RDS Custom

- Managed Oracle and Microsoft SQL Server Database with OS and database customization
- RDS: Automates setup, operation, and scaling of database in AWS
- Custom: access to the underlying database and OS so you can
  - Configure settings
  - Install patches
  - Enable native features
  - Access the underlying EC2 Instance using SSH or SSM Session Manager
- De-activate Automation Mode to perform your customization, better to take a DB snapshot before
- RDS vs. RDS Custom
  - RDS: entire database and the OS to be managed by AWS
  - RDS Custom: full admin access to the underlying OS and the database

Amazon Aurora

Amazon Aurora is a MySQL and PostgreSQL-compatible relational database offered as a managed service within the Amazon Web Services (AWS) ecosystem. It's designed to deliver high performance, availability, and scalability while reducing the overhead of database management tasks.

Key features of Amazon Aurora include:

1. **Compatibility:** It is compatible with MySQL and PostgreSQL, offering the same functionalities, APIs, drivers, and tools that these databases provide.
2. **Performance:** Amazon Aurora is designed for high performance, delivering faster read and write operations compared to traditional databases. It utilizes a distributed architecture that replicates data across multiple availability zones for enhanced performance and fault tolerance.
3. **Scalability:** Aurora offers scalability by automatically scaling storage capacity up to 64TB without the need for manual intervention. It also provides an auto-scaling feature for compute resources to accommodate fluctuating workloads.
4. **Durability and Availability:** It provides high durability with automatic backups and replication across multiple availability zones, ensuring minimal downtime and data loss.
5. **Security:** Amazon Aurora incorporates various security measures such as encryption at rest and in transit, IAM-based authentication, and fine-grained access control to protect sensitive data.

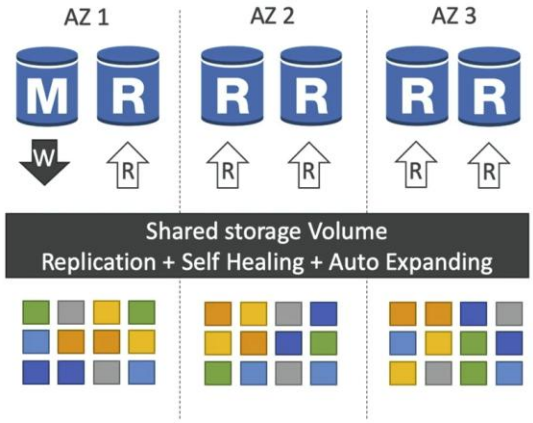
6. **Cost-Effectiveness:** It offers a pay-as-you-go pricing model, allowing users to pay only for the resources used without any upfront costs or long-term commitments.

Amazon Aurora is well-suited for applications requiring a highly performant and scalable relational database. Its compatibility with MySQL and PostgreSQL enables easy migration for existing applications using these databases while providing additional features and improvements in performance and scalability.

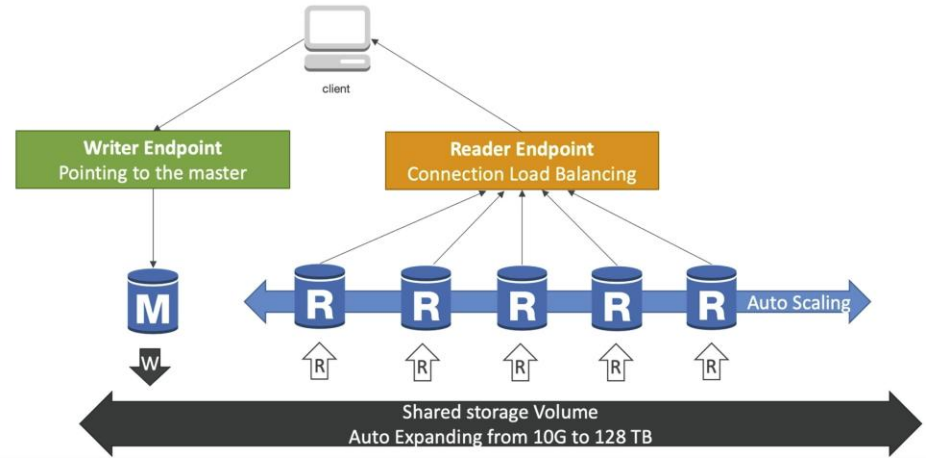
- Aurora is a proprietary technology from AWS (not open sourced)
- Postgres and MySQL are both supported as Aurora DB (that means your drivers will work as if Aurora was a Postgres or MySQL database)
- Aurora is "AWS cloud optimized" and claims 5x performance improvement over MySQL on RDS, over 3x the performance of Postgres on RDS
- Aurora storage automatically grows in increments of 10GB, up to 128 TB.
- Aurora can have up to 15 replicas and the replication process is faster than MySQL (sub 10 ms replica lag)
- Failover in Aurora is instantaneous. It's HA native.
- Aurora costs more than RDS (20% more) – but is more efficient

