Database

It is the system that store and manage data.

Two types of database

- Relational (SQL)
- Non- Relational(NoSQL)

Relational Database:

Relationship between tables

Schema – Structure in and between tables of data

Non – relational Database

NoSQL – Not one single thing

Anything that does not fit into SQL

Databases on EC2 Instance:

Databases created in EC2 instance.

To deploy any application, we have 3 layers i.e, webserver, application and database.

For database layer we create a database on ec2

Why should we run database on EC2

- If our database needs any information related to operating system or database need to communicate with operating system. Then wee go for database on EC2
- If vendor demands for database on EC2
- If we want the database or database version that is not supported by AWS, then we go for database on EC2
- We might need a specific version of an OS and DB that AWS doesn't provide.

Why shouldn't we run DB on EC2

- Admin overhead: If we want to upgrade the EC2, then we need to take care of the backup.
- Backup and DR: Everything is maintained by us.
- EC2 should be run in a single AZ only
- Will miss out some features from AWS DB products.
- Skills and setup time to monitor (Access to user and security issues)
- Performance will be slower than AWS option

Practical:

Installing Db(maria DB) on EC2:

Create an EC2 instance and connect to the instance

```
Sudo su –
Yum -y install mariadb-server
Systemctl enable mariadb
Systemctl start mariadb
Yum -y update
```

Mariadb works on Port no: 3306

We can set environmental variables instead of hard coding (values which are used in many places)

```
DBName=ec2db
DBPassord=admin123456
DBRootPassword=admin123456
DBUser=ec2dbuser
```

Database setup on ec2

```
echo "CREATE DATABASE ${DBName};" >> /tmp/db.setup
echo "CREATE USER '${DBUser}' IDENTIFIED BY '${DBPassword}';" >> /tmp/db.setup
echo "GRANT ALL PRIVILEGES ON *.* TO '${DBUser}'@'%';" >> /tmp/db.setup
echo "FLUSH PRIVILEGES;" >> /tmp/db.setup
mysqladmin -u root password "${DBRootPassword}"
mysql -u root --password="${DBRootPassword}" < /tmp/db.setup
rm /tmp/db.setup
```

Add some dummy data to the database

```
mysql -u root --password="${DBRootPassword}"
USE ec2db;
CREATE TABLE table1 (id INT, name VARCHAR(45));
INSERT INTO table1 VALUES(1, 'Virat'), (2, 'Sachin'), (3, 'Dhoni'), (4, 'ABD');
SELECT * FROM table1;
```

RDS: Relational Database Service

- It is Database as a service
- Completely managed by AWS – We create the database and we integrate with application
- Databases supported by AWS are MySQL, MariaDB, PostgreSQL, Oracle, Microsoft SQL
- We can host one or more databases on RDS instance
- There is one database supported by RDS which is developed by AWS i.e, Amazon Aurora

RDS Database Instance

- Database connects with a CNAME (canonical name). RDS uses standard database engine.
- RDS uses standard database engines. Database can be optimized for db.m5 general, db.r5 memory, db.t3 burst.
- When we provision an instance, we provision storage that is dedicated to that instance. That is EBS storage located in the same AZ. RDS is vulnerable to failures in that AZ.
- Storage can be allocated with SSD or magnetic
- RDS will come with an instance and storage. Billing will done accordingly. (how much storage is consumed)

Practical : Creating the RDS

Go to RDS (We can create up to 40 DB instances)
Click on create db
Choose a database creation method
Select Standard create
Engine Options (type of db): Mariadb
Engine Version: Select the version
Templates: Free tier
Settings:
DB instance identifier: Maria-DB
Credential Setting
DB instance class
Db.t2.micro

Storage : (Allows 20 GB)
Database options: (if we want to create a database with the creation of RDS we use this option)
Backup: (Enable automated backups)
Backup retention period:7
Logs : (We have option to push the logs to cloud watch)
Enable auto minor version upgrade: If we use this option, if AWS see any minor upgrade in the market it will automatically upgrade the db

Click create database

Now endpoint is created i.e CNAME (canonical name)

Migrating DB from ec2 to RDS

1) Get the dump of your existing DB on EC2
mysqldump -u root -p database_name > file_name.sql

2) Migrate the DB dump that you have taken in step 1 to RDS
mysql -h <replace-rds-end-point-here> -P 3306 -u <user_name> -p database_name < ec2db.sql

3) Connect to your RDS DB instance
mysql -h <replace-rds-end-point-here> -P 3306 -u rdsuser -p

4) Switch to the database and verify the details.
USE rdsdb
SELECT * FROM table1;

RDS Multi AZ (High Availability)

- If we use this option in RDS, Secondary hardware will be allocated in another availability zone. That is copy of the same database is created.
  That copy is called standby replica
- RDS enable the synchronous replication
- The standby replica cannot be used for extra capacity
- If for any reason there is problem in primary database and AWS will detect that and it automatically failover to the standby replica and will take 60 to 120 sec to change the database.
- This option is not providing fault tolerance (without down time)

Points To remember

- Multi AZ feature is not free and we need to pay twice the price.
- Standy replica cannot be accessed directly unless a failure occurs.
- Failover is highly available, not fault tolerant.
- Backups taken from standby (Removes performance impacts)
- Multi AZ will be created in same region.

RDS Backup and Restores:

Key Terminologies in Backup and Restores.

- RPO
- RTO

RPO - Recovery Point Objective

- Time between the last backup and whenever the failure occurred.
- Ex: We have taken backup at 6 a.m and failure happened at 7 a.m
- then RPO is for 1 hour and we will lose 1 hr of data.
- Amount of maximum data loss.
- Business provides an RPO value.
- influence technical solution and case.

RTO- Recovery Time Objective

- Time between the DR event and full recovery.
- Ex: Disaster happened at 6 a.m and you took 30 mins to restore back the DB.
- Then 30 mins we can say as RTO.
- