AWS Relational Database Service (RDS)

What id RDS ?

- A managed relational database service provided by AWS
- Used for OLTP (Online Transactional Processing) systems
- Launched in a region
- Supports with single-az and multi-az deployment
- Read replica supported across regions (Aurora, MySQL)
- Backups are in the form of snapshots and stored in S3 and it takes incremental backups
- Lives inside VPC and supported with SG
- Access is DB user based
- Managed by AWS
    - Software update of the underlying OS and DB engine
    - Security and Patching
    - Automated backup
    - In case of multi-az deployment for high availability replication between master and slave will be managed by AWS
    - Automatic failover in case of multi-az deployment
    - DB maintenance , default is 7 days and max 35 days
- Managed by Customer
    - DB settings
    - Creation of DB schema
    - Performance tuning
- Types of RDS DB engines
    - MySQL
    - MSSQL : support 64TB
    - Oracle
    - Aurora : High throughput
    - PostgreSQL : highly reliable and stable
    - MariaDB : MySQL Compatible : 64TB
- Licensing Options
    - BYOL
    - License included
- RDS Limits
    - Up to 40 DB instances per account
    - 10 of the 40 can be Oracle or MSSQL servers under license included model
    - Under BYOL model all 40 can be any DB engine
- RDS Instance Storage
    - EBS is the only storage used for RDS DB
    - General purpose : used for moderate I/O requirement
        - Min-20GB
        - Max-16TB
    - Provisioned IOPS : used for high performance or OLTP workloads

        - Min-100GB
        - Max-16TB
- Deployment/Provisioning type
    - Free tier
    - Dev/test
    - Production

- DB Instance Size/Class
    - Standard
        - M-class
        - Max-96vCPU
        - 384GB RAM
        - EBS : 1400Mbps
    - Memory optimized
        - R, X Class
        - Max-96vCPU
        - 768GB RAM
        - EBS : 1400 Mbps
    - Burstable class
        - T class
        - Max-8vCPU
        - 32GB RAM
        - EBS : 1500Mbps
- **Subnet Groups** :
    - Collection of subnets from your VPC one from each availability zone which allow RDS to create DB instances on the specified subnets or in case of multi-az deployment the replicas get created on the specified subnets
- **Parameter Groups** :
    - Database engine configuration which holds the values that are applied to one or more DB instances. If you do not specified any specific parameter group it uses the default parameter group of RDS instance
- **Option Groups** :
    - Offer additional features that make it easier to manage data and databases, and to provide additional security for your database. Amazon RDS uses option groups to enable and configure these features

Multi-AZ RDS Concept :

- Multi AZ can be selected during creation only, not after
- Standby or slave can be created in any of the AZs than the primary one in the same region, you cannot decide on which AZ standby should be created, however you can view on which AZ it is created after creation
- It configure the automatic Synchronous Replication between master and slave
- You cannot access Read/Write on the standby

- Inc case of failover standby become as primary and another secondary will get created automatically. In case the existing (old) primary get ready in few minutes then that may become as standby, but can't be the primary again unless the current new primary become failure.
- Depending on the Instance class it may take 1 minutes to few minutes to failover on the standby, during the failure it holds the requests/processes and start sending them to the standby when it becomes as primary
- Recommendation is use provisioned IOPS when you use Multi-AZ
- RDS failover triggers :
    - When primary DB is fail/down inaccessible due to any reason
    - In case of RDS instance failure or EBS Storage failure
    - In case of failure in primary AZ, can be network, hardware.
    - Because of connectivity between you application and primary DB
    - During patching, upgrade or maintenance, probably when your RDS needs reboot
    - When failover happen CNAME of the primary RDS get updated and mapped to the IP of the standby RDS automatically
    - As a best practice always use RDS endpoint/DNS instead of IP
    - RDS endpoint remain same in case of failover
    - You can do manual failover by rebooting the primary by selecting reboot with failover option
    - Use case can you may want to change instance type or making any changes on the DB setting by modifying DB Parameter
    - When failover occur RDS sends SNS notification

Backup and Retention :

- Using API and CLI you can get last 14 days events history
- From console you can only get last 1 days events history
- Patching, upgrade happens initially on the standby and then on the primary
- Snapshots and automated backups are done on the standby to avoid any interruption or performance
- If you have read-only RDS instance then its recommended to perform backup and snapshots on the read-only instance
- DB version upgradation or any other maintenance should be performed during maintenance window as it get applied on the both at the same time
- Backups can a manual one or automated one, automated backup will happened during defined backup window and it takes on daily basis and starting from 1days and max 35 days and default id 7 days, if you define 0 then it won't take any backup
- In case of Aurora by default 1 day from console
- In case of API or CLI 1 day is default in all the cases
- Backups can be taken only for the DB or for the whole instance as snapshots
- Backups stored on S3, no charge for automated backup but charges applied on backup storage
- Automated backups are deleted when you delete RDS instance, but if you want you can keep them
- Automated backups are only applied if the DB instance is ACTIVE

