

# Fireside Chat: Best practices for cost optimizations for the AWS database portfolio

Jagdish Mirani, Sr. Product Marketing Manager, AWS

Vlad Vlasceanu, Sr. Pr Database SA Tech Leader, AWS

07/20/23

# AWS Database Cost Optimization

- Choosing the right infrastructure
- Optimizing your license structure
- In-Memory Caching
- Data lifecycle decisions
- Monitoring and tracking cost metrics
- Using pricing calculators and estimation tools

# NEW! Improved price performance and price predictability with Amazon Aurora I/O-Optimized

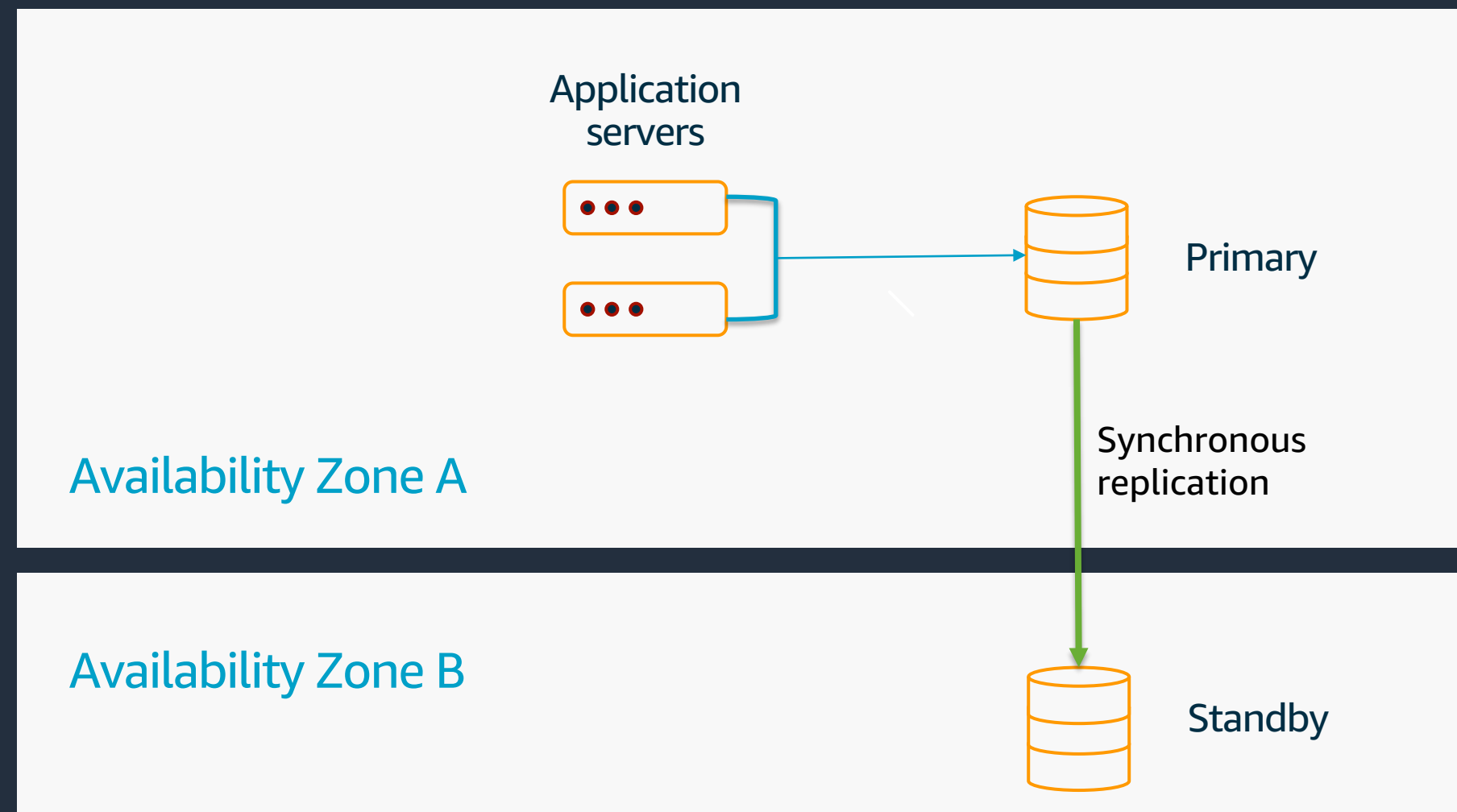


- Aurora cluster configuration with the option to pay for compute and storage only with no charges for read and write I/O operations
- Price predictability: no pay-per-request I/O charges making it easy to estimate database spend upfront
- For customers whose I/O spend exceeds 25% of total Aurora database spend, customers can save up to 40% cost savings
- Improved performance: increasing throughput and reducing latency for I/O-intensive applications
- Available for Aurora PostgreSQL-Compatible Edition and Aurora MySQL-Compatible Edition
- Supported on Aurora Serverless v2 and provisioned (on-demand and reserved) instances

