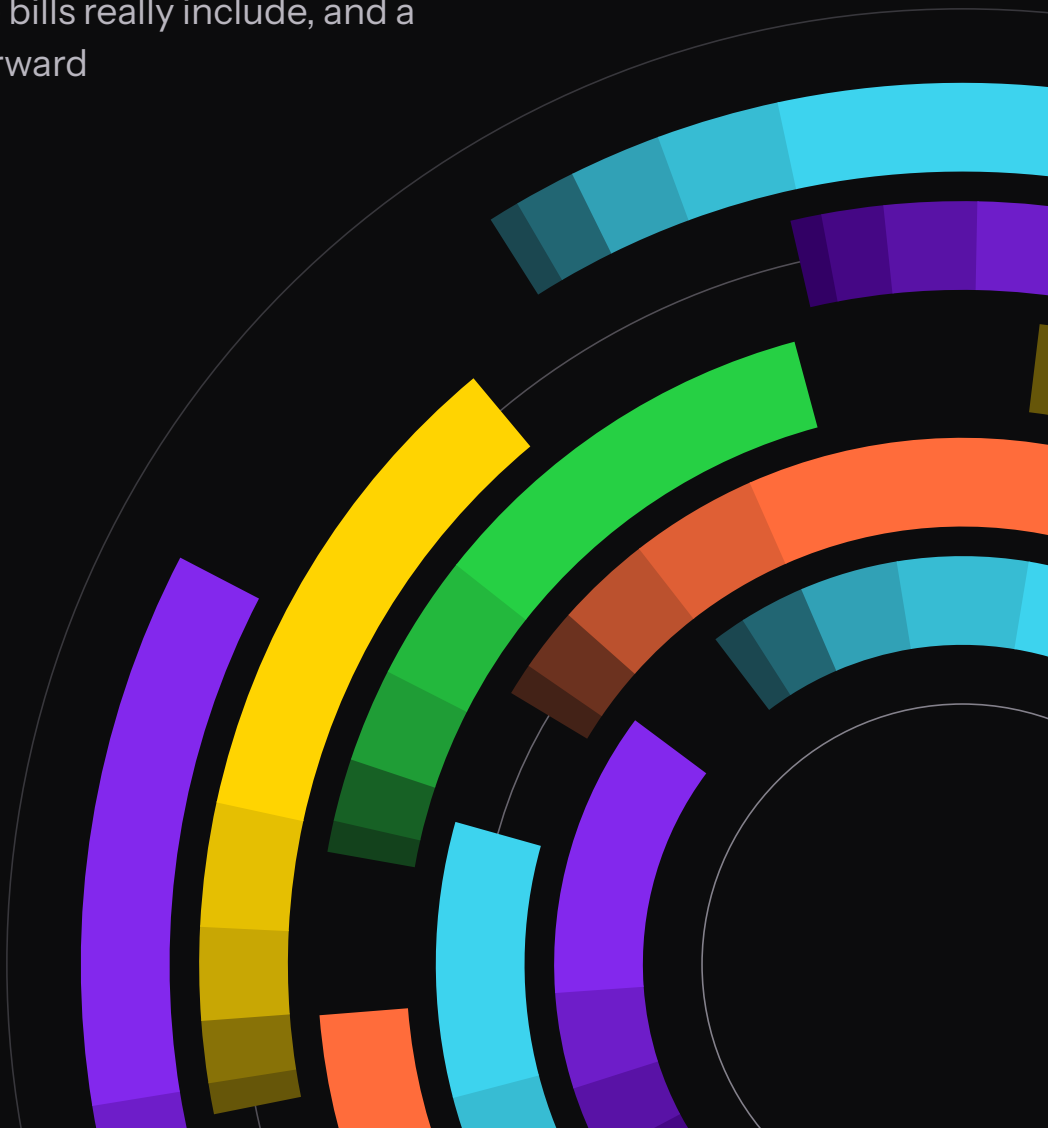
mariaDB

WHITE PAPER

The hidden cost of Amazon RDS

What managed database bills really include, and a more predictable path forward



CONTENTS

Table of contents

01 **Introduction to cloud database economics** 3

02 **What is RDS** 5

03 **Why choose RDS** 7

04 **The visible costs** 9

- Instance compute pricing 10
- Storage provisioning 10
- Data transfer egress and ingress 11

05 **The hidden costs** 12

- RDS Proxy and connection management 12
- Observability: CloudWatch Database Insights and Performance Insights 13
- Backup storage accumulation 13
- Snapshot export and S3 integration 14
- Extended Support pricing crisis 14
- The hidden networking tax: NAT Gateways and PrivateLink 15
- OLAP support via Redshift and Zero-ETL 16
- Vector search, RAG, and AI workloads 17
- The idle standby penalty and active-active complexities 18
- The absence of native compute autoscaling 18
- The hidden human cost: database administrators 18

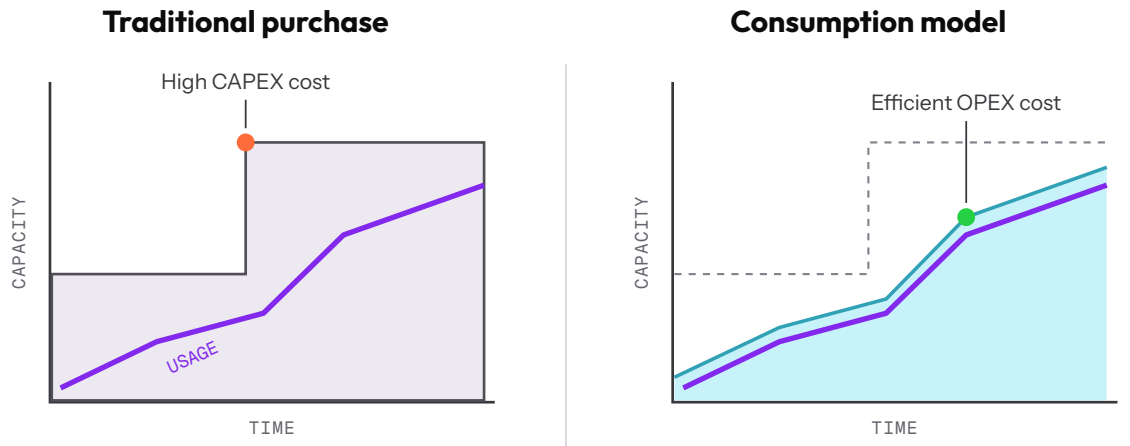
06 **A predictable alternative: MariaDB Cloud** 19

- Unified workload architecture and HTAP 19
- Native vector search and agentic AI integration 20
- Proactive support and Managed Operations 20
- Transparent and predictable economics 21

