Amazon Aurora

Before we jump into – what is Aurora & why we use it – it is quite important for us to learn & understand few terms

Database Engine: The underlying software that stores, retrieves, and manages data.

For Amazon Aurora, the database engines are

- Aurora MySQL-compatible and
- Aurora PostgreSQL-compatible.

Instance: A virtual server in the AWS cloud that runs your database. Aurora instances come in different sizes to suit various workloads.

Cluster: A collection of database instances and shared storage used in Aurora to manage databases. An Aurora cluster consists of a primary instance for read/write operations and up to 15 read replicas for read-only operations.

Aurora Storage Engine: A distributed and fault-tolerant storage system that automatically scales up to 128 TB. It uses six-way replication across three Availability Zones (AZs) for durability and high availability.

Aurora Replicas: Read-only instances that share the same underlying storage as the primary instance. These replicas can be used to offload read traffic and enhance read scalability.

Cluster Endpoints:

- Cluster Endpoint: Connects to the primary instance for read/write operations.
- Reader Endpoint: Distributes read-only connections across available read replicas.
- Custom Endpoints: Can be defined to direct traffic to specific instances based on your needs.

Aurora Global Database: A feature that allows you to create a single Aurora database that spans multiple AWS regions, providing disaster recovery and low-latency global reads.

Backtrack: A feature that allows you to rewind your database cluster to a previous point in time without needing to restore data from backups.

Multi-AZ Deployment: Aurora automatically replicates data across multiple Availability Zones for high availability and durability.

Failover: The process of switching to a standby database instance in the event of a failure. Aurora provides automated failover to ensure minimal downtime.

AWS Database Migration Service (DMS): A tool that helps you migrate databases to Aurora with minimal downtime.

On-Demand Instances: Pay for database capacity by the hour with no long-term commitments.

Reserved Instances: Pay a one-time fee and commit to using Aurora for a one or three-year term to receive a significant discount.

Introduction Parameters

- Aurora is a proprietary technology from AWS (Not open sourced)
- Postgres and MySQL are both supported as Aurora Database (as we discussed before, the supported engines).
- 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 replication process is faster than MySQL
- Failover in Aurora is instantaneous and its high availability native
- Aurora costs more than RDS (20 % more) – but it is more efficient

Amazon Aurora

What makes Amazon Aurora efficient – that would be Aurora High availability & Read scaling, we will see – what that means

6 Copies of your data across 3 AZ:

- 4 Copies out of 6 needed for writes
- 3 copies out of 6 needs for reads
- Replication + Self-Healing + Auto expanding
- Storage is striped across 100s volumes

→ One Aurora Instance acts as Master, it manages the writes, remaining would be read instances

→ Automated failover for master would in less than 30 seconds

Automated failover

- A read replica is promoted as the new master in less than 30 seconds
- Aurora flips the **CNAME** record for your DB Instance to point at the healthy replica
- In case **no replica** is available, Aurora will attempt to **create a new DB Instance** in the **same AZ** as the original instance. This replacement of the original instance is done on a **best-effort basis** and may not succeed

→ Master + up to 15 Aurora read replicas serve reads

→ Support for the cross-region replication

Endpoints

- **Writer Endpoint** (Cluster Endpoint)
  - Always points to the master (can be used for read/write)
  - Each Aurora DB cluster has one cluster endpoint
- **Reader Endpoint**
  - Provides load-balancing for read replicas only (used to read only)
  - If the cluster has no read replica, it points to master (can be used to read/write)
  - Each Aurora DB cluster has one reader endpoint
- **Custom Endpoint:**
  - Load balance to a subset of replicas
  - Provides load-balanced based on criteria other than the read-only or read-write capability of the DB instances like instance class (ex, direct internal users to low-capacity instances and direct production traffic to high-capacity instances)

Aurora Working Model

Client Writes Data:

- Applications or clients send write requests (such as INSERT, UPDATE, DELETE) to the *Aurora cluster endpoint*.
- The cluster endpoint directs these requests to the *primary instance*, which handles all write operations.

Data Written to Storage:

- The primary instance processes the write requests and writes the data to the *Aurora storage engine*.
- Aurora storage is distributed and fault-tolerant, automatically replicating the data *six ways across three Availability Zones (AZs)* to ensure high durability and availability.