# Graviton based price / performance

- AWS designed processors for performance optimization of cloud workloads running on EC2
- Currently on third generation Graviton3 processors
- Databases available on Graviton2: Amazon Aurora, RDS, Neptune, DocumentDB, MemoryDB for Redis, ElastiCache
- Graviton2: 52% better price/performance on RDS and 35% better price/performance on Aurora
- Graviton3: 20% price/performance improvement vs. Graviton2 for Aurora and 27% for RDS open-source

# Compute: cost optimization strategies



## Cost

- DB Instance per hour cost (metered per second, min 10 minutes)

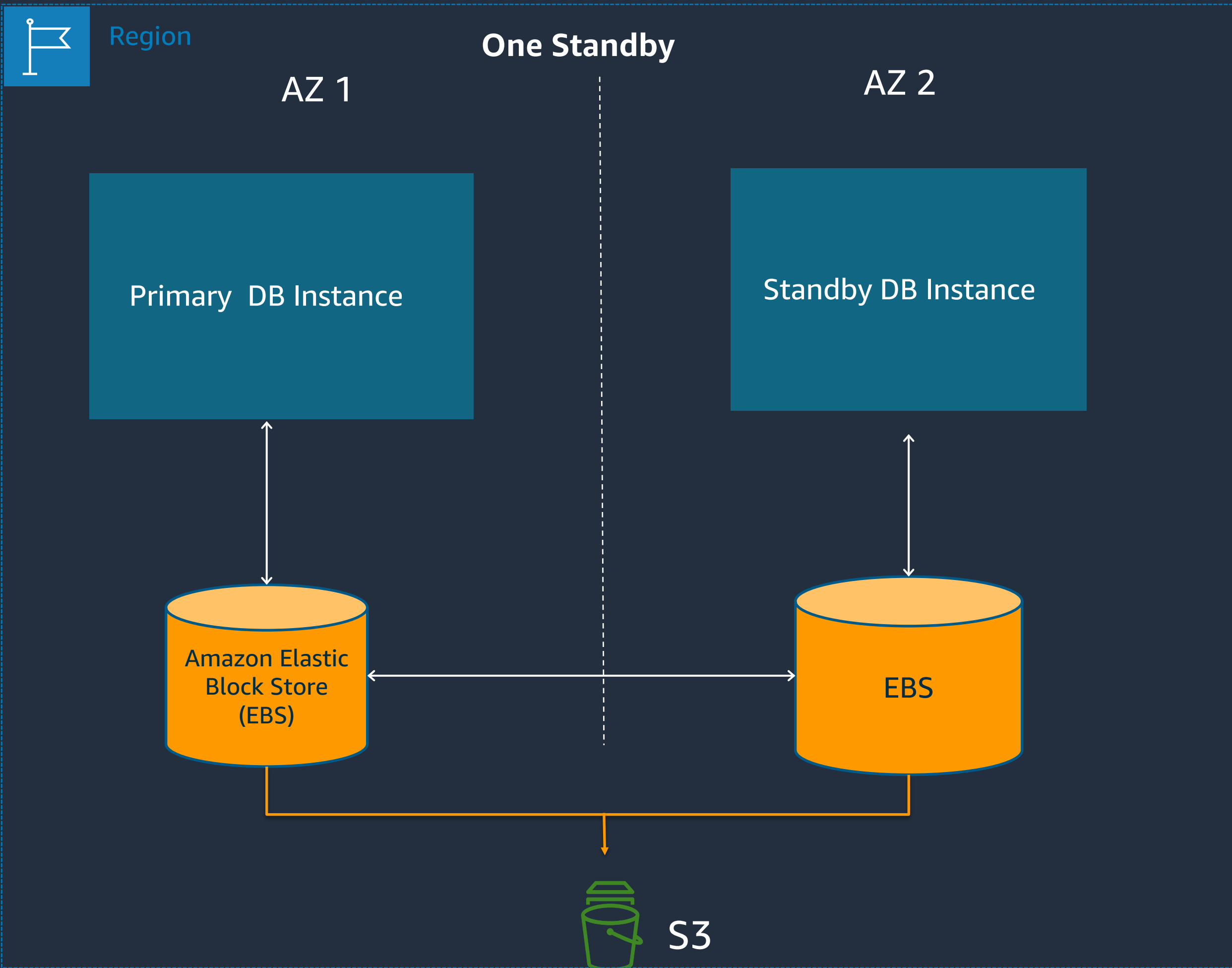
## Cost Monitoring

- CloudWatch metrics: CPUUtilization , FreeableMemory, NetworkReceive/TransmitThroughput
- AWS Trusted Advisor: Idle DB instances.

