regulatory frameworks.** Financial services regulators (OCC in the U.S., PRA/FCA in the U.K., BaFin in Germany, MAS in Singapore, and others) have evolving guidance on cloud-resident data, and the trend is toward expecting customer-controllable infrastructure at the data tier. BYOA satisfies this expectation in a way that AWS RDS structurally does not.
- **Customer-controlled forensic response.** In an incident response scenario where the customer's security team needs to investigate the database tier, BYOA gives the team direct access to host-level evidence: file system snapshots, memory captures, network flow logs, OS audit. AWS RDS gives the team only what AWS chooses to expose through service APIs.
- **Customer-controlled key management.** Some customers operate under key management requirements that AWS-managed encryption cannot satisfy – for example, key escrow requirements, hardware security module (HSM) integration with customer-controlled HSMs, or specific cryptographic algorithm choices. BYOA puts the customer in control of these decisions.

Honest treatment of trade-offs. BYOA is not a free architectural upgrade – it shifts the shared-responsibility boundary in ways that the customer's operations team needs to understand and explicitly accept. Two trade-offs are worth flagging:

Shared responsibility model becomes more complex. With AWS RDS, AWS owns essentially everything below the SQL layer – the operating system, the database engine, the underlying compute, the underlying storage. The customer's responsibility boundary is clean: schema, queries, application-side connection management, and any data processed through the database. With BYOA, the customer owns the underlying compute and storage (it is in their account), MariaDB owns the database engine and operational management, and the responsibility boundaries are less crisp. Incidents that span the boundary ("Is this an infrastructure issue or a database issue?") need clearer runbooks than RDS-style operations require.

Customers adopting BYOA should explicitly assign ownership of the gray areas: who handles host-level vulnerability remediation if the customer's scanning tool flags an issue? Who handles AWS-side incidents that affect the database tier (e.g., AWS region issues, IAM policy changes that affect the control plane)? These questions have clear answers, but the answers need to be agreed upfront rather than discovered during an incident.

The control plane has privileged access into the customer account. BYOA is not "zero-vendor-access" architecture. The MariaDB Cloud control plane requires a scoped IAM role in the customer's account in order to provision, patch, scale, back up, and orchestrate the database tier. The role is scoped to the minimum permissions required for these operations, the credentials are short-lived (minutes to hours, not days), and the role's actions are captured in the customer's CloudTrail. But the role exists, and customers should review it carefully against their security policy.

The right comparison is not "BYOA is zero-trust" versus "AWS RDS requires trust." Both deployment models require trust in a vendor that has privileged access to the database tier. The difference is that under BYOA, the privileged access is a documented IAM role that the customer can audit and adjust, in their own account, with credential rotation under their control. Under AWS RDS, the privileged access is whatever AWS chooses to grant itself in the AWS-managed account, with no customer visibility or control. Different trust models, not "trust" versus "no trust."

BYOA collapses a structural compliance and security gap that AWS RDS cannot close. For a customer with sophisticated security controls, regulated data, customer-controlled key management requirements, or compliance frameworks that expect customer-auditable infrastructure at the data tier, BYOA delivers a security posture that AWS RDS cannot match by architectural design. The shared-responsibility model is more complex, and the control-plane access requires a scoped trust relationship – but these are manageable trade-offs, while the AWS RDS architectural constraint is permanent.

04

Conclusion

This paper has made the case for migration from AWS RDS for MariaDB to MariaDB Cloud across two dimensions: total cost of ownership and application experience and security. The combined argument is the one that matters for the buying committee.

The TCO case

On a like-for-like fleet migration with matched active vCPU sizing and list prices on both sides, the RetailCo illustrative case study – a hypothetical $10 billion retail e-commerce operator running a 60-environment MariaDB fleet – saves approximately $1.36 million per year (47%), with a payback period of under one month on a $100,000 typical migration cost. Three-year cumulative net savings are approximately $3.98 million.

The savings come from five compounding categories: auto-scaling that eliminates 24×7 peak provisioning (modeled at peak load = 2× normal), active-active architecture that eliminates the idle Multi-AZ standby, bundled platform components (proxy, observability, monitoring) that AWS bills separately, avoided forced major-version upgrade events, and analytics pipeline collapse where the customer can move from a multi-service Redshift pipeline to MariaDB Exa.

