Amazon RDS

The basic terms we need to understand are

Amazon RDS (Relational Database Service): A managed database service provided by AWS that simplifies the setup, operation, and scaling of relational databases.

Instance Type: The type of virtual server used to run your RDS database. Different instance types offer varying levels of CPU, memory, and storage performance.

Database Engine: The software that manages and interacts with the database. RDS supports several engines, including MySQL, PostgreSQL, MariaDB, Oracle, and SQL Server.

DB Instance: An instance of a database engine within RDS. It's the compute resource where your database is hosted.

DB Cluster: A collection of DB instances that work together for high availability and read scalability. This concept is more relevant for Amazon Aurora, a specific RDS database engine.

Read Replica: A copy of the database instance that handles read requests. It helps in scaling read operations and improving performance.

Backup: A copy of your database data. RDS provides automated backups and manual snapshots to protect your data.

Snapshot: A manual backup of the database instance. Snapshots are user-initiated and stored until explicitly deleted.

Parameter Group: A collection of settings that define how your RDS database engine behaves. You can modify the parameters to tune database performance.

Option Group: A collection of features or options that can be enabled or configured for your database instance, such as SQL Server Integration Services.

Maintenance Window: A specified time period during which RDS applies updates and patches to your database instance.

Multi-AZ Deployment: A deployment option that enhances the availability of your RDS instance by creating a synchronous standby replica in a different Availability Zone.

Automatic Failover: The process of automatically switching to a standby replica if the primary database instance fails, ensuring high availability.

Amazon RDS

Introductory Parameters

- *Supported Engines*
    - *Postgres*
    - *MySQL*
    - *MariaDB*
    - *Oracle*
    - *Microsoft SQL Server*
    - *Aurora (AWS Proprietary database)*
- ***Backed by EC2 instances with EBS storage***
- *We don't have access to the underlying instance (cannot SSH into it)*
- ***DB connection is made on port 3306***
- *SG is used for network security (must allow incoming TCP traffic on port 3306 from specific IPs)*

Backups

- **Automated Backups** (enabled by default)
    - Daily full backup of the database (during the defined maintenance window)
    - Backup retention: 7 days (max 35 days)
    - **Transaction logs** are backed-up every **5 minutes** for **Point In Time Recovery (PITR)**
    - **Automated backups happen in the same region** (can happen in multiple AZs in a multi-AZ deployment)
- **DB Snapshots:**
    - Manually triggered
    - Backup retention: unlimited
    - **Snapshots can be saved across regions**

Amazon RDS

Amazon RDS Working model

Client Writes Data

Step 1: Client Request

A client (e.g., application or user) initiates a write operation by sending a query to the database instance. This could be an INSERT, UPDATE, or DELETE operation.

Step 2: Query Execution

The RDS instance receives the write query and processes it. The database engine executes the query, modifying the data in the database.

Step 3: Data Storage

The database engine writes the updated data to the storage layer. Amazon RDS uses a storage subsystem that ensures data durability and consistency. For instance:

- **Transactional Logs**: The database writes changes to transaction logs (e.g., write-ahead logs). These logs help in maintaining data integrity and enable recovery in case of failures.
- **Data Files**: The actual data changes are written to data files on disk.

Step 4: Acknowledgment

Once the write operation is complete, the database instance acknowledges the completion to the client. This ensures that the data has been successfully written to the database.

Client Reads Data

Step 1: Client Request

The client sends a read query (e.g., SELECT) to retrieve data from the database.

Step 2: Query Processing

The RDS instance processes the read query. The database engine retrieves the requested data from the storage layer.

Step 3: Data Retrieval

The data is fetched from the data files on disk. If the data has been recently written, it may be read from the buffer cache (in-memory storage) to improve performance.