Sources and references 22

01

Introduction to cloud database economics



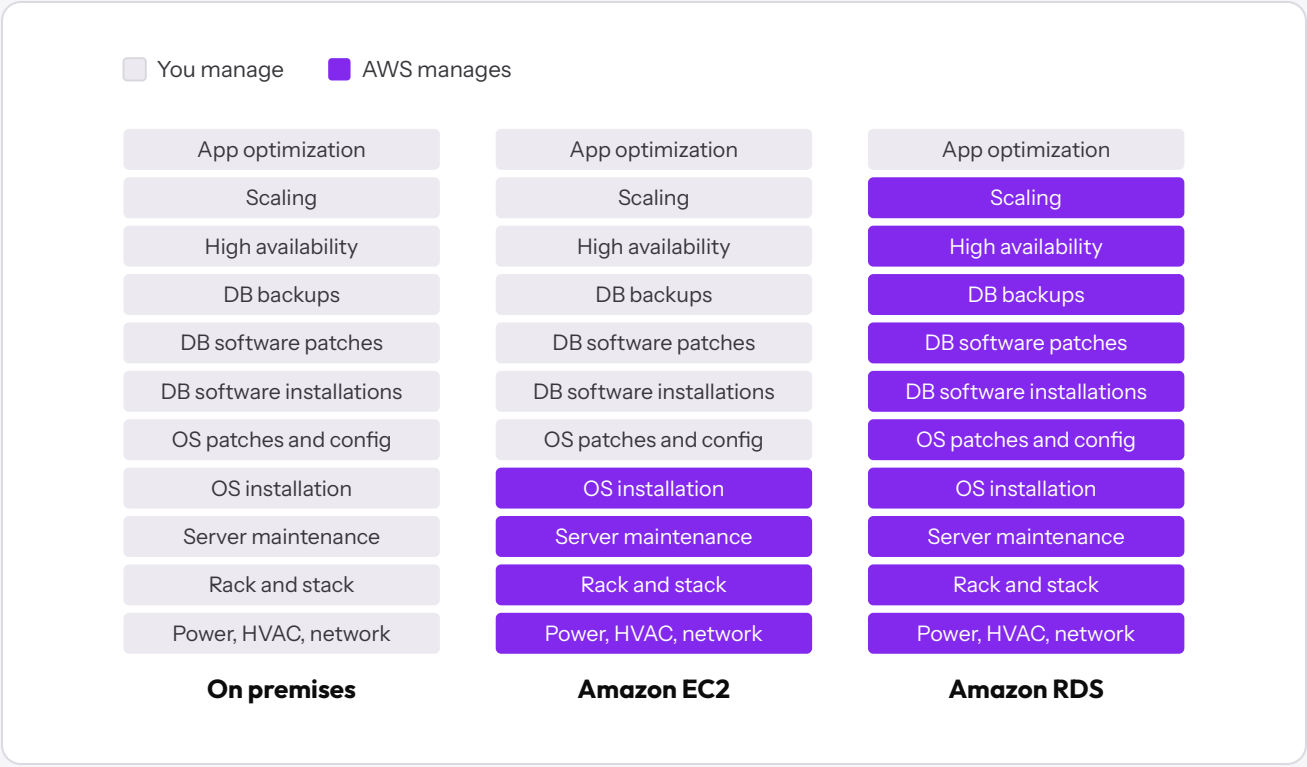
The fundamental transition from capital expenditure models to operational expenditure models has systematically restructured enterprise information technology budgets over the past decade. Within the specific domain of relational database management, this paradigm shift was heralded as a mechanism to liberate engineering teams from the constraints of hardware provisioning, manual software patching, and the implementation of complex disaster recovery configurations. Amazon Relational Database Service (RDS) emerged as a foundational cornerstone of this modern infrastructure paradigm, offering a fully managed, scalable environment for deploying, operating and tuning relational databases in the cloud. However, the operational abstraction provided by such managed services introduces a secondary, highly complex layer of financial obligation.

The billing architecture of contemporary cloud environments has evolved into a highly granular, metered framework where virtually every discrete operation, network hop, computational cycle, and storage transaction is individually quantified and subsequently monetized.[1] This layered approach to resource metering creates thousands of potential billing dimensions that extend far beyond the foundational compute and storage costs initially modeled by procurement teams.[1] As cloud ecosystems mature and application architectures become increasingly distributed, the initial allure of simplified database management is frequently counterbalanced by unpredictable, opaque, and rapidly escalating financial obligations.

A forensic examination of these billing mechanics reveals a substantial and consistent divergence between the advertised base costs of managed database services and the realized total cost of ownership in production environments. Organizations frequently discover that cloud bills in 2026 exceed initial forecasts by substantial margins, driven not by raw storage volume growth or core compute requirements, but by data access charges, integration penalties, and auxiliary observability tools.[1] This comprehensive analysis deconstructs the economic model of Amazon RDS, categorizing both the anticipated visible costs and the insidious hidden fees that accumulate within modern architectures, ultimately exploring how unified database platforms are emerging to restore financial predictability.

02

What is RDS



Amazon Relational Database Service is a fully managed cloud service engineered by Amazon Web Services (AWS) to automate the administration of relational databases. By abstracting the underlying infrastructure layer, Amazon RDS enables organizations to provision highly available database instances without the requisite overhead of managing physical hardware, configuring operating systems, or executing base-level database engine installations. The architectural design of the service operates by providing users with complete access to the data plane - the database engine itself and its associated schemas - while the cloud provider assumes absolute authoritative control over the control plane, which encompasses the automation governing backups, failovers, and security patching.

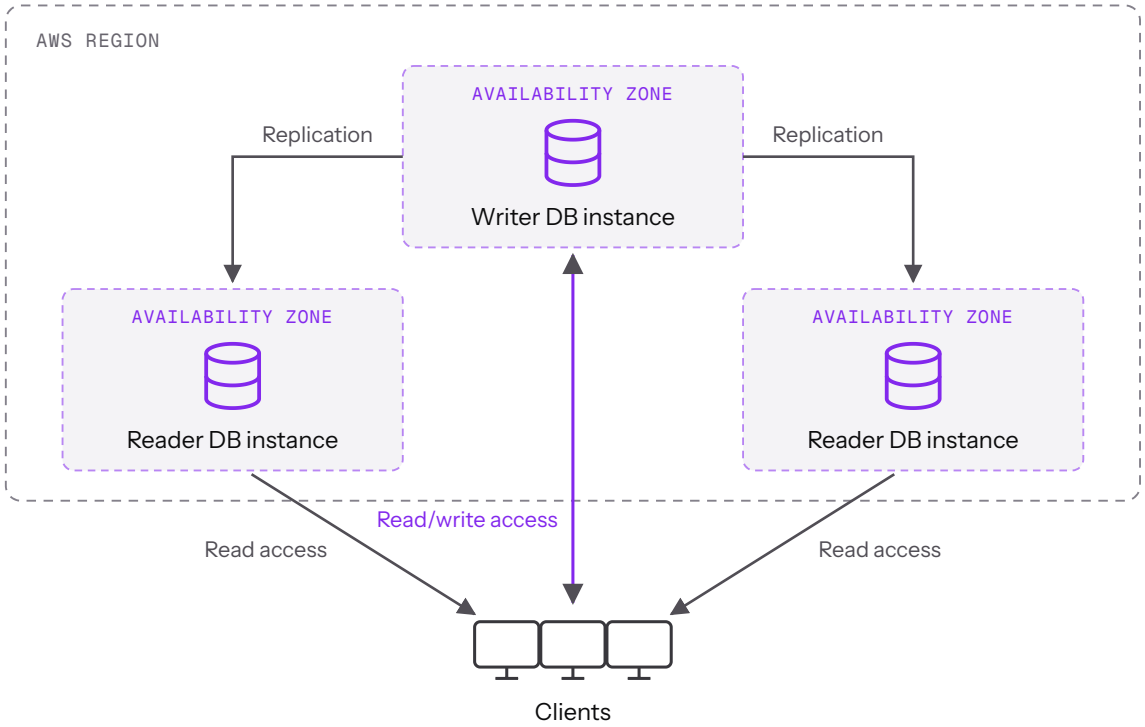
The platform is designed to support a diverse ecosystem of database engines, thereby accommodating various corporate licensing requirements and application compatibility mandates. These supported engines include open-source standards such as MariaDB, MySQL, and PostgreSQL, alongside enterprise commercial offerings including Oracle Database, Microsoft SQL Server, and IBM Db2. Additionally, Amazon provides its own proprietary, cloud-native engine known as Amazon Aurora, which is built for deep MySQL and PostgreSQL compatibility but features a proprietary, distributed, fault-tolerant, and self-healing storage subsystem designed specifically for cloud infrastructure.[2]

The fundamental architectural design of Amazon RDS relies on the strict segregation of compute capacity from persistent storage. This decoupling allows system administrators to scale these two primary resources relatively independently based on shifting application demands. An instance is typically launched within an Amazon Virtual Private Cloud (VPC), which establishes a secure, isolated network perimeter around the database. This managed paradigm effectively transfers the burden of routine operational maintenance from internal database administrators to the cloud provider.

From a licensing and procurement perspective, Amazon RDS offers flexible consumption models. For commercial engines such as Microsoft SQL Server and Oracle, the service provides a "License Included" model, where the cost of the proprietary database software license is seamlessly bundled into the hourly compute rate, eliminating the need for enterprises to procure separate software agreements.[2] Alternatively, organizations possessing existing enterprise agreements can leverage the "Bring Your Own License" model for specific engines to maximize historical software investments.[5] For organizations experimenting with the platform, the AWS Free Tier offers limited usage of specific instance sizes and automated backup storage for a 12-month period, after which the billing transitions to standard, usage-based metering.[2]

03

Why choose RDS



The strategic rationale for adopting Amazon RDS centers on maximizing operational efficiency, mitigating systemic infrastructure risks, and accelerating developer velocity. In a traditional, self-managed database architecture hosted on raw compute instances or on-premises hardware, organizations bear the full, undivided responsibility for engineering high availability, configuring point-in-time recovery, securing the operating system, and manually scaling storage volumes. Amazon RDS systematically eliminates these manual imperatives through highly orchestrated automation.

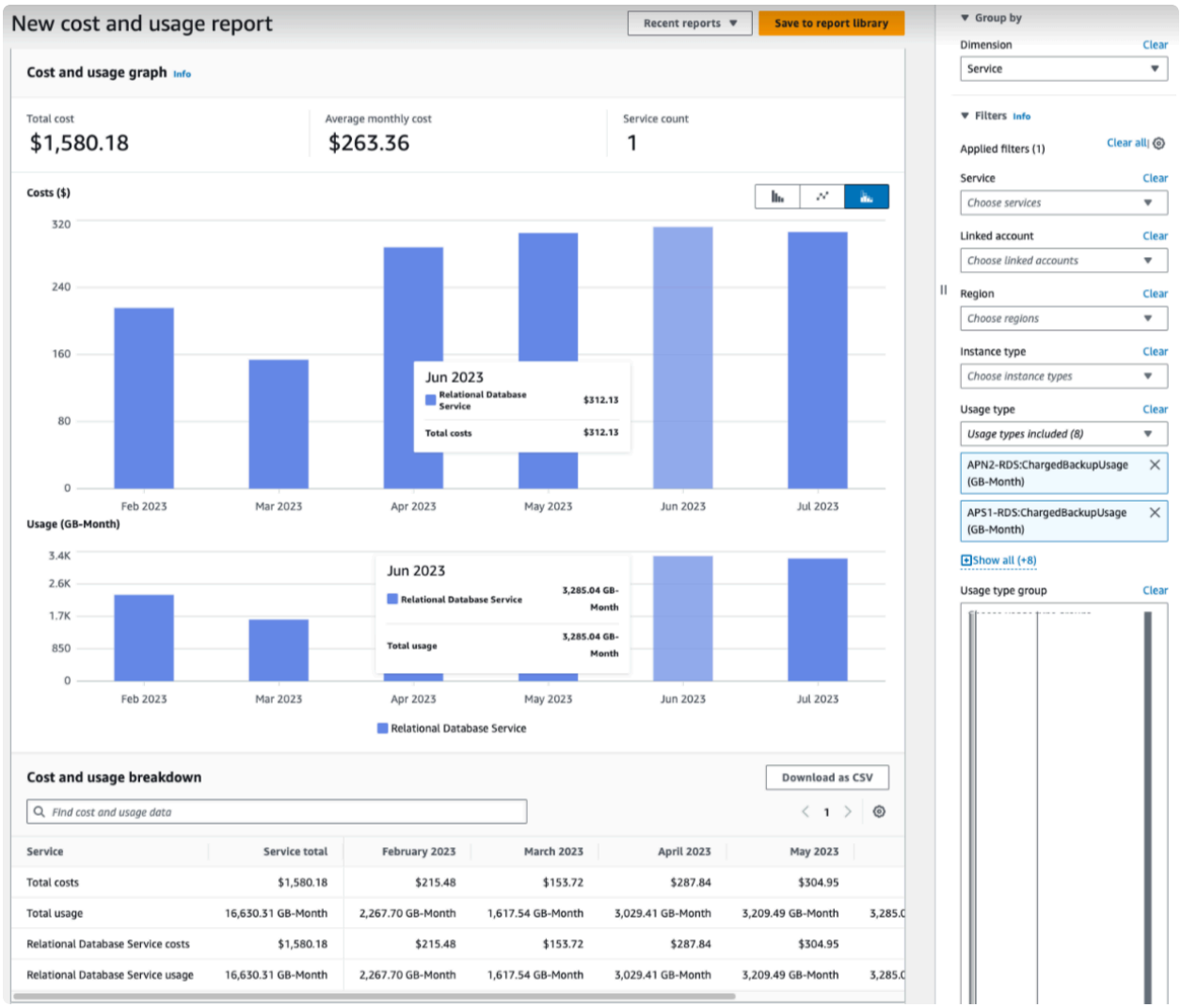
High availability is arguably the most compelling feature driving enterprise adoption. This is achieved through the Multi-AZ deployment option, which automatically provisions and maintains a synchronous standby replica of the database in a geographically distinct Availability Zone within the same AWS Region. In the event of an infrastructure failure, storage degradation, or network isolation affecting the primary node, the control plane detects the anomaly and initiates an automatic failover sequence. This orchestration reroutes application database connections to the standby instance, typically restoring service with minimal interruption and ensuring stringent Recovery Time Objectives (RTOs) and Recovery Point Objectives (RPOs) are met without human intervention.

For applications characterized by heavy read-to-write ratios, the service facilitates the creation of Read Replicas. This feature utilizes asynchronous replication to copy data from the primary node to multiple supplementary instances, allowing architects to offload read-heavy reporting queries and analytical workloads from the primary transactional database. This horizontal scaling capability is critical for consumer-facing applications experiencing sudden spikes in traffic.

Furthermore, automated backup protocols guarantee that transaction logs and storage volume snapshots are continuously archived to highly durable object storage. This mechanism enables database administrators to execute granular point-in-time recovery down to the precise second, safeguarding the enterprise against accidental data deletion or malicious corruption. Routine maintenance operations, including critical operating system security patching and minor database version upgrades, are seamlessly executed within predefined, customer-controlled maintenance windows. This automation ensures compliance with organizational security mandates while minimizing the risk of unpatched vulnerabilities. For engineering teams, the delegation of these tasks translates directly into accelerated deployment cycles. By offloading infrastructure maintenance, technical personnel are free to redirect their focus and resources toward high-value activities such as application development, logical schema optimization, and complex query performance tuning.

04

The visible costs



SOURCE / AWS COST EXPLORER

The foundational billing dimensions of Amazon RDS are generally well-documented and comprise the bulk of initial financial forecasts generated by engineering teams. These primary cost vectors are categorically divided into compute instance provisioning, persistent storage volume allocation, and fundamental data transfer operations.[7]

Instance compute pricing

Compute resources represent the primary and most highly visible financial commitment in any Amazon RDS deployment. Pricing algorithms are dictated by a combination of the selected database engine, the instance family, the instance size, and the architectural deployment model (Single-AZ versus Multi-AZ). Compute instances are billed strictly on a per-hour basis, with fractional hours dynamically billed in one-second increments following a mandatory 10-minute minimum charge for any billable status change.[6]

The selection of the underlying instance node directly dictates the allocation of Central Processing Units (CPUs), Random Access Memory (RAM), and dedicated network bandwidth. Instance families are categorically designed to accommodate varying workload profiles based on their specific resource ratios. The Burstable Performance family, commonly designated as the T-series, is engineered for developmental environments or highly intermittent workloads.[8] These nodes accrue CPU credits during periods of idleness and consume those credits during traffic spikes. If the instance is configured in "unlimited mode" to prevent performance throttling, any sustained CPU utilization that exceeds the baseline credit accumulation will rapidly generate supplementary charges billed per vCPU-hour, introducing a variable cost element to what appears to be a fixed-rate instance.[8]

The General Purpose family, designated as the M-series, provides a balanced, symmetrical ratio of computational power, memory, and network resources. These instances form the backbone of predictable, moderate-intensity enterprise workloads that demand consistent baseline performance without the necessity for extreme memory caching.[8] Conversely, the Memory-Optimized families, encompassing the R-series and X-series, are specifically engineered for complex, memory-intensive queries and large datasets that require substantial in-memory buffer pools.[8] By maximizing the RAM-to-vCPU ratio, these instances prevent the database engine from executing slow, expensive disk-swapping operations, thereby preserving query latency at the expense of a significantly higher hourly billing rate.