The figures are conservative. List prices on both sides, DBA labor savings excluded entirely, strategic optionality unmodeled, and BYOA cost capture omitted from the headline numbers – for an AWS-committed customer adopting BYOA, the realized saving is materially larger than the figures shown.

The application experience and security case

Beyond the dollar saving, MariaDB Cloud delivers four structural advantages over AWS RDS for MariaDB that compound over time:

- **Zero-downtime failover.** MaxScale-mediated active-active failover completes in under one second, compared to AWS Multi-AZ failover at 90–120 seconds with application-side disruption. For a retail customer at peak hours, the difference is real revenue.
- **White-glove dedicated support.** MariaDB engineers who built the engine, with proactive performance review, capacity planning, and architectural guidance – operationally distinct from AWS Enterprise Support's database-agnostic generalist model.
- **Multi-cloud portability.** Native deployment on AWS, Azure, and GCP. Cross-cloud disaster recovery, hyperscaler negotiation leverage, and regional data residency flexibility – none of which AWS RDS can deliver by definition.
- **Security posture via BYOA on AWS.** Database compute and storage in the customer's own AWS account, governed by the customer's IAM, KMS, and audit pipelines. Closes a structural compliance gap that AWS RDS cannot close by architectural design.

The combined argument

For a customer evaluating their cloud database posture in 2026, the case for migration is more than a cost story. The cost advantage is real, and at fleet scale it is large. But the cost advantage exists alongside an availability advantage, an operational support advantage, a strategic-optionality advantage, and a security-posture advantage – each of which compounds with environment count, with regulatory scope, and with time. The decision to remain on AWS RDS for MariaDB is, in 2026, a decision to absorb each of these gaps as a permanent property of the deployment.

Next step: TCO and architecture validation workshop

MariaDB offers a 90-minute structured workshop in which the figures presented in this paper are replaced with the customer's actual situation: actual environment count, actual instance sizing, actual observability spend, actual workload duty cycle, and actual AWS commercial position. The workshop produces a customer-specific cost stack, a workload sizing recommendation, and a like-for-like migration plan outline.

The workshop is designed for a finance lead, an engineering lead, and a security lead from the customer side, plus a MariaDB solutions engineer. Output is a customer-specific TCO model the customer can take to their internal procurement and budget process. There is no commercial commitment associated with the workshop.

Schedule your TCO and architecture validation workshop

mariadb.com/contact/

05

Appendix A: Detailed cost stack derivation

This appendix walks through the derivation of each line in the cost stack presented. Readers running their own analysis with their own inputs can use this as a template to substitute their actual observed numbers.

Compute. AWS RDS for MariaDB pricing in US East (N. Virginia) at publication date, applied to the RetailCo fleet:

- Production: 15 × db.r6g.8xlarge Multi-AZ at $5.96/hour × 8,760 hours = $782,866 annualized
- UAT: 15 × db.r6g.4xlarge Multi-AZ at $2.98/hour × 8,760 hours = $391,433 annualized
- Development: 30 × db.t4g.large Single-AZ at $0.115/hour × 8,760 hours = $30,222 annualized (continued business-hours model in headline)
- Production peak-provisioning premium captured separately in auto-scaling line item
- Total compute on AWS: ~$1,455,375

MariaDB Cloud Foundation Tier pricing with active-active scale-fabric surcharge:

- Production: 15 × 2 × Sky-16×128 active-active at matched active vCPU = $1,070,000 annualized at peak; auto-scaled below
- UAT: 15 × 2 × Sky-8×64 active-active at matched active vCPU = $267,000 annualized at peak; auto-scaled
- Development: 30 × Sky-2×4 single-node = $33,420 annualized
- Total compute on MariaDB Cloud at peak provisioning: $1,203,420 (before auto-scaling adjustment)

Storage. Both AWS RDS gp3 Multi-AZ and MariaDB Cloud bundled storage are priced at $0.23 per GB per month at publication date. RetailCo storage volume:

- Production: 15 × 2 TB = 30 TB
- UAT: 15 × 1 TB = 15 TB
- Development: 30 × 200 GB = 6 TB
- Total: 51 TB → $0.23/GB-month × 51,000 GB × 12 = $140,760 (excluding analytics/backup)

Storage is approximately a wash on both sides at list prices. The small delta visible in the cost stack reflects bundled-storage replication economics on the MariaDB Cloud side.

Connection proxy. AWS RDS Proxy is priced at $0.015 per vCPU-hour against the active vCPU count. MariaDB Cloud's MaxScale proxy is included in the scale-fabric surcharge.

- Active vCPU at peak across production and UAT: 15 × 32 + 15 × 16 = 720 vCPU
- RDS Proxy: 720 vCPU × $0.015/hour × 8,760 hours = $94,608 annualized

- MaxScale on MariaDB Cloud: included

Observability. AWS CloudWatch and Database Insights pricing across the fleet – derived from typical observed spend per environment, not from raw vendor unit pricing. The figure below represents the observed pattern for fleet-scale RDS deployments where customers are using the observability stack to a level appropriate for production-grade operations:

- CloudWatch metrics, custom metrics, log ingestion, log storage, Database Insights advanced retention, observability data egress to third-party tools
- Typical observed spend: ~$2,500 per production environment per year, ~$1,200 per UAT environment per year, ~$300 per development environment per year
- Across the RetailCo fleet: ~$155,550 annualized
- MariaDB Cloud: bundled, $0 incremental

Customers running their own analysis should substitute their actual observed observability spend per environment from their AWS bill. The figure is highly sensitive to how aggressively the customer uses CloudWatch custom metrics and how much observability egress they have to external tools.

Major-version upgrade event. MariaDB Community Server has a 3-year LTS support window. AWS RDS forces a major-version upgrade approximately every three years across all environments running on the deprecating version. The upgrade event consists of: regression testing across all applications, parallel-running pre-production environments on the new version, coordinating maintenance windows, troubleshooting application-side compatibility issues, and absorbing operational risk during the cutover. Typical fleet-wide cost for a 60-environment fleet:

- Application engineering effort for compatibility testing and remediation: ~$200,000 amortized
- DBA and SRE effort for upgrade execution: ~$150,000 amortized
- Operational risk cushion (allowance for incidents during cutover): ~$100,000 amortized
- Total: ~$450,000 every 3 years, annualized at $150,000

MariaDB Enterprise Server provides 8 years of supported life (5 standard + 3 extended), eliminating the forced upgrade event for the RetailCo TCO horizon. The avoided cost is reflected as a $150,000 annualized saving.

Auto-scaling adjustment. The RetailCo workload duty cycle is modeled with a single simple assumption: peak load is approximately 2× normal load. AWS RDS for MariaDB cannot vertically auto-scale compute without downtime, so customers peak-provision 24×7 – paying for 2× the average required capacity continuously. MariaDB Cloud auto-scales without downtime, billing for actual size minute-by-minute.

Effective compute saving on auto-scaled environments:

- Provisioned capacity (peak): 2× normal load
- Average actual usage on a 24×7 basis: 1× normal load (averaged across peak and off-peak periods)
- Effective utilization on AWS (peak-provisioned 24×7): ~50%
- Effective utilization on MariaDB Cloud (auto-scaled): ~100% (pays only for actual usage)

- Auto-scaling saving on production+UAT compute: 50% × $1,174,300 = $587,150 annualized
- Auto-scaling not applied to development (sized as fixed single-node both sides)

Analytics pipeline. The RetailCo configuration includes a consolidated analytics pipeline. On AWS, this is the typical RDS → DMS → S3 → Redshift pattern: change data capture replication into S3 staging, then loading into Redshift for business intelligence. On MariaDB Cloud, the pattern collapses into MariaDB Exa fed directly from production via internal CDC.

Annualized cost on AWS:

- Redshift cluster: ~$120,000 annualized
- DMS replication tasks: ~$30,000 annualized
- S3 staging storage and transfer: ~$10,000 annualized
- Pipeline orchestration (Airflow, Glue, monitoring): ~$40,000 annualized
- Total: ~$200,000 annualized

Annualized cost on MariaDB Cloud:

- MariaDB Exa, 32 vCPU + CDC: ~$109,100 annualized
- Saving: ~$90,900 annualized

Note: Analytics workload performance and feature parity should be validated against the customer's specific BI requirements during the 90-minute workshop. Exa's columnar storage and HTAP design are well-matched to retail analytics workloads but may require workload-specific validation.

Standard support. AWS Enterprise Support is priced at 3% to 10% of monthly AWS spend depending on tier and minimum commitments. The cost stack uses a 10% factor against the AWS RDS spend, consistent with typical Enterprise tier pricing. MariaDB Cloud standard support is bundled into the service price; the implicit support component is calculated as a similar percentage of the underlying compute and storage.

- AWS RDS Enterprise Support: 10% × $2,175,123 (compute + storage + proxy + observability + analytics) = $217,512 annualized
- MariaDB Cloud bundled support equivalent: ~10% × $1,832,720 = $183,272 annualized
- Net support delta: $34,240 annualized

06

Appendix B: Glossary

Definitions of terms used in this paper. Industry-standard terms are defined briefly; MariaDB-specific terms and capabilities are defined in more detail.

Active-active topology — A database replication architecture in which two (or more) database nodes are simultaneously active and serve client traffic. Distinct from active-passive (also called "primary-standby") topology, in which only one node serves traffic at any time and the other is reserved for failover.

BYOA – Bring Your Own Account — A deployment model in which a managed service vendor (in this paper, MariaDB Cloud) deploys their service into the customer's own cloud account rather than into a vendor-managed account. The vendor manages the service via scoped IAM roles into the customer account; the data and compute reside in the customer's account. AWS BYOA is generally available as of June 30, 2026; Azure BYOA preceded; GCP follows.

CDC – Change Data Capture — A technique for replicating changes from a source database to a target database or analytics platform by capturing the source's transaction log and applying the captured changes to the target. Used by MariaDB Cloud to feed MariaDB Exa from a production transactional database without intermediate ETL infrastructure.

CMK – Customer-Managed Key — An encryption key stored and managed in AWS Key Management Service (KMS) under the customer's account, controllable by the customer's IAM policies. The customer can rotate, audit, and revoke the key. AWS RDS supports CMK encryption at rest, but the encryption operations themselves occur on AWS-managed hosts that the customer cannot inspect.

ENI – Elastic Network Interface — A virtual network interface in AWS VPC. The ENI has a private IP address from the customer's subnet CIDR and is the addressable endpoint for clients in the customer's VPC. AWS RDS uses an ENI projection model in which the ENI is in the customer's VPC but the underlying EC2 host is in an AWS-managed account.

Exa – MariaDB Exa — MariaDB's managed columnar analytics product, designed for HTAP (hybrid transactional and analytical processing) workloads. Fed from production transactional databases via internal CDC, eliminating the need for an external ETL pipeline (DMS → S3 → Redshift in the typical AWS pattern).

Fractional DBA service — MariaDB Cloud's recurring engagement model in which a named MariaDB engineer dedicates a defined fraction of their time to the customer's MariaDB operations on an ongoing basis. Distinct from reactive support (per-ticket) and from full-time hired DBAs. Engagement scope typically includes performance review, capacity planning, migration support, and architectural guidance.