Amazon High Availability and Read Scaling

- 6 copies of your data across 3 AZ:
  - 4 copies out of 6 needed for writes
  - 3 copies out of 6 need for reads
  - Self healing with peer-to-peer replication
  - Storage is striped across 100s of volumes
- One Aurora Instance takes writes (master)
- Automated failover for master in less than 30 seconds
- Master + up to 15 Aurora Read Replicas serve reads
- Support for Cross Region Replication



Aurora DB Cluster

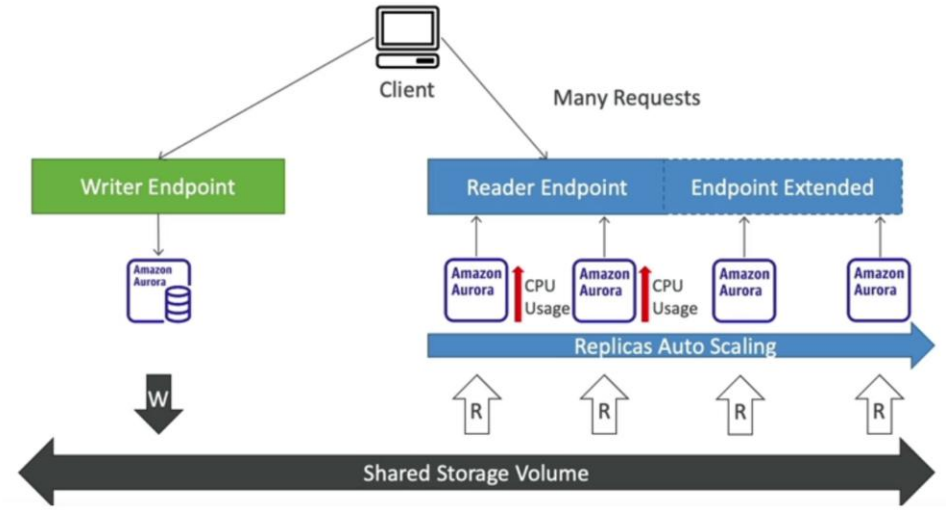


Feature of Aurora

- Automatic fail-over
- Backup and Recovery
- Isolation and security
- Industry compliance
- Push-button scaling
- Automated Patching with Zero Downtime
- Advanced Monitoring
- Routine Maintenance
- Backtrack: restore data at any point of time without using backups