The deployment architecture profoundly impacts this compute baseline. Implementing a Multi-AZ configuration effectively doubles the aggregate compute cost, as the organization is financially responsible for both the primary operational instance and the synchronous, fully provisioned standby node.[5] Organizations attempt to mitigate these compounding on-demand compute costs by leveraging commitment-based discount mechanisms. Database Savings Plans and Reserved Instances allow enterprises to exchange capital predictability and rigid, long-term usage commitments (spanning one to three years) for substantially reduced hourly rates, scaling the discount based on whether the commitment is paid upfront, partially upfront, or not at all.[2]

Storage provisioning

With the exception of Amazon Aurora, which utilizes a proprietary, auto-scaling distributed storage cluster, standard Amazon RDS deployments require systems administrators to explicitly provision persistent Elastic Block Store capacity, measured and billed in GiB-months.[5]

The storage tiers are segmented by performance characteristics. General Purpose SSD storage provides a reliable baseline of Input/Output (I/O) performance that is highly suitable for the vast majority of standard transactional workloads. The evolution to the gp3 tier allows administrators to provision IOPS and storage throughput entirely independent of the raw storage capacity, offering a more granular mechanism for financial control compared to legacy storage volumes.[7]

For latency-sensitive, high-throughput applications requiring sustained and guaranteed I/O consistency, organizations must utilize Provisioned IOPS SSD storage. This premium storage tier introduces a dual-vector billing mechanism. The organization incurs one distinct charge for the allocated GiB storage capacity, and a separate, independent charge for the exact mathematical number of IOPS provisioned per month. Crucially, the organization is billed for the provisioned IOPS rate regardless of the actual I/O operations consumed by the database.[6]

If an administrator enables the storage auto-scaling feature, the control plane will automatically allocate additional block storage capacity when utilization thresholds are consistently breached. While this automation prevents catastrophic out-of-space application outages, it introduces an insidious ratchet effect on the monthly billing baseline. Amazon RDS storage volumes can be dynamically expanded, but the underlying architecture prohibits them from ever being shrunk, ensuring that any temporary spike in data volume results in a permanent, irreversible increase in the monthly storage cost.

Data transfer egress and ingress

Data transfer pricing within cloud infrastructure operates on a rigid boundary-crossing principle. The ingress of data - specifically traffic flowing from the public internet or external AWS regions into the database instance - is entirely free of charge.[10] This policy is designed to encourage the accumulation of data within the cloud provider's ecosystem. Conversely, egress traffic moving outward from the database triggers metered charges that vary based on the destination.

Data transferred from an Amazon RDS instance to the public internet is billed on a tiered, volume-based scale. The egress pricing structure consistently penalizes smaller workloads while offering marginal discounts at massive scale.[5]

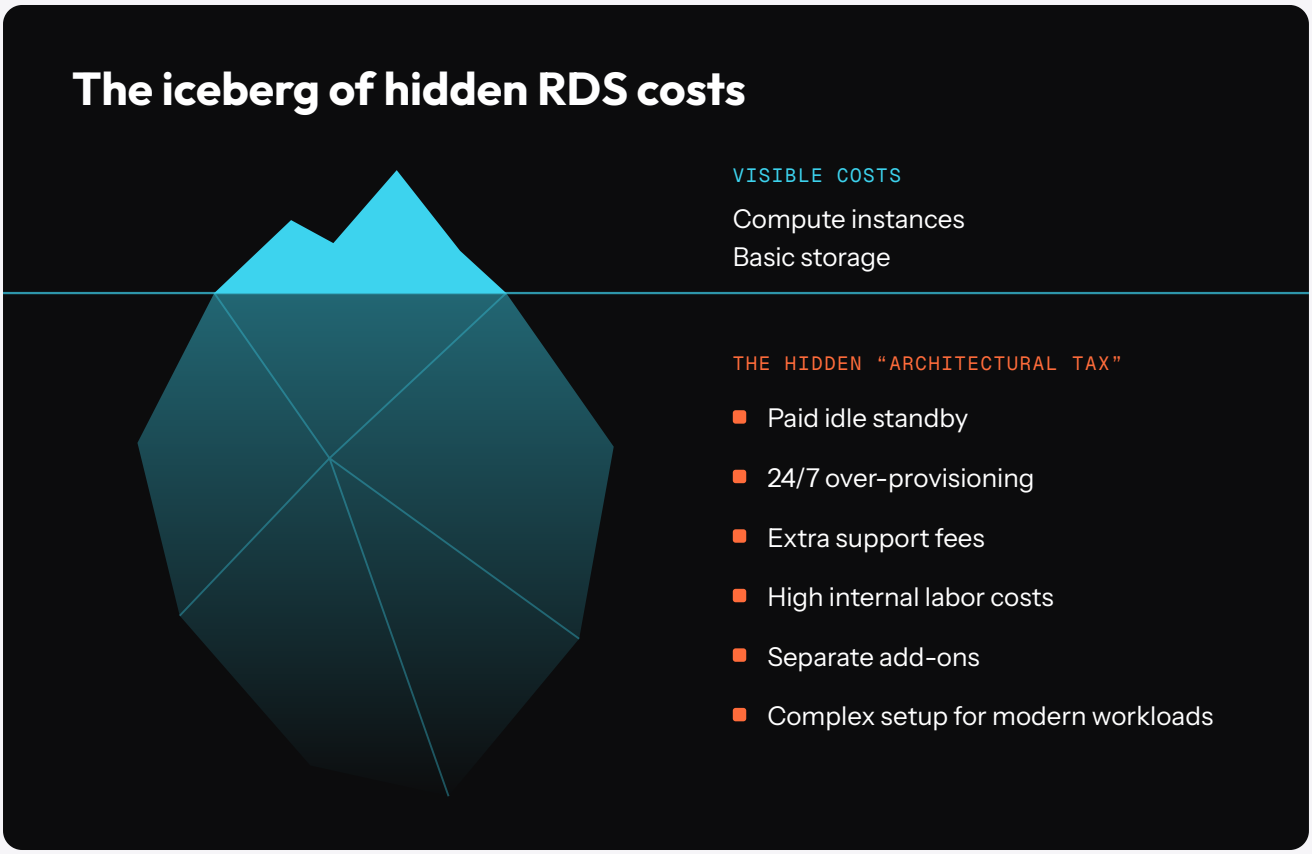
TRANSFER DESTINATION	MONTHLY VOLUME TIER	EGRESS COST PER GB
Internet egress	First 10 TB / month	$0.09
Internet egress	Next 40 TB / month	$0.085
Internet egress	Over 150 TB / month	$0.070
Cross-region	Any volume	$0.02
Cross-Availability Zone	Any volume	$0.01 (each direction)

TABLE 1 Standard data transfer egress rates illustrating the tiered pricing model.

More critically for modern, distributed microservice architectures, data traversing between distinct Availability Zones within the exact same AWS Region incurs a $0.01 per GB charge in both directions.[5] While the service specifically exempts the internal replication traffic moving between a primary database node and its synchronous Multi-AZ standby from these fees, any cross-AZ traffic generated by application servers querying the database remains fully billable.[2] In highly available architectures where application fleets are distributed randomly across three Availability Zones, a mathematical probability ensures that the majority of database queries cross an AZ boundary, silently compounding the monthly data transfer invoice.

05

The hidden costs



The true complexity of cloud financial management reveals itself not in the primary compute and storage layers, but in the myriad auxiliary services, operational penalties, and external integration mechanisms required to operate a production-grade database securely and effectively. These secondary and tertiary billing dimensions frequently escape initial cost modeling, compounding silently as workloads scale and architectural complexity increases.

RDS Proxy and connection management

Relational database engines consume significant internal CPU and memory resources to negotiate, establish, and maintain open client connections. In modern serverless architectures, where thousands of ephemeral compute functions may spin up simultaneously in response to traffic spikes, the database can rapidly exhaust its finite connection limits. This phenomenon, known as a connection storm, leads to severe application timeouts, memory exhaustion, and service degradation.