Storage Acknowledgement:

- Once the data is successfully written to the storage layer, the storage nodes *acknowledge the write* to the primary instance.
- The primary instance then acknowledges the *write operation back to the client*, confirming that the data is safely stored.

Data Replication:

- Aurora automatically replicates the data from the primary instance to *all read replicas* within the same cluster.
- This replication ensures that all *read replicas have the latest data* and can serve read requests accurately.

Client Reads Data:

- Applications or clients send read requests (such as SELECT queries) to the *Aurora reader endpoint.*
- The reader endpoint *load-balances these read requests across all available read replicas*, distributing the read workload and ensuring optimal performance.

Read Replica Response:

- Read replicas process the read requests by *fetching the data* from the shared storage.
- The read replicas return the *requested data to the client*, completing the read operation.

Amazon Aurora

Amazon Aurora offers several advanced features to enhance scalability, availability, and cost efficiency. Three such features are Aurora Serverless, Aurora Multi-Master & Aurora Global Database.

Aurora Serverless

- Aurora Serverless is a configuration for Amazon Aurora that automatically starts up, shuts down, and scales capacity up or down based on your application's needs. It's designed to handle variable workloads without requiring manual management of database capacity.
    - Optional
    - Automated database instantiation and auto scaling based on usage
    - Good for unpredictable workloads
    - No capacity planning needed
    - Pay per second

Aurora Multi-Master

- Aurora Multi-Master allows you to create multiple write nodes in an Aurora database cluster. This feature enhances the availability and scalability of your database by allowing multiple nodes to accept write operations.
    - Optional
    - Every node (replica) in the cluster can read and write
    - Immediate failover for writes (high availability in terms of write). If disabled and the master node fails, need to promote a Read Replica as the new master (will take some time).
    - Client needs to have multiple DB connections for failover

Aurora Global Database

- Aurora Global Database allows you to create a single Aurora database that spans multiple AWS regions, providing a globally distributed database solution with low-latency reads in different regions and robust disaster recovery capabilities.
    - Entire database is replicated across regions to recover from region failure
    - Designed for globally distributed applications with low latency local reads in each region
    - 1 Primary Region (read / write)
    - Up to 5 secondary (read-only) regions (replication lag < 1 second)
    - Up to 16 Read Replicas per secondary region
    - Helps for decreasing latency for clients in other geographical locations
    - RTO of less than 1 minute (to promote another region as primary)

One more noticeable feature would be

Aurora Events

- Invoke a **Lambda** function from an **Aurora MySQL-compatible DB cluster** with a **native function** or a **stored procedure** (same as RDS events)
- Used to capture data changes whenever a row is modified

Security

At-rest Encryption

- The master and replicas can be encrypted as rest using KMS, but encryption must be enabled at launch time.
- If master is not encrypted, read replicas cannot be encrypted.
- To encrypt an unencrypted database, create a snapshot and restore the database as encrypted using a key from KMS.

In-flight Encryption

Client needs to use TLS root certificates provided by AWS to establish TLS (in-flight encryption) with RDS or Aurora.

DB Authentication

- Username:Password based authentication (available on both RDS and Aurora)
- **IAM DB Auth** - use IAM roles to authenticate to DB instead of configuring users on DB (recommended as it allows us to manage all the authentication in IAM)

Network Security

- Security Groups control network access to the underlying EC2 instances
- No SSH access available as the underlying instances are AWS managed (except on RDS custom)

Audit Logs

- Audit logs can be enabled - store logs locally for a short period of time
- Send audit logs to CloudWatch for log retention

Keep in mind

- Aurora Replicas are created in the same DB cluster within a Region. With Aurora MySQL you can also enable binlog replication to another Aurora DB cluster which can be in the same or different region.

Important parameters:

- max_connections - max number of simultaneous connections Aurora allows
- max_user_connections - max number of simultaneous connections Aurora allows for a single user

Amazon Aurora

Numerical Questions:

https://lisireddy.medium.com/amazon-aurora-numerical-questions-7faceafbdb12

Scenario based Questions:

https://lisireddy.medium.com/amazon-aurora-scenario-based-questions-002d63866e31

Wish you the best …! Happy Learning ..!

Yours' Love (@lisireddy across all the platforms)