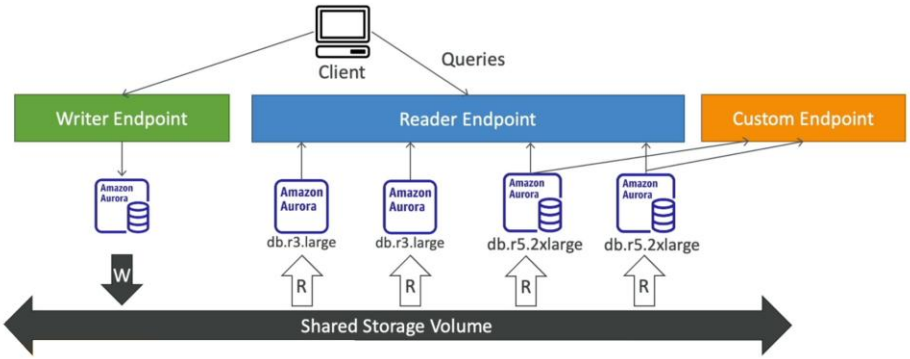
Lab: create an aurora database

Aurora Replicas – Auto Scaling

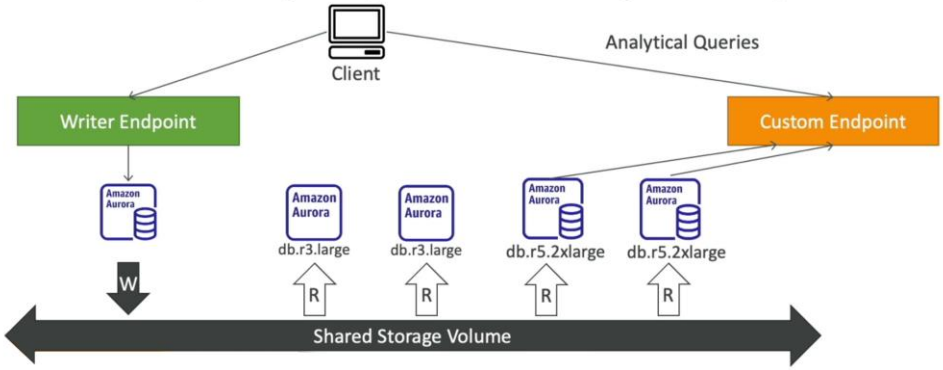


Aurora - Custom Endpoints

- Define a subset of Aurora Instances as a Custom Endpoint

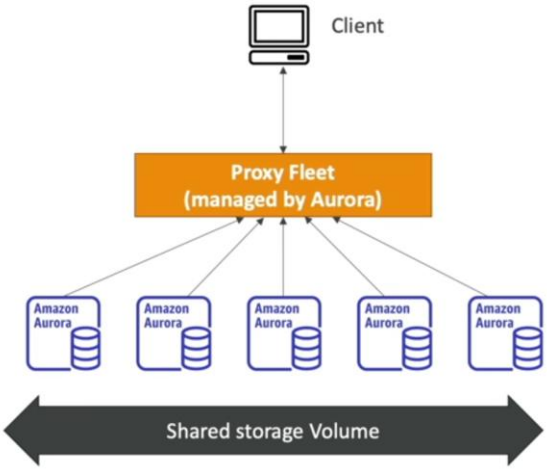


- Define a subset of Aurora Instances as a Custom Endpoint
- Example: Run analytical queries on specific replicas
- The Reader Endpoint is generally not used after defining Custom Endpoints



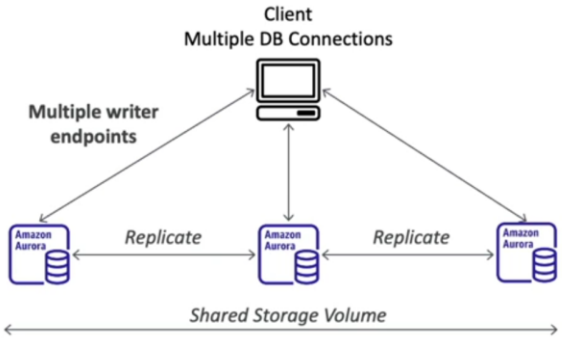
Aurora – Serverless

- Automated database instantiation and auto-scaling based on actual usage
- Good for infrequent, intermittent or unpredictable workloads
- No capacity planning needed
- Pay per second, can be more cost-effective



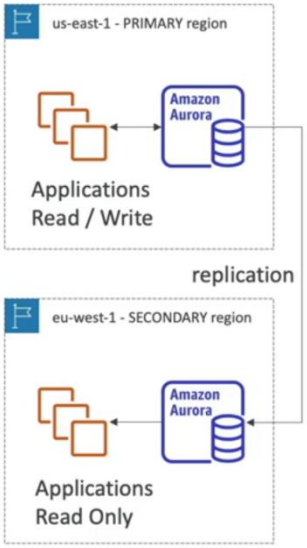
Aurora multi-Master

- In case you want **continuous write availability** for the writer nodes
- Every node does R/W - vs promoting a Read Replica as the new master



Global Aurora- cross region replica

- Aurora Cross Region Read Replicas:
  - Useful for disaster recovery
  - Simple to put in place
- Aurora Global Database (recommended):
  - 1 Primary Region (read / write)
  - Up to 5 secondary (read-only) regions, replication lag is less than 1 second
  - Up to 16 Read Replicas per secondary region
  - Helps for decreasing latency
  - Promoting another region (for disaster recovery) has an RTO of < 1 minute
  - Typical cross-region replication takes less than 1 second