## Optimizations

- Right-scaling (CPU, memory, network)
- Use Reserved Instances (long commitment, constant workloads)
- Stop/start DB instance/cluster(intermittent load)
- Tune SQL queries to avoid additional rows processing and high resource utilization for SQL execution
- Use Aurora Serverless (variable/intermittent load)

# RDS storage: cost optimization strategies



## Cost

- Depending on the storage type provisioned - gp2, gp3, io1
- General Purpose (GP2) - Cost based on the amount of storage provisioned
- Provisioned IOPS – Cost based on the amount of storage provisioned and IOPS separately.

## Cost Monitoring

- CloudWatch metrics: FreeStorageSpace, Read/Write IOPS, Read/Write Latency, DiskQueueDepth

## Optimizations

- Avoid unused/unnecessary secondary indexes
- Implement data life cycle management policies by periodically archiving old data into Amazon S3
- Consider logical export and import using native tools into new RDS DB instance to reduce storage usage



# Aurora storage: cost optimization strategies

- Applies for one copy of data only
- Storage consumption is billed in per GB-month increments
- Storage and I/O costs are decoupled and charged separately
- Storage grows and shrinks automatically\*
- No pre-provisioning of storage, grows in 10 GB increments

## GB-month calculation

Day 1 of month: 1000 GB

Day 2 of month: 20 GB

Day 3 of month: 20 GB

-----

Total monthly Aurora storage usage : GB-month:  $(1,000 \text{ GB} * 30 \text{ days} + 20 \text{ GB} * 29 \text{ days} + 20 \text{ GB} * 28 \text{ days} + \dots + 20 \text{ GB} * 1 \text{ day}) / 30 \text{ days} = 1,290 \text{ GB-month}$ .

Total monthly Aurora cost in USE1:  $1,290 \text{ GB-month} * \$0.10 \text{ per GB-month} = \$129$

## Cost monitoring

[Billed] Volume Bytes Used (GiB)

# Optimizing Aurora storage cost

- Partition the large tables
- Drop older partitions or unused table
- Analyze the need to storing data in Aurora
- Data archiving
- Purge duplicate, unused indexes
- Data type consideration
- Aurora storage automatic resize



# Aurora Read I/O cost optimization

## Cost

- Per-page Read I/O Cost: Number of physical pages read from Aurora storage
- Aurora MySQL page size: 16KB
- Aurora PostgreSQL: 8KB

## Cost monitoring:

- CloudWatch Metrics: [Billed] Volume Read IOPS (Count)
- Use Performance Insights to identify I/O intensive queries

## Optimizing Aurora Read I/O Cost

- Optimize queries to read from memory as much as possible
- Monitor CloudWatch metrics BufferCacheHitRatio (Percent)
- Optimize and right-size your DB instance
- Tune queries to avoid full table scans on large tables
- Use Aurora native backup and snapshots when possible. Logical backup will cause excessive reads.

# Aurora Write I/O Cost

## Cost

- Write I/O Unit: 4 KB unit
- Write I/O Cost : Per 1 million requests
- Example: For us-east-1 AWS region I/O rate: \$0.20 per 1 million requests
- No cost for transaction log record streaming and page updates on all the read replica instances of the cluster

## Cost monitoring

- CloudWatch Metrics: [Billed] Volume Write IOPS (Count)

## Optimizing Aurora Write I/O Cost

- Aurora groups concurrent writes of 4KB against the same pages/segments
- Write IO usage is optimized is built-in process
- Remove un-used or duplicate indexes from tables
- Logical replication – Use when needed

# Aurora: Features and cost

## Aurora Global Database

- Billed for replicated write I/O between regions
- Instances, storage, backup storage and cross-region data transfer
- Optimize by choosing appropriate number of regions, replicas and headless cluster