To resolve this architectural friction, Amazon provides RDS Proxy, a fully managed, highly available database proxy that sits between the application and the database to pool and share established connections. While this proxy effectively shields the database engine from connection exhaustion, it introduces a separate, continuous, and significant billing dimension. For provisioned instances, RDS Proxy is billed at $0.015 per vCPU hour based entirely on the size of the underlying database instance it serves, demanding a minimum charge of 2 vCPUs.[13] Therefore, if an enterprise is operating a highly scaled fleet of r5.4xlarge instances, each possessing 16 vCPUs, attaching the proxy mechanism adds hundreds of dollars in hidden compute charges per instance, per month. This cost accrues steadily, hour by hour, completely irrespective of the actual volume of application traffic passing through the proxy layer.[13]

Observability: CloudWatch Database Insights and Performance Insights

Deep database observability is a non-negotiable operational requirement for identifying slow query execution, resolving transaction deadlocks, and mitigating index inefficiencies. Amazon RDS addresses this by automatically enabling Performance Insights during the instance creation process, providing a specialized graphical tool designed to visualize database load and isolate precise performance bottlenecks at the query level.[14]

The baseline free tier of Performance Insights offers a rolling seven-day retention period for performance data history and accommodates up to one million API requests per month at no additional charge.[15] However, comprehensive database performance tuning often necessitates longitudinal data analysis spanning multiple weeks to capture seasonal traffic variations and long-term degradation trends. Extending this retention period incurs a substantial premium. If an administrator selects a paid flexible retention tier ranging from one to 24 months, the organization is billed a flat rate of $1.50 per vCPU per month.[15] On highly scaled instances with dozens of cores, this observability tax rapidly evolves into a prominent line item on the monthly invoice.

Furthermore, the architectural landscape for database observability is currently undergoing a forced, provider-mandated migration. By June 30, 2026, the native Performance Insights console experience and its associated flexible retention pricing models will reach official End of Life (EOL).[14] Users will be compelled to migrate their observability workflows to the "Advanced mode" of Amazon CloudWatch Database Insights. Instances that are not proactively upgraded by administrators before the 2026 deadline will default to a restricted "Standard mode." This default state results in the permanent loss of access to performance data history beyond seven days, alongside the revocation of critical execution plan diagnostics and on-demand lock analysis capabilities.[14] The transition to CloudWatch also exposes organizations to secondary API polling costs. Custom monitoring dashboards that refresh frequently generate massive volumes of API calls; any requests beyond the initial 1 million free tier allocation are billed at $0.01 per 1,000 requests, silently accumulating against operations teams utilizing real-time monitoring.[15]

Backup storage accumulation

Data protection and disaster recovery are foundational tenets of managed database services. Amazon provides automated backup storage at no additional charge, but this allocation is strictly capped at 100% of the total provisioned database storage size for a specific region.[5] For example, an active database provisioned with 500 GiB of Elastic Block Store capacity is allocated a commensurate 500 GiB of free backup storage. However,

stringent organizational compliance mandates and auditing requirements frequently dictate extended backup retention windows, coupled with the routine creation of manual snapshots prior to risky application deployments.

When the aggregate volume of stored manual snapshots and automated transaction logs exceeds the 100% provisioned storage baseline, the excess capacity is billed at a rate of $0.095 per GiB-month.[5] A highly active transactional database generating large volumes of transaction logs can easily and quickly breach this free threshold. Critically, if an infrastructure administrator temporarily stops a database instance in an attempt to save on hourly compute costs during non-business hours, the instance will persistently continue to accrue charges for its provisioned storage volume and any associated backup storage.[2] When the underlying database instance is permanently terminated, the free-tier offset is instantly eradicated, and any retained manual snapshots are subsequently billed at the full $0.095 per GiB-month rate indefinitely.[5]

Snapshot export and S3 integration

Data engineering pipelines frequently rely on relational database snapshots to seed analytical data lakes or external business intelligence environments. Amazon provides a specific feature to automatically export snapshot data directly to Amazon Simple Storage Service (S3) in the Apache Parquet format. This columnar format is highly advantageous as it reduces storage consumption and accelerates downstream processing by query engines such as Amazon Athena.[16]

However, this convenience is governed by a highly rigid and expensive pricing structure. The snapshot export process is metered and billed at $0.010 per GB of the entire, aggregate snapshot size, regardless of whether the user applies filtering rules to export only specific, smaller tables.[5] For instance, executing a filtered export of a 10 GB subset table from a massive 1 TB database snapshot incurs compute processing charges based entirely on the full 1 TB volume. Furthermore, the export process is fundamentally non-incremental.[16] Subsequent exports of data from the exact same snapshot, or the routine daily export of newly generated snapshots, trigger full billing for the entire dataset volume every single execution. These distinct export processing costs run concurrent with, and in addition to, the standard Amazon S3 PUT request fees and ongoing object storage fees for housing the resulting Parquet files.[18]

Extended Support pricing crisis

Database engines operate on strict lifecycle schedules dictated by their open-source community maintainers or commercial vendors. When a major version reaches community EOL, the upstream maintainers officially cease providing critical security patches, bug fixes, and performance updates. To protect legacy workloads that cannot be immediately upgraded due to complex application dependencies or technical debt, AWS provides an Extended Support program, guaranteeing the provision of critical security patches for up to three additional years.[2]

This grace period, however, functions as an aggressive, escalating financial punitive mechanism explicitly designed to force organizations into executing upgrades. As of March 1, 2026, the pricing structure for Extended Support Year 3 underwent a massive escalation, effectively doubling the penalty rates for highly deployed legacy engines, most notably MySQL 5.7 and PostgreSQL 11.[20]

The financial impact of this penalty is calculated on a strict per-vCPU, per-hour basis. Prior to March 2026, the surcharge in regions such as US East (Ohio) was $0.100 per vCPU-hour. Upon reaching the March 1, 2026 threshold, this rate doubled to $0.200 per vCPU-hour.[16]

INSTANCE NODE SIZE	VCPU ALLOCATION	BASE MONTHLY COMPUTE (APPROX.)	EXTENDED SUPPORT SURCHARGE (POST-MARCH 2026)	TOTAL MONTHLY COST IMPACT
db.m5.large	2	~$124	+$292	**~$416**
db.r5.2xlarge	8	~$730	+$1,168	**~$1,898**
db.r5.12xlarge	48	~$4,380	+$7,008	**~$11,388**

TABLE 2 The financial impact of the Extended Support (Year 3) penalty pricing effective March 1, 2026.

As the data clearly demonstrates, the Extended Support surcharge easily and consistently exceeds the base cost of the underlying compute instance itself.[20] This dynamic creates catastrophic, unforecasted budgetary overruns for engineering teams paralyzed by application compatibility constraints. For organizations utilizing serverless deployments such as Aurora Serverless v2, the penalty is applied per Aurora Capacity Unit per hour, creating equal, scaling financial exposure for unpredictable workloads.[2]

The hidden networking tax: NAT Gateways and PrivateLink

Strict security architecture best practices mandate that relational databases must be deployed exclusively within private subnets, maintaining no direct ingress or egress routes to the public internet. However, auxiliary architectural requirements frequently necessitate outbound internet connectivity. For example, a Lambda function residing in the same private subnet may require access to both the internal database instance and an external third-party payment API, or the database itself may require outbound access to pull specific software updates. This outbound routing is traditionally achieved via a Network Address Translation (NAT) Gateway.

NAT Gateways represent one of the most insidious and rapidly accumulating hidden costs in cloud networking.[21] They are metered across three compounding dimensions:

1. **Hourly provisioning rent:** Each deployed NAT Gateway costs $0.045 per hour, equating to approximately $32.85 per month simply to exist, regardless of whether a single byte of traffic passes through it.[12] A highly available, enterprise-grade architecture spanning three Availability Zones demands the deployment of three independent NAT Gateways, pushing the base idle cost to nearly $100 per month.[23]
2. **Data processing toll:** Every gigabyte of data passing through the NAT Gateway incurs a strict $0.045 data processing fee.[12]
3. **Internet egress:** Once processed by the NAT Gateway, the traffic incurs standard regional outbound data transfer fees, typically beginning at $0.09 per GB.[12]

Consequently, processing one terabyte of outbound traffic routed through a NAT Gateway effectively costs $135 ($45 for the processing toll plus $90 for the egress fee), representing a massive financial burden entirely separate from the core database billing metrics.