Aurora Machine Learning

- Enables you to add ML-based predictions to your applications via SQL
- Simple, optimized, and secure integration between Aurora and AWS ML services
- Supported services
  - Amazon SageMaker (use with any ML model)
  - Amazon Comprehend (for sentiment analysis)
- You don't need to have ML experience
- Use cases: fraud detection, ads targeting, sentiment analysis, product recommendations

In this image the application send aurora what is the recommended product for a user then Aurora send to Machine learning then Machine send a prediction to the Aurora that then user buy read shirt and blue pants then Aurora send it to the application

RDS Backup

- Automated backups:
  - Daily full backup of the database (during the backup window)
  - Transaction logs are backed-up by RDS every 5 minutes
  - => ability to restore to any point in time (from oldest backup to 5 minutes ago)
  - 1 to 35 days of retention, set 0 to disable automated backups
- Manual DB Snapshots
  - Manually triggered by the user
  - Retention of backup for as long as you want

- Trick: in a stopped RDS database, you will still pay for storage. If you plan on stopping it for a long time, you should snapshot & restore instead

Aurora Backup

- Automated backups
  - 1 to 35 days (cannot be disabled)
  - point-in-time recovery in that timeframe

- Manual DB Snapshots
  - Manually triggered by the user
  - Retention of backup for as long as you want

RDS & Aurora Restore options.

- Restoring a RDS / Aurora backup or a snapshot creates a new database

- Restoring MySQL RDS database from S3
  - Create a backup of your on-premises database
  - Store it on Amazon S3 (object storage)
  - Restore the backup file onto a new RDS instance running MySQL

- Restoring MySQL Aurora cluster from S3
  - Create a backup of your on-premises database using Percona XtraBackup
  - Store the backup file on Amazon S3
  - Restore the backup file onto a new Aurora cluster running MySQL

Aurora Database Cloning

- Create a new Aurora DB Cluster from an existing one
- Faster than snapshot & restore
- Uses *copy-on-write* protocol
  - Initially, the new DB cluster uses the same data volume as the original DB cluster (fast and efficient – no copying is needed)
  - When updates are made to the new DB cluster data, then additional storage is allocated and data is copied to be separated
- Very fast & cost-effective
- Useful to create a "staging" database from a "production" database without impacting the production database

RDS & Aurora Security

- At-rest encryption:
  - Database master & replicas encryption using AWS KMS – must be defined as launch time
  - If the master is not encrypted, the read replicas cannot be encrypted
  - To encrypt an un-encrypted database, go through a DB snapshot & restore as encrypted
- In-flight encryption: TLS-ready by default, use the AWS TLS root certificates client-side
- IAM Authentication: IAM roles to connect to your database (instead of username/pw)
- Security Groups: Control Network access to your RDS / Aurora DB
- No SSH available except on RDS Custom
- Audit Logs can be enabled and sent to CloudWatch Logs for longer retention

Amazon RDS Proxy

Amazon RDS Proxy is a fully managed, highly available database proxy service provided by Amazon Web Services (AWS). It helps applications manage database connections and improve scalability, security, and manageability of Amazon Relational Database Service (RDS) and Aurora databases.

Key features and benefits of Amazon RDS Proxy include:

1. **Connection Pooling:** RDS Proxy helps manage and maintain a pool of database connections, reducing the overhead of creating and managing connections on the application side. It allows applications to efficiently use database resources.
2. **Scalability:** It automatically scales database connection capacity in response to application demand without requiring application changes. This helps handle sudden increases in connection requests.
3. **Performance:** RDS Proxy optimizes database connection management, reducing latency and improving application performance, especially for applications with fluctuating or high connection usage.
4. **Security:** It simplifies database access management by reducing the number of open connections directly to the database, thereby enhancing security. It supports IAM authentication and integrates with AWS Secrets Manager for managing database credentials.

5. **Fault Tolerance:** RDS Proxy enhances database availability by providing a highly available and fault-tolerant proxy layer between the application and the database. It helps maintain database availability during failovers or maintenance events.
6. **Monitoring and Insights:** It offers metrics and monitoring capabilities through Amazon CloudWatch, providing insights into database connections and performance.
7. **Compatibility:** RDS Proxy is compatible with various RDS database engines, including MySQL, PostgreSQL, and Amazon Aurora.

By abstracting and managing database connections, RDS Proxy allows applications to efficiently scale and manage connections, reducing the load on the database backend and improving overall application performance and reliability.