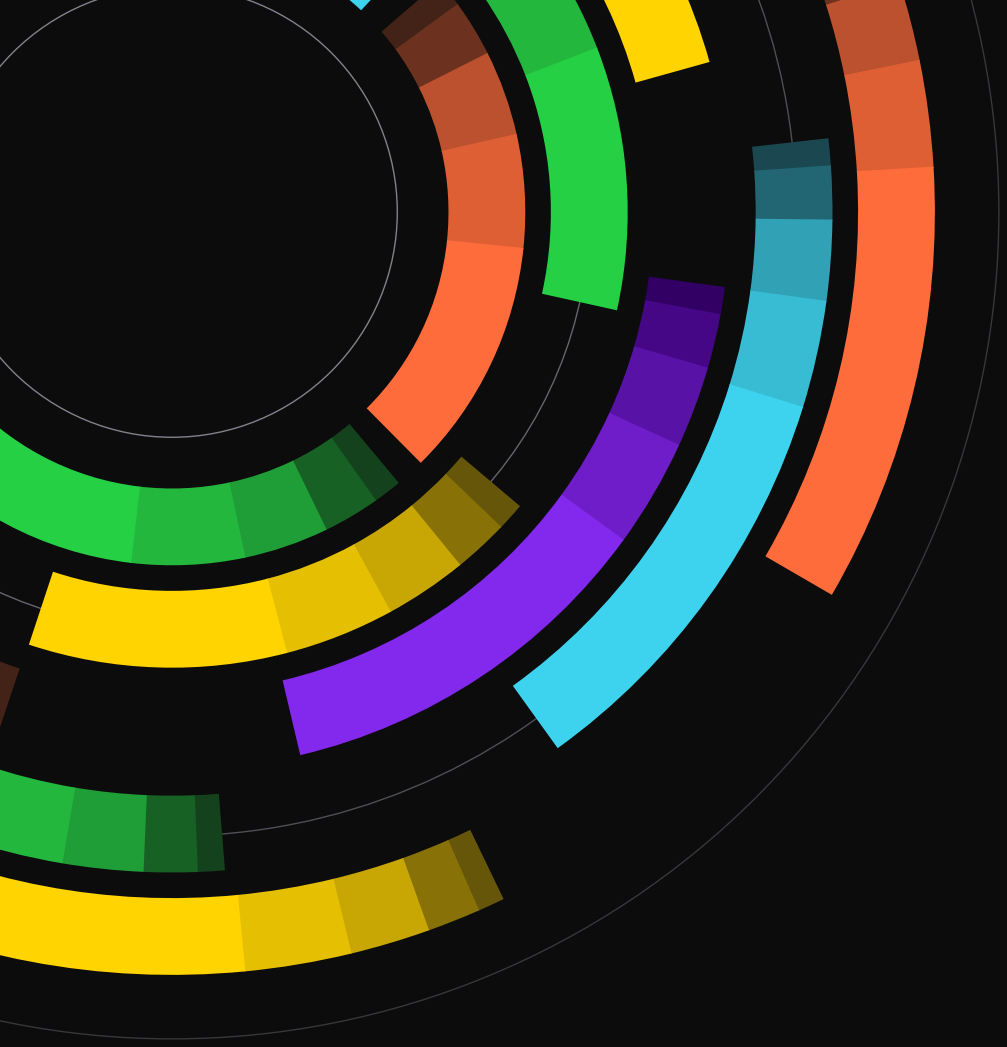
HTAP – Hybrid Transactional and Analytical Processing — A database architecture that supports both transactional (OLTP) and analytical (OLAP) workloads on the same data without requiring an external ETL pipeline. MariaDB Exa is HTAP.

MaxScale — MariaDB's intelligent database proxy. In MariaDB Cloud's reference architecture, MaxScale terminates client connections, performs read/write splitting, runs continuous health checks against the database nodes, and orchestrates failover by removing failed nodes from the routing rotation. MaxScale is the architectural mechanism that delivers sub-second failover.

Multi-AZ — AWS RDS deployment configuration in which the database has a primary instance in one Availability Zone and a synchronous standby in a second Availability Zone. The standby does not serve read traffic in standard Multi-AZ; AWS Multi-AZ Cluster (a separate product) provides reader endpoints with different pricing and architectural characteristics.

Scale-fabric surcharge — MariaDB Cloud's pricing surcharge for active-active configurations. Applied as a percentage uplift on Foundation Tier pricing. Bundles the active-active replication, MaxScale proxy, observability, and managed control plane.

Sky shape — MariaDB Cloud's instance sizing nomenclature. "Sky-N×M" denotes an instance with N vCPUs and M GiB of memory. For example, Sky-16×128 is a 16-vCPU, 128 GiB instance.



mariaDB

Schedule your TCO and architecture validation workshop

mariadb.com/contact/