Amazon Relational Database Service (Amazon RDS)
Managed relational database service with a choice of popular database engines

- **Easy to administer**: Easily deploy and maintain hardware, OS, and DB software; built-in monitoring
- **Performant & scalable**: Scale compute and storage with a few clicks; minimal downtime for your application
- **Available & durable**: Automatic Multi-AZ data replication; automated backup, snapshots, and failover
- **Secure & compliant**: Data encryption at rest and in transit; industry compliance and assurance programs

Amazon Aurora, MySQL, MariaDB, PostgreSQL, Microsoft SQL Server, Oracle
<table>
<thead>
<tr>
<th>Commercial</th>
<th>Open source</th>
<th>Cloud native</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORACLE</td>
<td><strong>MySQL</strong></td>
<td><strong>Amazon Aurora</strong></td>
</tr>
<tr>
<td>Microsoft SQL Server</td>
<td><strong>MariaDB</strong></td>
<td><strong>MySQL compatible</strong></td>
</tr>
<tr>
<td><strong>Amazon Elastic Block Store (Amazon EBS)-based storage</strong></td>
<td><strong>PostgreSQL</strong></td>
<td><strong>PostgreSQL compatible</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Amazon Aurora storage system</strong></td>
</tr>
</tbody>
</table>
## Amazon RDS database engines

<table>
<thead>
<tr>
<th>Commercial</th>
<th>Open source</th>
<th>Cloud native</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORACLE</td>
<td>MySQL</td>
<td>Amazon Aurora</td>
</tr>
<tr>
<td>Microsoft SQL Server</td>
<td>MariaDB</td>
<td>MySQL compatible</td>
</tr>
<tr>
<td></td>
<td>PostgreSQL</td>
<td>PostgreSQL compatible</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Amazon Aurora storage system</td>
</tr>
</tbody>
</table>

- **Amazon Elastic Block Store (Amazon EBS)-based storage**
- **MySQL**
- **MariaDB**
- **PostgreSQL**
- **MySQL compatible**
- **PostgreSQL compatible**
Agenda

• Why run MySQL
• Why run managed MySQL on Amazon RDS
• Why Aurora MySQL?
Why run MySQL?
1. Popular

2. Innovative

3. Flexible

Image credit: By Mackphillips - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=55946550
1. Popular

2. Innovative

3. Flexible

Image credit: By Mackphillips - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=55946550
MySQL is the world’s most popular database

All respondents

- MySQL: 58.7%
- SQL Server: 41.2%
- PostgreSQL: 32.9%
- MongoDB: 25.9%
- SQLite: 19.7%
- Redis: 18.0%
- Elasticsearch: 14.1%
- MariaDB: 13.4%
- Oracle: 11.1%

Source: Stack Overflow Developer Survey Results 2018 (https://insights.stackoverflow.com/survey/2018/#technology)
“Most popular” buys you . . .

- Highly exercised, **stable** code
- Large **ecosystem** of ISVs, Tools, Implementation and Support Partners
- Large **community** of users and community-driven resources and a **larger DBA talent pool**
1. Popular

2. Innovative

3. Flexible
MySQL 8.0 highlights - FUNCTIONALITY

- Common Table Expressions
- Window functions
- JSON improvements
- 5108 Spatial Reference Systems
- utf8mb4
MySQL 8.0 highlights - AVAILABILITY

- Instant ADD COLUMN
- Unified, transactional data dictionary
- Crash-safe, atomic DDL
MySQL 8.0 highlights - PERFORMANCE

- Hot-spot management
- Descending indexes
- Invisible indexes
- Improved optimizer cost model
- Resource Groups
- Improved replication
MySQL 8.0 highlights – SECURITY, MANAGEABILITY

• Roles
• Password strength
• Open SSL as default TLS/SSL library
• Enhanced observability
MariaDB 10.3 highlights

- Oracle compatibility
  - PL/SQL compatibility parser
  - Sequences
  - INTERSECT and EXCEPT to complement UNION
  - New ROW type and TYPE OF stored functions
  - Invisible Columns
  - Cursor with parameters

- Temporal data processing

- User-defined aggregates

- Instant ADD COLUMN for InnoDB

Image credit: By Mackphillips - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=55946550
MariaDB 10.4 highlights (Release Candidate)

- SQL Server compatibility
  - `sql_mode='mssql'`
  - Subset of Microsoft SQL Server's language

- Password expiry and account locking

- Instant DROP COLUMN for InnoDB
1. Popular

2. Innovative

3. Flexible

Image credit: By Mackphillips - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=55946550
Why run managed MySQL on Amazon RDS?
1. Popular
2. Innovative
3. Flexible
Hundreds of thousands of customers
Popular buys you . . .