Alternatively, network architects can utilize AWS PrivateLink via Interface VPC Endpoints to securely route traffic to other AWS services without traversing the public internet or a highly metered NAT Gateway. While this effectively mitigates the heavy NAT processing fees, VPC Endpoints carry their own distinct financial burden: an hourly provisioning charge of $0.01 per endpoint per Availability Zone, plus a data processing charge of $0.01 per GB.[12] An enterprise maintaining dedicated endpoints for S3, Secrets Manager, CloudWatch, and Systems Manager across multiple Availability Zones rapidly accumulates hundreds of dollars in baseline networking overhead. Furthermore, the assignment of public IPv4 addresses across the infrastructure generates an unavoidable, continuous rent of $0.005 per hour, or roughly $3.65 per month, per IP address.[12]

OLAP support via Redshift and Zero-ETL

Amazon RDS is fundamentally engineered and optimized for Online Transaction Processing (OLTP). Its row-based storage architecture excels at executing high-velocity, single-record read and write operations but degrades severely when subjected to complex, table-scanning analytical queries typical of Online Analytical Processing (OLAP). To execute real-time business intelligence without paralyzing the primary transactional system, organizations are forced to replicate their relational data into a dedicated, columnar data warehouse, specifically Amazon Redshift.[26]

Historically, achieving this replication required the construction of bespoke extract, transform, and load (ETL) pipelines, or the deployment of separate change data capture (CDC) services such as the AWS Database Migration Service (DMS), Managed Streaming for Apache Kafka (MSK), or Apache Flink. Utilizing DMS introduces dedicated compute replication instances billed hourly based on their size, alongside supplementary storage requirements for replication logs, swap space, and data caching.[9] Furthermore, DMS scales compute capacity through DMS Capacity Units (DCUs), where pricing scales aggressively depending on whether the replication is configured for Single-AZ or Multi-AZ resilience.[9]

To streamline this complex integration and reduce pipeline maintenance, AWS introduced the Zero-ETL integration. This feature promises near real-time replication of transactional data from sources like Amazon Aurora and Amazon RDS directly into Redshift without the need to maintain custom code.[28] While AWS explicitly and frequently states that the Zero-ETL integration itself does not carry an additional fee, the downstream architectural requirements generate severe, unavoidable cost spikes.[26]

The primary financial danger lies deeply within the mechanics of Redshift Serverless and continuous CDC. Redshift Serverless bills compute capacity in Redshift Processing Units per second, enforcing a minimum charge of 60 seconds for any activity.[31] The Zero-ETL integration relies on a continuous, never-ending stream of CDC events to maintain near real-time data latency. This persistent, low-volume streaming completely disables the native auto-pause capabilities of Redshift Serverless.[30] Furthermore, configuring a Redshift Serverless workgroup as a target for a Zero-ETL integration algorithmically mandates a strict minimum base capacity of 32 Redshift Processing Units.[29] At a standard regional rate of approximately $0.36 per RPU-hour, an active 32-RPU cluster costs $11.52 per hour.[26] Because the continuous influx of CDC data prevents the cluster from ever sleeping or scaling down to zero, organizations are billed 24 hours a day, seven days a week for 32 RPUs. This results in an absolute baseline cost of over $8,200 per month simply to receive the replicated data, completely irrespective of whether a single analytical query is ever executed by a human user or BI tool.[30]

Compounding this financial burden is the highly rigid nature of the replicated data. Tables populated via Zero-ETL in the Redshift environment are strictly read-only.[30] To perform necessary downstream dimensional modeling, data engineers must provision secondary materialized views or transformation tables, inflating the required Redshift Managed Storage (RMS) allocation and consuming further compute RPUs for ongoing data restructuring, generally adding 30% to 40% more development overhead.[30]

Vector search, RAG, and AI workloads

The rapid proliferation of generative artificial intelligence (GenAI) has positioned retrieval-augmented generation (RAG) as a mandatory architectural pattern for modern applications. Implementing RAG requires the ability to transform proprietary textual data into high-dimensional mathematical vectors, known as embeddings, and subsequently perform semantic similarity searches against those vectors in response to user queries.

Standard relational database engines natively lack optimized, high-performance vector processing capabilities. While PostgreSQL supports the popular pgvector extension, deploying it on standard infrastructure introduces profound operational risks and performance bottlenecks. Vector similarity search algorithms, particularly Hierarchical Navigable Small World graphs, are exceptionally memory-intensive.[32] Forcing an OLTP database to house, index, and query millions of high-dimensional embeddings violently pollutes the database buffer pools. This process evicts standard relational data from memory, forcing the database to read from disk and severely degrading core transactional throughput. While updates such as pgvector version 0.8.0 introduced iterative scanning and enhanced query planning, offloading vector workloads onto a relational node almost always forces the organization to execute vertical scaling to highly expensive, memory-heavy R-series instances to compensate for the memory pressure.[32]

To avoid destabilizing the primary transactional database, cloud architects are corralled into utilizing highly fragmented, disparate AI ecosystems. This typically involves routing data from the database into Amazon SageMaker for custom embedding generation, or leveraging Amazon Bedrock orchestrated with Amazon OpenSearch Serverless. This highly distributed architecture is exceptionally complex and financially punitive.

The integration establishes a hard billing floor through vector store minimums. Setting up a Knowledge Base in Amazon Bedrock triggers the automatic, background creation of an OpenSearch Serverless vector collection. This mechanism mandates a minimum provisioning of two OpenSearch Compute Units - one dedicated to indexing and one for search replication - costing approximately $350 per month, even if the database sits entirely idle with zero incoming queries.[33]

Transforming the raw relational data into vectors requires routing data through embedding models hosted on SageMaker or Bedrock, triggering strict inference charges billed per million tokens processed.[35] Furthermore, advanced RAG pipelines require semantic reranking to improve the mathematical accuracy of the returned results. Applying Amazon Rerank 1.0 adds a distinct surcharge of $1.00 per 1,000 queries.[34] Utilizing Bedrock Guardrails to filter malicious inputs or hallucinated outputs adds another continuous processing fee of $0.30 per 1,000 text units.[34] Finally, the constant requirement to synchronize data - moving updated relational records from the source database to the OpenSearch vector index - requires maintaining custom Lambda functions or continuous streaming pipelines, adding yet another layer of compute and data processing fees to the monthly ledger.

The idle standby penalty and active-active complexities

While Multi-AZ deployments are heavily marketed as the standard for high availability, they introduce a massive cost inefficiency: the standby node is strictly passive. In a standard Amazon RDS Multi-AZ instance deployment, the synchronous standby replica cannot be utilized to serve read traffic; it sits entirely idle until a failover event is triggered. The organization is effectively paying for 100% of the compute capacity of a secondary node that provides zero throughput value during normal operations. If an application requires read scaling, the organization must provision, and pay for, entirely separate Read Replicas on top of their Multi-AZ deployment.

Furthermore, while Amazon RDS for MySQL (versions 8.0.35 and higher) supports active-active architectures via the MySQL Group Replication community plugin, this is not a fully managed, seamless experience. It requires significant manual configuration, and if a database instance fails to rejoin the group, administrators must frequently intervene and manually restart the group replication.

The absence of native compute autoscaling

Standard Amazon RDS natively supports storage autoscaling, seamlessly expanding persistent block storage as datasets grow. However, it explicitly lacks native vertical or horizontal compute autoscaling. If a standard, non-Aurora RDS instance experiences a surge in traffic, the service has no built-in capability to automatically scale up the CPU or memory. To achieve automated compute scaling, infrastructure teams are forced to build custom, complex workflows using Amazon CloudWatch alarms, Amazon SNS notifications, and AWS Lambda functions to trigger instance modifications. Because this DIY automation introduces operational risk and downtime during vertical scaling, most organizations simply choose to permanently over-provision their instance sizes to handle peak traffic, resulting in massive wasted compute spend during idle hours.

The hidden human cost: database administrators

The marketing promise of fully managed cloud databases often leads organizations to falsely believe they no longer require database administrators (DBAs). Amazon RDS automates infrastructure-level tasks such as hardware provisioning, operating system patching, and backups, but it operates on a strict shared responsibility model. Critical database functions such as query optimization, indexing strategies, schema design, and deep performance tuning remain entirely DIY. AWS provides reactive infrastructure support, meaning they will not proactively optimize your slow queries. Consequently, organizations pay a massive managed service markup on the computer instances while simultaneously bearing the retained cost of employing a full-time, in-house DBA team to keep the database running efficiently.