Step 4: Result Delivery

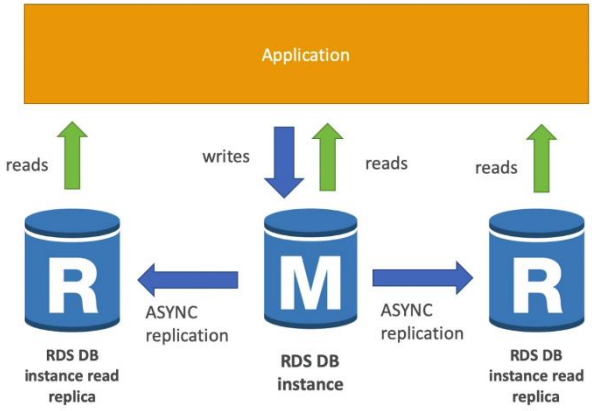
The database engine sends the retrieved data back to the client.

Amazon RDS

Now we need to understand the multiple features or aspects of the RDS

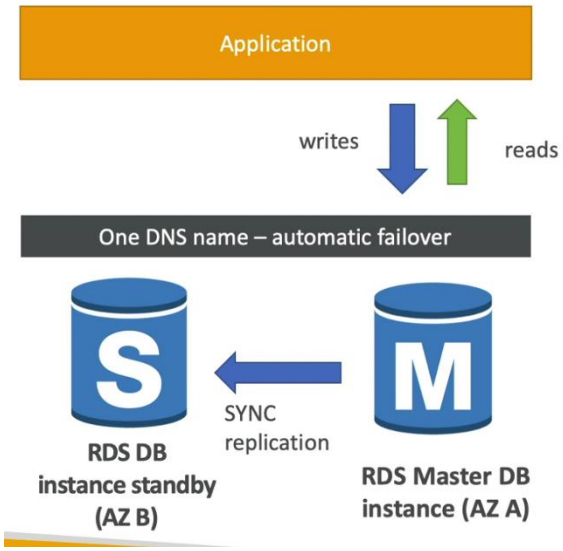
Read Replicas

- Allows us to scale the read operation (SELECT) on RDS
- **Up to 5 read replicas** (within AZ, cross AZ or cross region)
- **Asynchronous Replication** (seconds)
- **Replicas can be promoted to their own DB**
- **Applications must update the connection string to leverage read replicas**
- Network fee for replication
    - Same region: free
    - Cross region: paid



Multi AZ

- Increase availability of the RDS database by replicating it to another AZ
- **Synchronous Replication**
- **Connection string does not require to be updated** (both the databases can be accessed by one DNS name, which allows for automatic DNS failover to standby database)
- When failing over, **RDS flips the CNAME** record for the DB instance to point at the standby, which is in turn promoted to become the new primary.
- Cannot be used for scaling as the standby database cannot take read/write operation while the master database is still active.

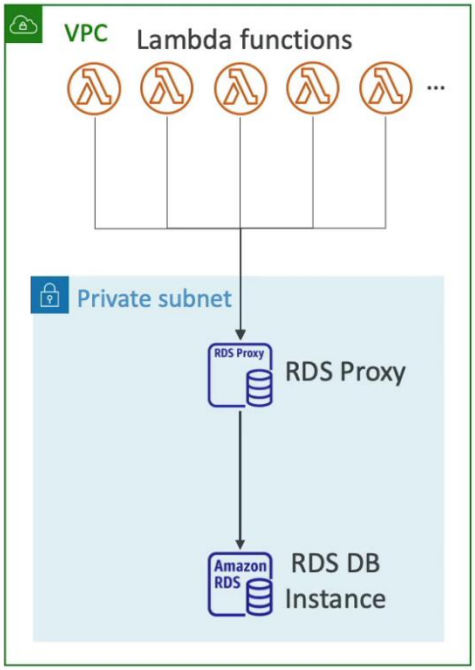


You can create a read replica as a Multi-AZ DB instance. A standby of the replica will be created in another AZ for failover support for the replica.