Highly exercised, stable code

Unrivalled operational excellence
Who can you trust?

Automated remediation

World’s most experienced operators

Automation
1. Popular
2. Innovative
3. Flexible
Amazon RDS highlights - AVAILABILITY

- Automated, 0-RPO failover across AZs
- Managed x-region replicas for DR
- Automated backups, manual snapshots
- Point-in-time recovery
Automated, 0-RPO failover across AZs

Each host manages set of EBS volumes with a full copy of the data

Instances are monitored by an external observer to maintain consensus over quorum
Automated, 0-RPO failover across AZs

Each host manages set of EBS volumes with a full copy of the data

Instances are monitored by an external observer to maintain consensus over quorum

Failover initiated by automation or through RDS API
Automated, 0-RPO failover across AZs

Each host manages set of EBS volumes with a full copy of the data

Instances are monitored by an external observer to maintain consensus over quorum

Failover initiated by automation or through RDS API

Redirection to the new primary instance is provided through DNS
Automated, 0-RPO failover across AZs

Each host manages set of EBS volumes with a full copy of the data

Instances are monitored by an external observer to maintain consensus over quorum

Failover initiated by automation or through RDS API

Redirection to the new primary instance is provided through DNS
Read Scaling with Read Replicas

Use Amazon RDS read replicas to relieve pressure on your source database with additional read capacity

Create up to five replicas per source database

Monitor replication lag in Amazon CloudWatch or Amazon RDS console
Planning for Disaster Recovery

Use a cross-region read replica as a standby database for recovery in the event of a disaster

Read replicas can be configured for Multi-AZ to reduce recovery time

Can use delayed replication for MySQL to protect from self-inflicted disasters
Backups, Snapshots, and Point-in-time restore

Two options – automated backups and manual snapshots

EBS snapshots stored in Amazon S3

Transaction logs stored every 5 minutes in S3 to support Point in Time Recovery

No performance penalty for backups

Snapshots can be copied across regions or shared with other accounts
New Amazon RDS backup features

Retain automated backups **NEW!**

Automated backups are retained for the retention period for the instance

Optionally keep automated backups and transaction logs upon instance deletion

Specify parameter group value on restore **NEW!**

Incremental encrypted snapshot copy **NEW!**
Amazon RDS highlights – SECURITY, MANAGEABILITY

- IAM DB Authentication
- Automated OS and database upgrades
- Push-button scaling
- Managed binlog replication
- Log upload to CloudWatch Logs
- Industry compliance
- Per-second billing NEW!
Recommendations

Example issues:
Engine version outdated, Pending maintenance available, Automated backups disabled, Enhanced Monitoring disabled, Encryption disabled

Parameter recommendations: NEW!
Non-default custom memory parameters, Change buffering enabled, Logging to table

Aurora cluster recommendations NEW!
Start and Stop

Solution for development and test environments

Stop and start a running database instance from the console or AWS Command Line Interface (AWS CLI)

Now available for both single-AZ and Multi-AZ DB instances and Aurora DB clusters

While instance is stopped, you only pay for storage
Backup retention window is maintained while stopped

Instances are restarted after seven days
Amazon RDS highlights – PERFORMANCE

- R5, M5, and T3 database instance family NEW!
- Elastic volumes up to 64 TB  NEW!
- Up to 80K Provisioned IOPS  NEW!
Performance Insights

- Measures DB Load
- Identifies bottlenecks (top SQL, wait events)
- Adjustable time frame (hour, day, week, longer)
New features in RDS Performance Insights