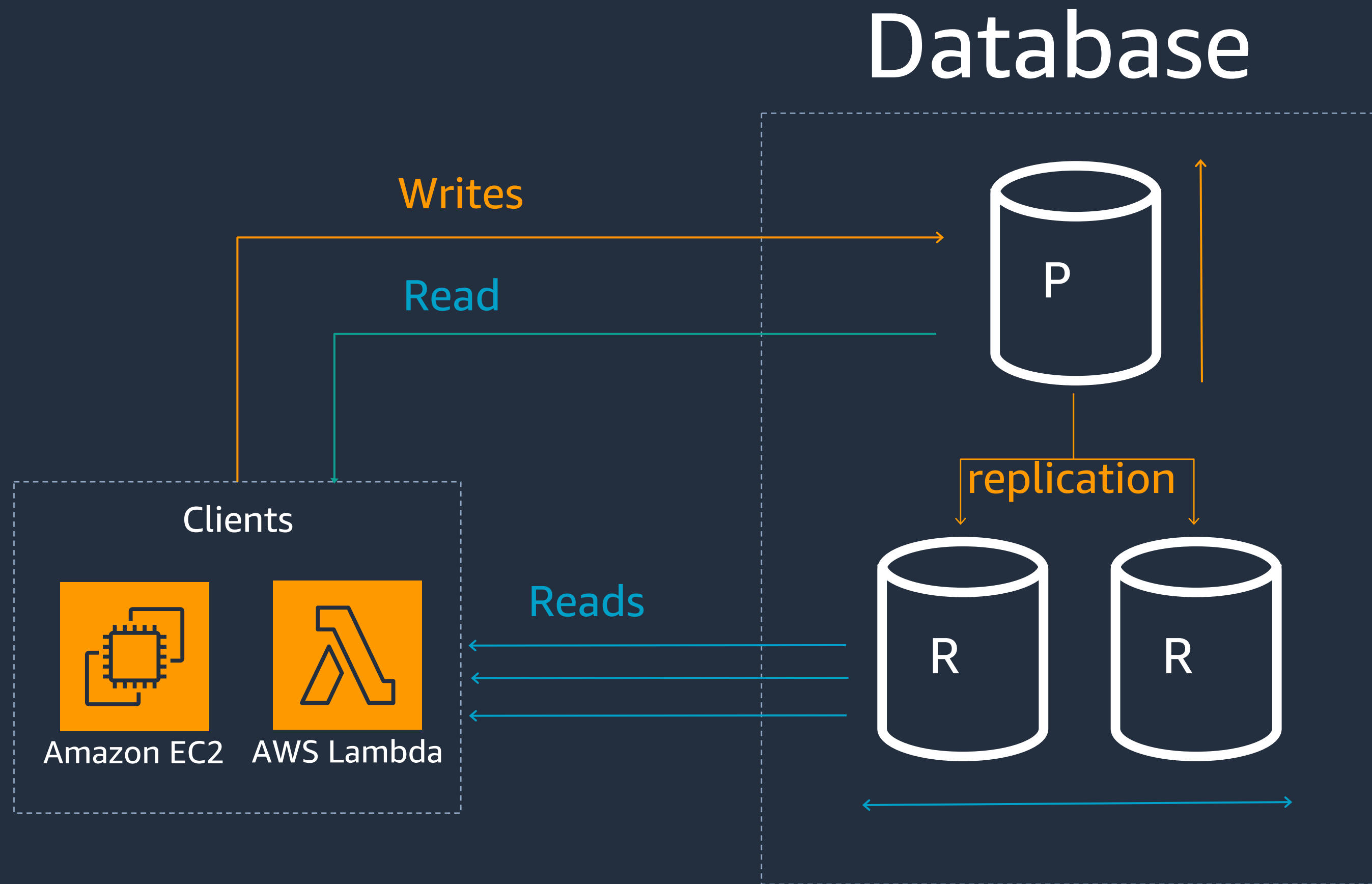
## Aurora Fast Clones

- No additional storage cost initially
- Instances, backup storage, cost applies
- When not in use , drop the clone
- Aurora headless clone cluster

## Snapshot Export to S3

- Every export of data from the same snapshot is billed at full snapshot size, even while exporting selective database or tables.
- Billed per GB of snapshot size (\$.010 in US-EAST). Example: Export of 100GB snapshot will cost  $100 \times \$ 0.01 = \$1$ . Storage, encryption and PUT requests are charged by S3 separately.
- Optimize by dropping unneeded snapshots exports.

# Relational database scaling



## DB Scale Vertically

- Increased memory
- Increased CPU count
- Increased network capacity

## DB Scale Horizontally

- Increased read capacity

## Drawbacks

- Vertical scaling is limited
- Unnecessary data on replicas
- Impacted by disk-based latency
- Costly & limited in scope

# Financial advantages of caching RDS/Aurora

*"Save up to 55% in cost and gain up to 80x faster read performance using ElastiCache with RDS for MySQL (vs. RDS for MySQL alone)."*



- Significantly cheaper than scaling RDS for the same workload
- Only pay a per instance fee and no separate IO charge
- It can scale independently from RDS at a lower cost
- With ElastiCache optimizations such as Enhanced IO multiplexing, obtain up to 72% better price/performance, and with data tiering, obtain up to 60% better price/capacity

\* Internal tests. Can share analysis and customize to customer workloads, instance types.

# Recent In-Memory Price/Performance Optimizations

- Data tiering for ElastiCache and MemoryDB for Redis with up to 60% cost savings versus maximum utilization on R6g nodes
- ElastiCache for Redis 7 including enhanced I/O multiplexing with up to 72% increased throughput and up to 71% reduction in query latency
- MemoryDB enhanced I/O multiplexing with up to 46% increased throughput and up to 21% reduction in latency

\* Internal tests. Can share analysis and customize to customer workloads, instance types.