Amazon RDS

RDS Proxy

- ***Serverless**, auto-scaling, **multi-AZ** proxy for RDS*
- *The applications connect to the RDS proxy instead of connecting directly to the RDS. The **RDS proxy pools and shares DB connections among the applications**.*
- *Reduces CPU and RAM requirements of the DB when a large number of applications need to connect to the DB*
- ***Minimizes open connections and timeouts***
- ***DB failover is handled by RDS proxy** (reduces failover time by up to 66%)*
- *Supports **RDS (MySQL, PostgreSQL, MariaDB)** and **Aurora (MySQL, PostgreSQL)***
- *No code changes required (just update the connection URL)*
- *Allows to enforce IAM DB Auth (credentials can be stored in Secrets Manager)*
- *RDS Proxy can only be accessed from within the VPC*



Encryption

At rest encryption

- KMS **AES-256** encryption
- Encrypted DB ⇒ Encrypted Snapshots, Encrypted Replicas and vice versa

In flight encryption

- **SSL certificates**
- Force all connections to your DB instance to use SSL by setting the rds.force_ssl parameter to true
- To enable encryption in transit, download the **AWS-provided root certificates** and use them when connecting to DB

To encrypt an un-encrypted RDS database:

- Create a snapshot of the un-encrypted database
- Copy the snapshot and enable encryption for the snapshot
- Restore the database from the encrypted snapshot
- Migrate applications to the new database, and delete the old database

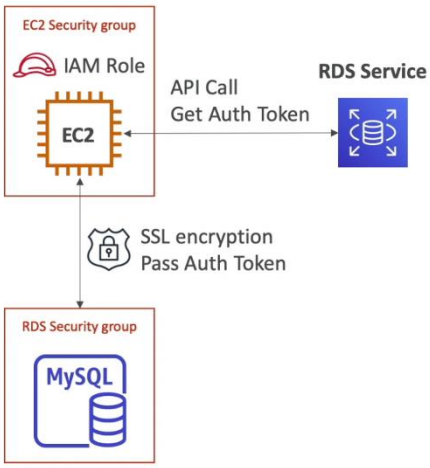
To create an encrypted cross-region read replica from a non-encrypted master:

- Encrypt a snapshot from the unencrypted master DB instance
- Create a new encrypted master DB instance
- Create an encrypted cross-region Read Replica from the new encrypted master

Amazon RDS

Access Management

- *Username and Password can be used to login into the database*
- *EC2 instances & Lambda functions should access the DB using **IAM DB Authentication** (**AWSAuthenticationPlugin** with **IAM**) - **token based access***
  - *EC2 instance or Lambda function has an IAM role which allows is to make an API call to the RDS service to get the **auth token** which it uses to access the MySQL database.*
  - *Only works with **MySQL** and **PostgreSQL***
  - *Auth token is valid for **15 mins***
  - *Network traffic is encrypted in-flight using SSL*
  - *Central access management using IAM (instead of doing it for each DB individually)*
- *EC2 & Lambda can also get DB credentials from Parameter Store to authenticate to the DB - **credentials based access.***



RDS Events

- RDS events only provide operational events on the DB instance (not the data)
- To capture data modification events, use **native functions** or **stored procedures** to invoke a **Lambda** function.

Monitoring

- **CloudWatch Metrics for RDS**
  - Gathers metrics from the **hypervisor** of the DB instance
    - CPU Utilization
    - Database Connections
    - Freeable Memory
- **Enhanced Monitoring**
  - Gathers metrics from an agent running on the RDS instance
    - OS processes
    - RDS child processes
  - Used to monitor different **processes or threads on a DB instance** (ex. percentage of the CPU bandwidth and total memory consumed by each database process in your RDS instance

Logging

- **Error Logs** - enabled by default
- **Slow Query Logs** - logs queries that took longer to execute (can be enabled)
- **General Logs** - can be enabled
- **Audit Logs** - can be enabled

Maintenance & Upgrade

Any database engine level upgrade for an RDS DB instance, with Multi-AZ deployment, triggers both the primary and standby DB instances to be upgraded at the same time. This causes **downtime** until the upgrade is complete. This is why it should be done during the maintenance window.

Other noticeable parameters

RDS supports using **Transparent Data Encryption (TDE)** to encrypt stored data on your DB instances running Microsoft SQL Server. TDE automatically encrypts data before it is written to storage, and automatically decrypts data when the data is read from storage.

Amazon RDS

Amazon RDS

Numerical Questions

https://lisireddy.medium.com/amazon-rds-numerical-questions-a70855d1190a

Scenario based Questions

https://lisireddy.medium.com/amazon-rds-scenario-based-questions-1e70df95989d

Wish you the best …! Happy Learning ..!

Yours' Love (@lisireddy across all the platforms)