- Engine support: MySQL, MariaDB **NEW!**
- Extended data retention **NEW!**
  - Retain up to two years of performance data
  - Trend performance over time, analyze month-over-month activity, and compare end-of-quarter or end-of-year performance with earlier performance
- Load metrics in CloudWatch **NEW!**
  - DBLoad
  - DBLoadCPU
  - DBLoadNonCPU
- AWS CloudFormation support
Monitor RDS with CloudWatch

- Amazon CloudWatch metrics
  - CPU/Storage/Memory
  - Swap usage
  - I/O (read and write)
  - Latency (read and write)
  - Throughput (read and write)
  - Replica lag
- Amazon CloudWatch alarms
  - Similar to on-premises monitoring tools
- Enhanced monitoring
  - Access to additional CPU, memory, file system, and disk I/O metrics
  - As low as one-second intervals
- Integration with third-party monitoring tools
Database activity monitoring and insights

- **Search**: Look for specific events across log files.
- **Metrics**: Measure activity in your Aurora DB cluster.
- **Visualizations**: Create activity dashboards
- **Alarms**: Get notified or take actions

- Continuously monitor activity in your DB clusters by sending audit logs to CloudWatch Logs.
- Export to S3 for long term archival; analyze logs using Athena; visualize logs with QuickSight.

Amazon CloudWatch → S3 → Amazon Athena ← Amazon QuickSight
1. Popular

2. Innovative

3. Flexible

Amazon RDS

Image credit: By Mackphillips - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=55946550
Why Aurora?
Amazon Aurora…
Enterprise database at open source price

Delivered as a **managed** service

- **Speed** and **availability** of high-end commercial databases
- **Simplicity** and **cost-effectiveness** of open source databases
- Drop-in **compatibility** with MySQL and PostgreSQL

**Simple pay as you go** pricing
Amazon Aurora innovations
Re-imagining databases for the cloud

1. Scale-out, distributed, multi-tenant design
2. Service-oriented architecture leveraging AWS services
3. Automate administrative tasks – fully managed service
Scale-out, distributed architecture

Purpose-built log-structured distributed storage system designed for databases

Storage volume is striped across hundreds of storage nodes distributed over 3 different availability zones

Six copies of data, two copies in each availability zone to protect against AZ+1 failures

Plan to apply same principles to other layers of the stack
Leveraging AWS services

- Invoke AWS Lambda events from stored procedures/triggers
- Load data from Amazon Simple Storage Service (Amazon S3), store snapshots and backups in S3
- Use AWS Identity and Access Management (IAM) roles to manage database access control
- Upload systems metrics and audit logs to CloudWatch
1. Popular

2. Innovative

3. Flexible

Image credit: By Mackphillips - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=55946550
Aurora customer adoption

Fastest growing service in AWS history

Aurora is used by $\frac{3}{4}$ of the top 100 AWS customers
Who is moving to Aurora and why?

Customers using open source engines

• Higher performance – up to 5x
• Better availability and durability
• Reduces cost – up to 60%
• Easy migration; no application change

Customers using commercial engines

• One tenth of the cost; no licenses
• Integration with cloud ecosystem
• Comparable performance and availability
• Migration tooling and services
1. Popular

2. Innovative

3. Flexible
Enterprise-grade Performance and Scalability
Write and read throughput
Aurora MySQL is 5x faster than MySQL

Using Sysbench with 250 tables and 200,000 rows per table on R4.16XL
Bulk data load performance

Aurora MySQL loads data 2.5x faster than MySQL

10 Sysbench Tables, 10MM rows per each
How did we achieve this?

Do less work
• Do fewer IOs
• Minimize network packets
• Cache prior results
• Offload the database engine

Be more efficient
• Process asynchronously
• Reduce latency path
• Use lock-free data structures
• Batch operations together

• Databases are all about I/O
• Network-attached storage is all about packets/second
• High-throughput processing is all about context switches
Aurora I/O profile

MySQL with Replica

MySQL I/O profile for 30 min Sysbench run
780K transactions
7,388K I/Os per million txns (excludes mirroring, standby)
Average 7.4 I/Os per transaction