- Backups are performed on standby if DB engines are on MySQL, MariaDB, PostgreSQL and Oracle
- During DB backup activities DB freeze any new operations
- If you change backup retention then there will be an outage

Amazon Aurora :

- Aurora is a proprietary database engine from AWS, enterprise grade database with cost effective price than commercial databases like MSSQL or Oracle
- Aurora is supported with PostgreSQL and MySQL,
- It is optimized database and claims for 5x performance improvement over MySQL on RDS, over 3x the performance of PostgreSQL on RDS
- Storage automatically grows in increments of 10GB up to 64 TB
- Aurora can have 15 replicas while RDS has 5 and replication is faster than RDS (10 ms replica lag)
- Failover in Aurora is instantaneous, its HA cloud native
- Costs more than RDS with around 20% more, but more efficient
- Stores 6 copies of your data across 3AZs
    - 4 copies needs for writes
    - 3 copies for reads
    - Self-healing with per-to-peer replication
    - Storage is stripped across 100s of volumes
- One aurora instance takes writes (master)
- Automated failover for master in less than 30seconds
- Master with up to 15 read replicas servers reads
- Support for cross region replication
- Aurora DB Cluster
    - Writer endpoint, pointing to the master
    - Reader endpoint, pointing to all readers with load balancing connections
    - Read replicas are auto scaled, while master is not part of auto scale
- Features/Security :
    - Automatic failover
    - 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
    - Encryption at rest using KMS
    - Automated backups, snapshots and replica are also encrypted
    - Encryption in transit using SSL same as MySQL or Postgres RDS
    - Authentication using IAM
    - Supported with Security Group

    - You can’t SSH to Aurora as RDS
- Aurora Serverless
    - No need to choose instance size
    - Supported with MySQL and Postgres
    - Useful for unpredictable workloads
    - DB cluster starts, terminate and scales automatically based on the CPU and connections
    - Can be migrated from Aurora cluster to Aurora serverless and vice versa
    - Usage is measured in ACU (Aurora Capacity Units)
    - Billed in 5 minutes increments of ACU
    - Important : Some features of aurora are not supported in serverless mode, so as a best practice to follow the document to compare required featured based on the requirement

1. MySQL DB Administration

1.0 To Connect RDS :

- **To Connect to the RDS (mysql database) :**

  # mysql -h <endpoint of RDS> -u <user name> -p

  # Password : (enter the password)

2.1 To Start/ Stop/Restart Service :

- **To start/stop/restart mysql service :**

  # servicemysqld start / stop /restart

2.2 User Management :

- **To create a user :**

  >create user ‘user_name’@’localhost’ identified by ‘password’;

- **To grant permissions to a user :**

  >grant all privileges on test to 'user1'@'localhost';
  >grant all privileges on *.* to 'user1'@'localhost';

- **To list the users :**

  >SELECT User FROM mysql.user;

- **To see the users count :**

  >SELECT count(*) as total_record FROM users;

- **To change the password fofr a user :**

  >SET PASSWORD FOR 'user1'@'localhost' = password ('sudhir');

- **To delete a user :**

  >drop user 'user3'@'localhost';

2.3 Database and Table Management :

- **To list all the databases :**

  >show databases;

- **To change one database to another :**

  >use databases;

- **To list All the tables :**

  >show tables;

- **To list items of a table :**

  >select * from table_name;

- **To create a database :**

  >create database db_name;

- **To create a table :**

  >create table pets (sl_noint, name varchar(20),color varchar(20),location varchar(20));

- **To insert data into a table :**

  >insert into pets (sl_no,name,color,location) values (1,"riku","red","bbsr");

- **To delete a table :**

  >DROP TABLE table_name;

- **To delete a database :**

  >drop database database_name;

2.4 Database Backup and Restore :

- **To take backup of database : (AWS)**

  # mysqldump -h <rds endpoint> -P:<port no> -u <usr-name> -p <passwd>db_name>>/home/bkp.mysql

- **To restore database :**

  # mysql -h <rds endpoint> -P:<port no> -u <usr-name> -p <passwd>db_name<< /home/bkp.mysql