06

A predictable alternative: MariaDB Cloud

The MariaDB advantage

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

The compounding, highly obfuscated hidden costs associated with traditional managed database services - driven extensively by proxy taxes, forced observability migrations, rigid and expensive ETL pipelines, and fragmented, multi-service AI architectures - highlight a systemic flaw in heavily unbundled cloud service models. Organizations seeking high performance without the labyrinthine billing dimensions are increasingly abandoning fragmented architectures and pivoting toward unified, natively multimodal database platforms. MariaDB Cloud, which encompasses advanced serverless capabilities, represents a profound architectural remedy to these specific financial and operational pain points.[36]

Unified workload architecture and HTAP

The most financially destructive aspect of the legacy relational ecosystem is the strict requirement to duplicate data across multiple distinct services to achieve both transactional speed and analytical depth. MariaDB Cloud circumvents this paradigm entirely through a native Hybrid Transactional/Analytical Processing architecture.[38]

Through the deep integration of MariaDB Exa - an advanced, massively parallel processing and in-memory columnar engine powered by Exasol - MariaDB Cloud allows complex, table-scanning analytical queries to execute directly against real-time business data.[38] A proprietary, highly intelligent query accelerator routes rapid transactional inserts to the traditional row-based engine while simultaneously enabling parallel analytical aggregations on the columnar format.[40]

This unified, single-engine paradigm completely eradicates the necessity for configuring expensive Zero-ETL pipelines, bypassing the crippling Redshift Serverless 32-RPU minimums, and eliminating the need for standalone AWS DMS replication instances.[38] By eliminating the fundamental requirement to move data across network boundaries for analysis, organizations bypass the compounding cross-AZ data transfer fees, avoid NAT Gateway data processing charges, and escape the dual-storage penalty inherent in maintaining separate, synchronized OLTP and OLAP environments.[41]

Native vector search and agentic AI integration

MariaDB Enterprise Platform directly incorporates native vector search capabilities into the absolute core of the database engine.[41] Unlike bolt-on extension implementations that suffer from severe memory configuration complexities and buffer pool pollution, MariaDB implements a highly optimized, native variation of HNSW indexing.[41] This engineering choice seamlessly integrates the storage and querying of vector embeddings alongside traditional relational structured data without degrading transactional performance.[41]

By unifying semantic search with standard SQL syntax, applications can seamlessly execute composite queries that filter on exact relational metrics while simultaneously ordering the results by vector similarity, all executed within a single, standard database connection.[41] This natively multimodal approach radically simplifies the application logic layer, entirely eliminates the need to synchronize with the vector store and maintain consistency, and significantly reduces the corporate security attack surface by maintaining all proprietary data within a single, governable perimeter.[41]

Furthermore, MariaDB Cloud expands upon this foundation with a Model Context Protocol (MCP) Server specifically engineered for agentic AI. The MariaDB Cloud MCP Server acts as an intermediary, allowing AI applications like Cursor, Claude Desktop, and Visual Studio Code Copilot to securely and efficiently interact with your MariaDB Cloud databases. It exposes your data in a structured way that AI tools can query, transforming complex database interactions into intuitive, real-time conversations.[38]

Proactive support and Managed Operations

Unlike the strictly reactive, infrastructure-only support models of traditional cloud providers, MariaDB Cloud offers a proactive, consultative approach through its Managed Operations service. Rather than leaving performance tuning and query optimization as a DIY exercise, Managed Operations provides fractional DBA services for expert database maintenance, deep troubleshooting, and tailored architectural scaling strategies.[3] This directly eliminates the hidden human costs of maintaining large, expensive in-house database administration teams while ensuring the platform runs at peak efficiency.

Transparent and predictable economics

MariaDB Cloud operates on a transparent, highly predictable tiered financial structure that actively resists the micro-metering philosophies of broader public cloud ecosystems. The platform provides a unified experience right out of the box, requiring no complex architectural "figuring things out" to achieve predictable billing. The platform is segmented into three primary, all-inclusive offerings:

- **Foundation:** A fully managed iteration of the community server, highly suitable for developmental environments and baseline production workflows, offering a free tier for developers.[38]
- **Power:** An enterprise-grade deployment offering continuous availability, real-time HTAP analytics via MariaDB Exa, the native AI and vector search capabilities, and advanced support mechanisms, starting at highly competitive hourly rates of $0.16 per hour.[38]
- **PowerPlus:** The apex tier, starting at $0.21 per hour, which inclusively packages geo-distributed resilience, without demanding secondary service subscriptions.[38]

Crucially, the serverless architecture resolves several hidden cost vectors natively. Storage auto-scaling is intelligent and entirely transparent, expanding dynamically without imposing the hidden, punishing IOPS tax penalties without imposing the additional tax with an over-provisioned storage layer. Primary storage pricing is highly predictable, leveraging standard AWS gp3 volumes at a flat rate of $0.1000 without obfuscated multipliers.[38] Furthermore, the inclusion of an intelligent proxy routing system directly within the MariaDB Cloud architecture ensures that always-on connections and complex session state management are handled gracefully and automatically. This native capability permanently mitigates the need to procure, configure, and fund external connection pooling compute layers, completely eliminating the hand stitched proxy taxes inherent in alternative managed database services.[47]

The total cost of ownership for managed database infrastructure cannot be accurately modeled by merely isolating base compute instance rates and provisioned storage volume capacity. The true economic burden resides deeply within the required architectural periphery: the networking taxes of NAT Gateways and cross-AZ replication, the punitive penalty pricing of forced Extended Support migrations, the non-incremental nature of data exports, and the forced dependency on highly metered, fragmented external services for modern analytics and artificial intelligence. By consolidating OLTP, OLAP, and vector operations into a single, cohesive, intelligently scaled engine, alternatives like MariaDB Cloud present a mathematically superior, predictable economic model. This convergence eradicates continuous compute penalties and successfully restores database infrastructure to its fundamental objective: reliable, high-performance data processing without the hidden operational tax.