# What Workloads can be cost-optimized with Caching

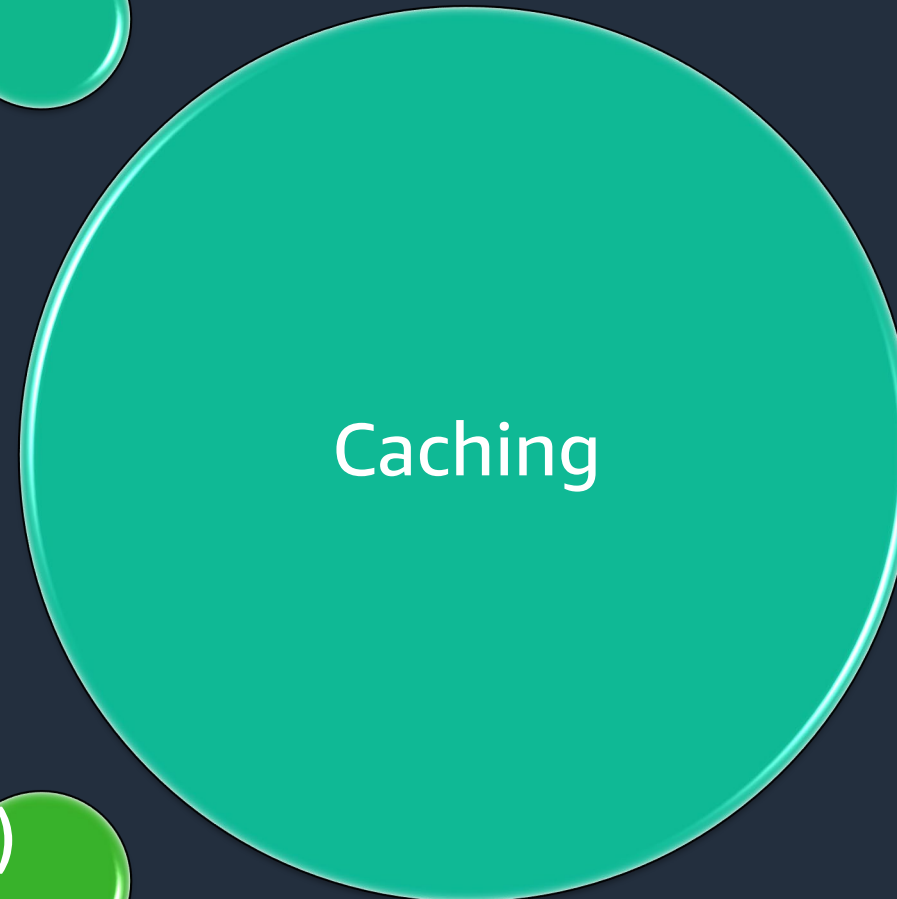
I/O bound and Spiky workloads (read-heavy SQL applications – 80:20 to 60:40 read:write ratio)



Workloads with in-Memory data processing and retrieval of large volumes of data



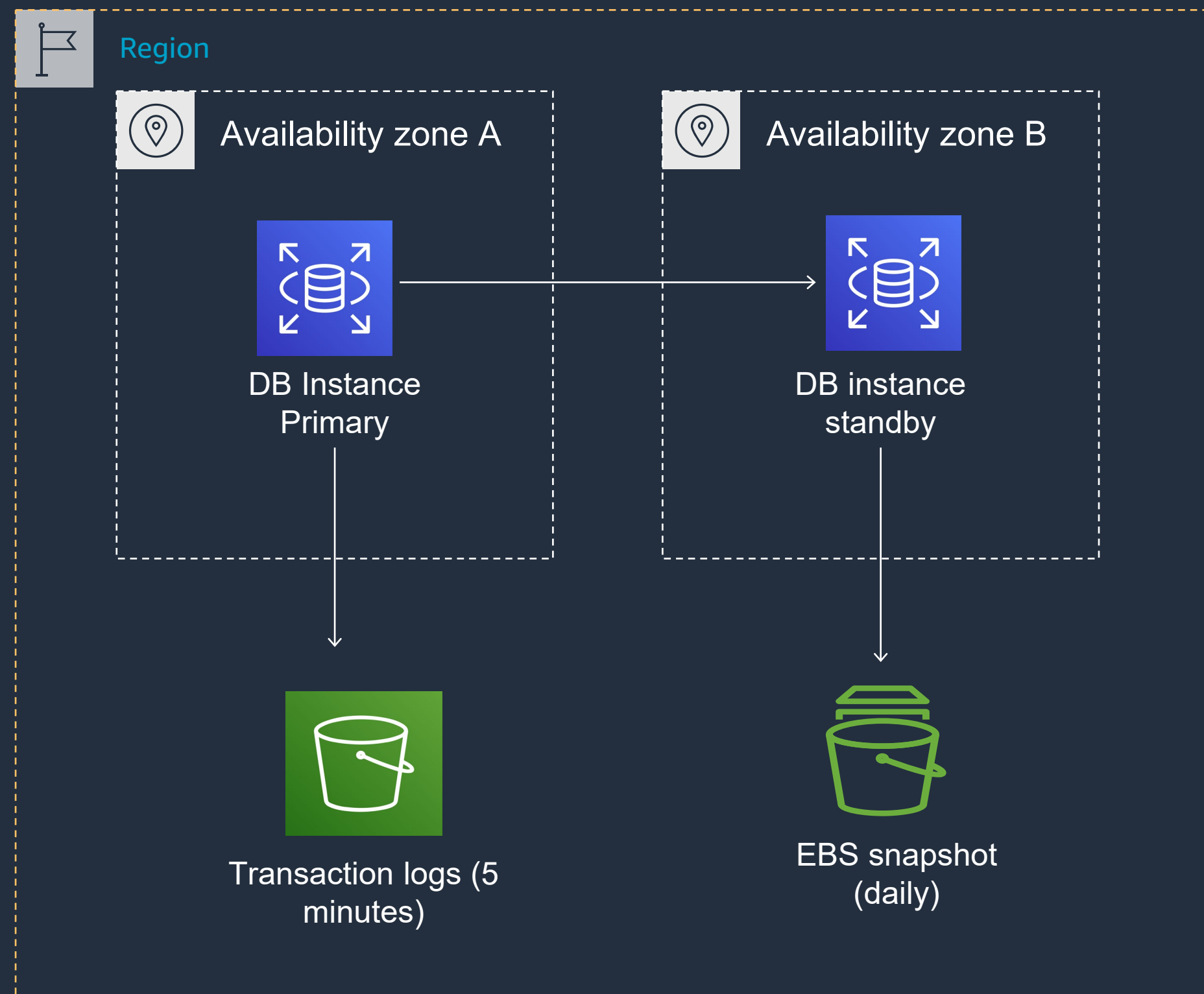
Workloads with Real-time data access (session stores, Gaming, ML Feature Stores)