Amazon Aurora

Aurora IO profile for 30 min Sysbench run
27,378K transactions – 35X MORE
0.95 I/Os per transaction (6X amplification) – 7.7X LESS
Aurora lock management

- Same locking semantics as MySQL
- Concurrent access to lock chains

MySQL lock manager

Scan ➔ Delete ➔ Insert ➔ Scan

Aurora lock manager

Scan ➔ Delete ➔ Insert ➔ Scan

- Concurrent access to lock chain and lock manager and to update simultaneously.
- Lock-free deadlock detection
Instant crash recovery

Traditional database
Have to replay logs since the last checkpoint
Typically 5 minutes between checkpoints
Single-threaded in MySQL; requires a large number of disk accesses

Amazon Aurora
Underlying storage replays redo records on demand as part of a disk read
Parallel, distributed, asynchronous
No replay for startup
When Database fails – recovery is fast <30 seconds

0 - 5s – 30% of fail-overs

10 - 20s – 25% of fail-overs

5 - 10s – 40% of fail-overs

20 - 30s – 5% of fail-overs
Recent Innovations
Aurora MySQL

Performance
• Parallel Query

Availability
• Global Database, Backtrack

Manageability
• Serverless, Cluster start/stop, DB Log upload to CloudWatch Logs, Synchronous Lambda calls, Custom endpoints

Security
• Encrypted self-managed MySQL to Aurora migration
Parallel query processing

Aurora storage has thousands of CPUs
• Opportunity to push down and parallelize query processing
• Moving processing close to data reduces network traffic and latency

However, there are significant challenges
• Data is not range partitioned – require full scans
• Data may be in-flight
• Read views may not allow viewing most recent data
• Not all functions can be pushed down

We were able to test Aurora’s parallel query feature and the performance gains were very good. To be specific, we were able to reduce the instance type from r3.8xlarge to r3.2xlarge. For this use-case, parallel query was a great win for us.

Jyoti Shandil, Cloud Data Architect
Aurora Global Database

High throughput: Up to 200K writes/sec – negligible performance impact

Low replica lag: < 1 sec cross-country replica lag under heavy load

Fast recovery: < 1 min to accept full read-write workloads after region failure

Database Backtrack

Backtrack brings the database to a point in time without requiring restore from backups

- Backtracking from an unintentional DML or DDL operation
- Backtrack is not destructive. You can backtrack multiple times to find the right point in time
- Also useful for QA (rewind your DB between test runs)

Fast database cloning

Create a copy of a database without duplicate storage costs

• Creation of a clone is nearly instantaneous – we don’t copy data
• Data copy happens only on write – when original and cloned volume data differ

Typical use cases:

• Clone a production DB to run tests
• Reorganize a database
• Save a point in time snapshot for analysis without impacting production system.

Aurora Serverless

- Starts up on demand, shuts down when not in use
- Scales up/down automatically
- No application impact when scaling
- Pay per second, 1 minute minimum

https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/aurora-serverless.html
Amazon RDS Data API for serverless applications

Access through simple web interface
- Public endpoint addressable from anywhere
- No client configuration required
- No persistent connections required

Ideal for Serverless applications (Lambda)
Ideal for light-weight applications (IOT)

1. Popular

2. Innovative

3. Flexible
“All we have to see
Is I don’t belong to you
And you don’t belong to me.”

George Michael
Freedom! ’90
Migrating to Aurora
<table>
<thead>
<tr>
<th>Source database</th>
<th>From where</th>
<th>Recommended option</th>
</tr>
</thead>
<tbody>
<tr>
<td>MySQL, PostgreSQL</td>
<td>RDS</td>
<td>Console-based automated snapshot ingestion and catch up via binlog replication.</td>
</tr>
<tr>
<td>MySQL, MariaDB</td>
<td>EC2, on premises</td>
<td>Binary snapshot ingestion through S3 and catch up via binlog replication.</td>
</tr>
<tr>
<td>PERcona</td>
<td>EC2, on premises, RDS</td>
<td>Schema conversion using SCT and data migration via DMS.</td>
</tr>
</tbody>
</table>
Thank you!

Chayan Biswas

cbbiswas@amazon.com