Get started with MariaDB Cloud for free

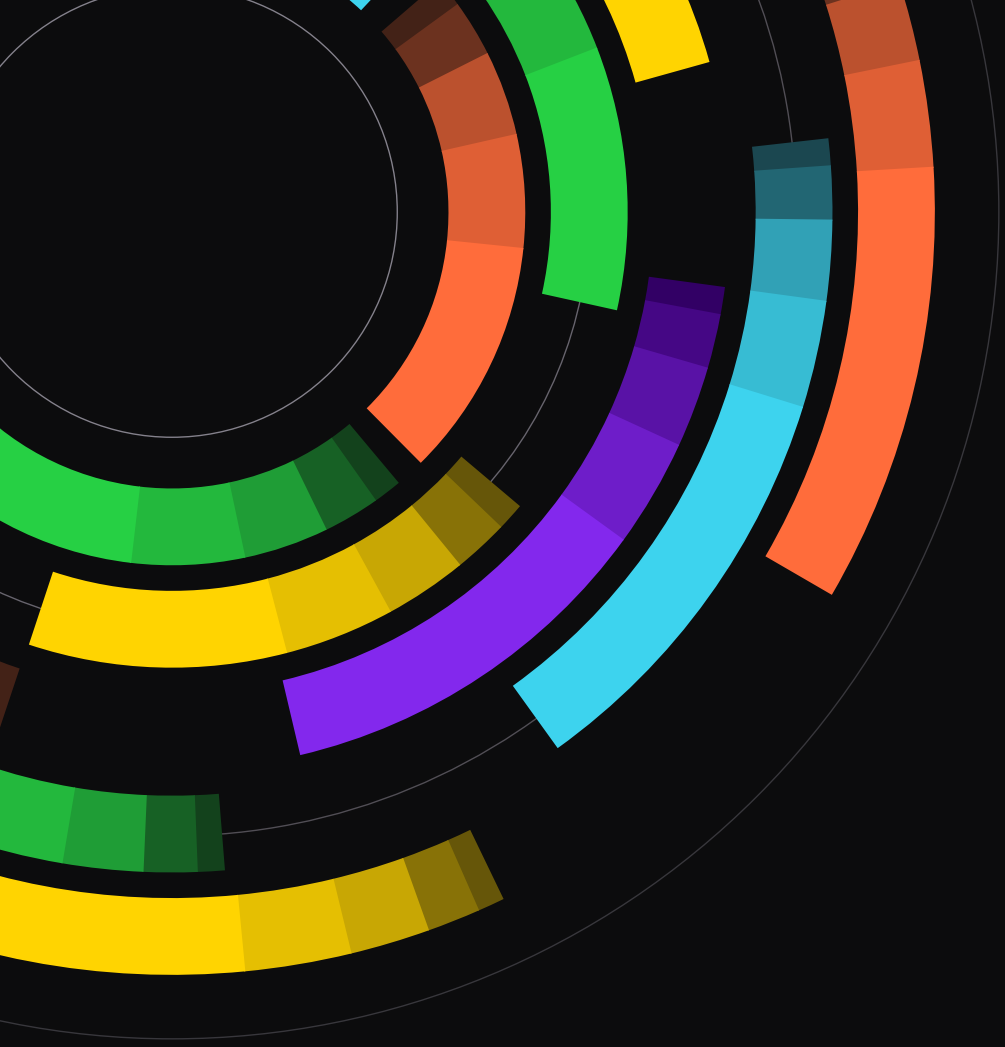
mariadb.com/cloud-get-started

SOURCES

Sources and references

1. The Ultimate Guide to Cloud Storage Pricing in 2026: Hidden Fees, Egress Costs & How to Avoid Overpaying - https://medium.com/@orboncloud/the-ultimate-guide-to-cloud-storage-pricing-in-2026-hidden-fees-egress-costs-how-to-avoid-b8e7bcac2c5d
2. Amazon RDS Pricing - https://aws.amazon.com/rds/pricing/
3. OpenWorks 2023 - SkySQL vs. AWS RDS vs. GCP Cloud SQL - https://mariadb.com/wp-content/uploads/2023/05/OpenWorks-2023-SkySQL-vs-AWS-RDS-vs-GCP-Cloud-SQL-Venkat-Iyer.pdf
4. Amazon RDS for SQL Server Pricing - https://aws.amazon.com/rds/sqlserver/pricing/
5. Amazon RDS Pricing | Every Engine Cost & Hidden Fee (2026) - https://go-cloud.io/amazon-rds-pricing/
6. Amazon RDS for MariaDB pricing - https://aws.amazon.com/rds/mariadb/pricing/
7. DB instance billing for Amazon RDS - https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/User_DBInstanceBilling.html
8. The Ultimate Guide to AWS RDS Pricing: A Comprehensive Cost Breakdown 2025 - https://cloudchipr.com/blog/rds-pricing
9. AWS DMS Pricing - https://aws.amazon.com/dms/pricing/
10. Overview of Data Transfer Costs for Common Architectures - https://aws.amazon.com/blogs/architecture/overview-of-data-transfer-costs-for-common-architectures/
11. Exploring Data Transfer Costs for AWS Managed Databases - https://aws.amazon.com/blogs/architecture/exploring-data-transfer-costs-for-aws-managed-databases/
12. The Complete Guide to Cloud Networking Costs: VPCs, NAT Gateways and Data Transfer - https://zop.dev/resources/blogs/the-complete-guide-to-cloud-networking-costs-vpcs-nat-gateways-and-data-transfer/
13. Amazon RDS Proxy Pricing - https://aws.amazon.com/rds/proxy/pricing/
14. Overview of Performance Insights on Amazon Aurora - https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/USER_PerfInsights.Overview.html
15. Performance Insights Pricing - https://aws.amazon.com/rds/performance-insights/pricing/
16. Amazon RDS for MySQL pricing - https://aws.amazon.com/rds/mysql/pricing/
17. Exporting DB snapshot data to Amazon S3 for Amazon RDS - https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_ExportSnapshot.html
18. Amazon Aurora Pricing - https://aws.amazon.com/rds/aurora/pricing/
19. Amazon S3 pricing - https://aws.amazon.com/s3/pricing/
20. Amazon RDS Pricing: MySQL 5.7 Users Just Got a $292/Month Fee - https://cloudburn.io/blog/amazon-rds-pricing
21. The Hidden Cloud Tax: How IPv4 Rent and Egress Fees Are Silently Crushing 2026 Budgets - https://www.cloudcostchefs.com/blog/cloud-networking-costs-ipv4-egress-2026
22. Amazon VPC Pricing: 4 Charges Hiding Behind the Free VPC - https://cloudburn.io/blog/amazon-vpc-pricing
23. The Silent $33/Month Charge: Understanding AWS NAT Gateway Costs - https://dev.to/cloudwiseteam/the-silent-33month-charge-understanding-aws-nat-gateway-costs-19c3
24. Trying to better understand NAT pricing : r/aws - https://www.reddit.com/r/aws/comments/1b83lu1/trying_to_better_understand_nat_pricing/
25. AWS PrivateLink Pricing - https://aws.amazon.com/privatelink/pricing/
26. Amazon Redshift Pricing - https://aws.amazon.com/redshift/pricing/
27. Amazon Redshift pricing: What to expect and how it compares - https://www.rudderstack.com/blog/amazon-redshift-pricing/
28. Simplify data integration using zero-ETL from Amazon RDS to Amazon Redshift - https://aws.amazon.com/blogs/database/simplify-data-integration-using-zero-etl-from-amazon-rds-to-amazon-redshift/

29. Zero-ETL integrations - Amazon Redshift - https://docs.aws.amazon.com/redshift/latest/mgmt/zero-etl-using.html
30. AWS Zero-ETL to Redshift: Real-Time Wins, Hidden Traps - https://medium.com/@amitmeme/aws-zero-etl-to-redshift-real-time-wins-hidden-traps-and-the-cost-smart-path-to-production-b15d0b5e5928
31. Amazon Redshift Pricing - https://www.amazonaws.cn/en/redshift/pricing/
32. Supercharging vector search performance and relevance with pgvector 0.8.0 on Amazon Aurora PostgreSQL - https://aws.amazon.com/blogs/database/supercharging-vector-search-performance-and-relevance-with-pgvector-0-8-0-on-amazon-aurora-postgresql/
33. Amazon OpenSearch Service - Pricing - https://aws.amazon.com/opensearch-service/pricing/
34. Amazon Bedrock Pricing: Token Rates Hide a $350/Month Trap - https://cloudburn.io/blog/amazon-bedrock-pricing
35. Amazon Bedrock Pricing - https://aws.amazon.com/bedrock/pricing/
36. MariaDB Accelerates Cloud Deployments, Adds Agentic AI and Serverless Capability with Acquisition of SkySQL - https://mariadb.com/newsroom/press-releases/mariadb-accelerates-cloud-deployments-adds-agentic-ai-and-serverless-capability-with-acquisition-of-skysql/
37. Announcing the Release of MariaDB Enterprise Platform 2026 - https://mariadb.com/resources/blog/announcing-the-release-of-mariadb-enterprise-platform-2026/
38. MariaDB Cloud Pricing - https://mariadb.com/pricing/#mariadb-cloud
39. MariaDB Enterprise Platform Overview - https://mariadb.com/resources/datasheets/mariadb-enterprise-platform/
40. MariaDB unifies transactional, analytical, and vector databases in MariaDB Enterprise Platform 2026 release - https://sdtimes.com/data/mariadb-unifies-transactional-analytical-and-vector-databases-in-mariadb-enterprise-platform-2026-release/
41. Introducing Vector Search With the Latest Version of MariaDB - https://mariadb.com/resources/blog/introducing-vector-search-with-the-latest-version-of-mariadb-enterprise-platform/
42. Database Workload Versatility - https://mariadb.com/products/enterprise/workload-versatility/
43. Evaluating vector indexes in MariaDB and pgvector: part 1 - http://smalldatum.blogspot.com/2025/01/evaluating-vector-indexes-in-mariadb.html
44. Introduction to MariaDB Vector Search - Severalnines - https://severalnines.com/blog/introduction-to-mariadb-vector-search/
45. Compare MariaDB vs. SkySQL in 2026 - https://slashdot.org/software/comparison/MariaDB-vs-SkySQL/
46. SkySQL MariaDB vs. AWS Aurora MySQL: Price-Performance Comparison - https://skysql.com/blog/skysql-mariadb-vs-aws-aurora-mysql-a-price-performance-comparison
47. MariaDB Cloud Serverless - https://mariadb.com/docs/mariadb-cloud/readme/serverless
48. Enterprise Database Pricing for MariaDB - https://mariadb.com/pricing/



mariaDB

Get started with MariaDB Cloud for free

mariadb.com