Improved Application latency

Improved Database Costs

# Backup and Snapshots: cost optimization strategies



## Cost

- RDS/Aurora Backup cost: Backup storage per GB-month

## Notes

- No charges for backup up to 100% of total RDS/Aurora database storage for a region
- The first snapshot of a DB instance is full and subsequent snapshots are incremental
- Manual snapshots that fall within the retention period are not charged
- Manual snapshots retained more than retention period charged

## Optimizations

- Review and configure the backup retention period based on business SLAs
- Use and maintain the manual snapshots as needed
- Identify orphan snapshots with no provisioned instance


# AWS Pricing Calculator

- <https://calculator.aws/#/>
- Web-based planning tool that can use to create estimates for your AWS use case
- Model your solutions before building them, explore the AWS service price point
- Review the calculations behind your estimates
- Plan how you spend, find cost saving opportunities

The screenshot shows the AWS Pricing Calculator interface for configuring Amazon Aurora MySQL-Compatible instances. The page title is "Configure Amazon Aurora MySQL-Compatible" with an "Info" link. The "Description" field contains the placeholder text "Enter a description for your estimate". The "Region" is set to "US East (N. Virginia)". Two instance types are available: "Aurora MySQL-Compatible" (selected) and "Aurora Serverless MySQL-Compatible". The "Aurora MySQL-Compatible" instance is selected, and the "Nodes" field is set to "1". The "Selected Instance" is "db.t2.medium" with specifications: vCPU: 2, Memory: 4 GiB, Network Performance: Low to Moderate.

# RDS: Tagging

## Add tags ✕

Add tags to your RDS resources to organize and track your Amazon RDS costs. [Learn more](#) 

Tag key	Value
<input type="text" value="DBEnv"/>	<input type="text" value="Test"/>

# Activate custom tags in the Billing console

The screenshot shows the AWS Billing console interface. The left navigation menu includes links for Home, Billing, Bills, Payments, Credits, Purchase orders, Cost & usage reports, Cost categories, Cost allocation tags (highlighted with a blue arrow and '1'), Free tier, Billing Conductor, Cost Management, and Cost explorer. The main content area is titled 'Cost allocation tags' and has a 'Download CSV' button. Below this, there are two tabs: 'User-defined cost allocation tags' (selected) and 'AWS generated cost allocation tags'. The 'User-defined cost allocation tags' section shows a search bar and a dropdown for 'All statuses'. A table lists the tags:

Tag key	Status
<input type="checkbox"/> auto_delete	Inactive
<input type="checkbox"/> auto_stop	Inactive
<input type="checkbox"/> auto_delete	Inactivation

At the top right of the table area, there are buttons for 'Undo', 'Deactivate', and 'Activate'. A blue arrow with the number '3' points to the 'Activate' button. The 'auto\_delete' tag in the first row is highlighted with a blue arrow and the number '2'.

# Cost allocation tags for Aurora storage

- Categorize resources such as storage, I/O, backups, snapshots, Aurora Backtrack, and Global Database replicated I/O
- Based on the purpose, owner, environment
- Usage categorization based on DB environments like production, test & dev etc
- Usage resources like storage, I/O, backups, snapshots, Aurora Backtrack, and Global Database replicated I/O etc

▶ GuardDuty		\$0.18
▶ Key Management Service		\$0.00
▶ Pinpoint		\$0.50
▼ Relational Database Service		<del>\$49.64</del>
▼ US East (N. Virginia)		<del>\$12.54</del>
Amazon Aurora Storage and I/O		\$1.92
\$0.10 per GB-month of consumed storage (Aurora)	0.671 GB-Mo	<del>\$0.07</del>
\$0.20 per 1 million I/O requests (Aurora)	9,268,227.000 IOs	<del>\$1.85</del>
Amazon Relational Database Service for Aurora MySQL		<del>\$1.89</del>
\$0.29 per RDS db.r4.large instance hour (or partial hour) running Amazon Aurora	164.190 Hrs	<del>\$47.18</del>
▶ Secrets Manager		\$1.20
▶ Simple Notification Service		\$0.00



# Cost Explorer

**AWS Cost Management**

- Home
- Cost Explorer**
- Reports
- Budgets
- Cost Anomaly Detection
- Rightsizing recommendations
- Savings Plans**
- Overview
- Inventory
- Recommendations
- Purchase Savings Plans
- Utilization report
- Coverage report
- Cart 0
- Reservations**
- Overview
- Recommendations
- Utilization report
- Coverage report
- Preferences
- Billing Console [↗](#)
- Documentation [↗](#)

AWS Cost Management > Cost Explorer

[Save](#) [Save as...](#) [Recent reports](#) [New report](#)

cost\_explorer\_report [✎](#)

Jan 02, 2022 - Jan 24, 2022 Monthly Stack

Group by: Usage Type | Service | Linked Account | Region | Instance Type | Resource | Cost Category | Tag | More

Costs (\$ in thousands)

Jan 2022\*

RDS:Multi-AZ-PIOPS Multi-AZUsage:db.r5.8xl InstanceUsage:db.r5.2xl USE1-NodeUsage:db.r6g.large  
RDS:PIOPS Others

To see usage data, filter by "Usage Type" or "Usage Type Group" filters with matching units (e.g., hours).

[Download CSV](#)

Usage Type	Jan 2022*	Usage Type Total
Total cost (\$)	13,559.78	13,559.78
RDS:Multi-AZ-PIOPS (\$)	2,847.31	2,847.31
Multi-AZUsage:db.r5.8xl (\$)	2,031.7	2,031.7

cost\_explorer\_report [✎](#)

Jan 02, 2022 - Jan 24, 2022 Monthly Stack

Group by: Usage Type | Service | Linked Account | Region | Instance Type | Resource | Cost Category | Tag | More

Costs (\$)

Jan 2022\*

InstanceUsage:db.r5.2xl InstanceUsage:db.r5.large USE1-RDS:ProxyUsage Aurora:StorageIOUsage  
Aurora:StorageUsage Aurora:BackupUsage

To see usage data, filter by "Usage Type" or "Usage Type Group" filters with matching units (e.g., hours).

[Download CSV](#)

Usage Type	Jan 2022*	Usage Type Total
Total cost (\$)	141.78	141.78
InstanceUsage:db... (\$)	117.26	117.26
InstanceUsage:db... (\$)	8.97	8.97
USE1-RDS:ProxyUsage (\$)	8.35	8.35
Aurora:StorageIO... (\$)	7.01	7.01
Aurora:StorageUsage (\$)	0.20	0.20
Aurora:BackupUsage (\$)	0.00	0.00

FILTERS [CLEAR ALL](#)

Service [Include all](#)

Linked Account [Include all](#)

Region [Include all](#)

Instance Type [Include all](#)

Usage Type [Include all](#)

Usage Type Group [Include all](#)

Resource [Include all](#)

Cost Category [Include all](#)

Tag [Include all](#)

**CostTest** [Include only](#)

AuroraPG AuroraPG1 3

API Operation [Include all](#)

Charge Type [Include all](#)

Availability Zone [Include all](#)

Platform [Include all](#)

Purchase Option [Include all](#)

Tenancy [Include all](#)

Database Engine [Include all](#)

Legal Entity [Include all](#)

Billing Entity [Include all](#)

[Show less](#)

ADVANCED OPTIONS [ⓘ](#)

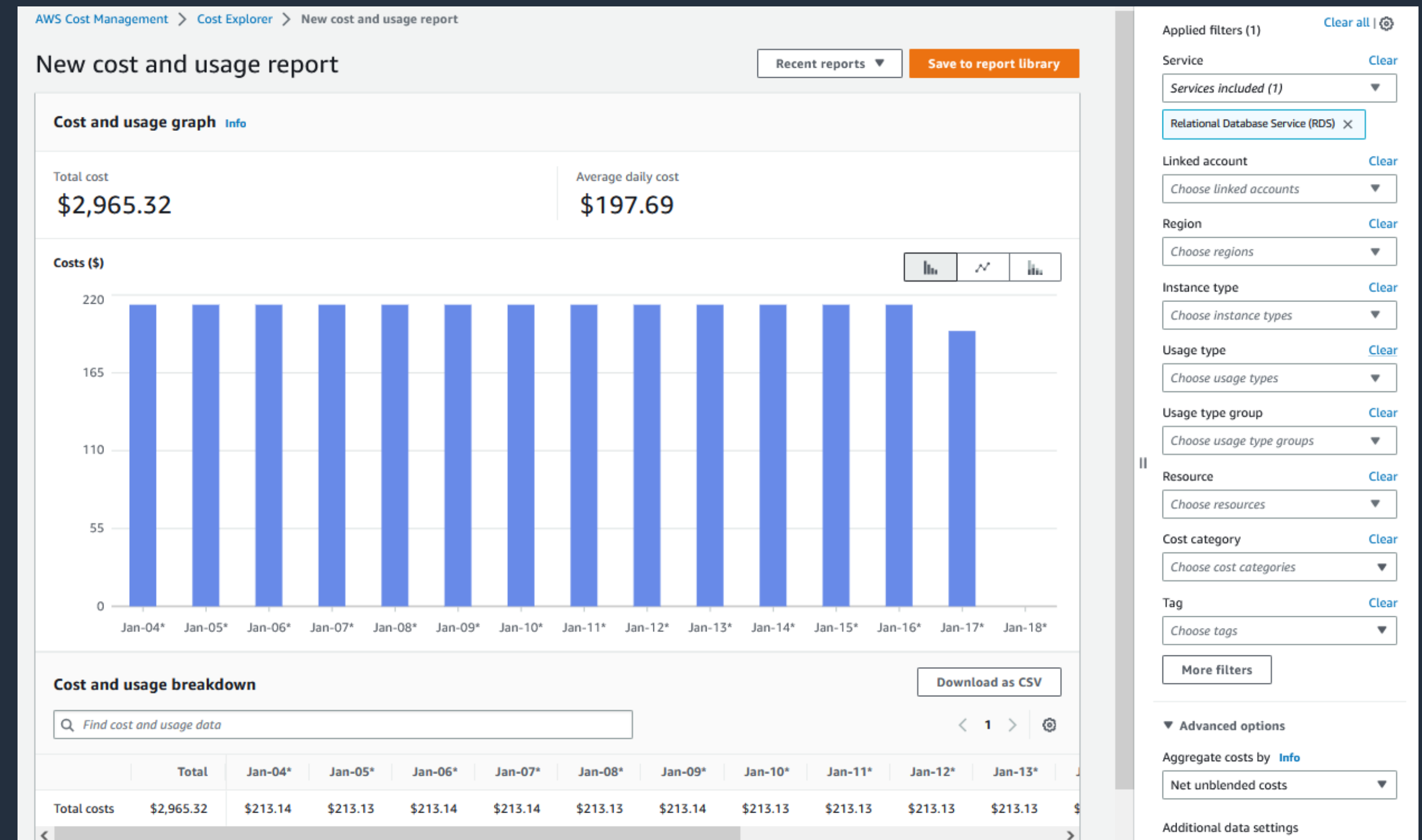
Show costs as [ⓘ](#)

Net unblended costs

Include costs related to [ⓘ](#)

# AWS Cost and Usage Report (CUR)

- Comprehensive set of cost and usage data, and estimated cost.
- Publish your AWS billing reports to an Amazon S3 bucket
- Break down your costs by the hour, day, or month, by product or product resource, or by tags
- Use Amazon Athena or Amazon QuickSight to analyze CUR data in S3
- Upload CUR reports to Amazon Redshift to analyze AWS cost and usage.



# Call to action

- Understand key cost components: cost calculations, monitoring and optimization
- Setup an alert to monitor abnormal changes in critical cost components
- Analyze top 10 intensive read IO queries and tune to reduce query overhead
- Use Aurora/RDS tagging on RDS databases and Aurora clusters for cost categorization and usages analysis
- Frequently review the cost usage report and plan for optimization
- Reach out to Database Specialist TAM for additional help

# More Questions?

## Office Hours

Your questions answered: Best practices for AWS database cost optimizations.

Thursday, 07/27/23 | 8:00 AM–8:45 AM PT

# Resources

## Cost explorer

- <https://aws.amazon.com/aws-cost-management/aws-cost-explorer/>

## AWS Pricing calculator

- <https://calculator.aws/>

## Planning Aurora I/O cost

- <https://aws.amazon.com/blogs/database/planning-i-o-in-amazon-aurora/>

## Planning Aurora I/O cost

- <https://docs.aws.amazon.com/cur/latest/userguide/what-is-cur.html>

## Aurora pricing page

- <https://aws.amazon.com/rds/aurora/pricing/>

# Thank You!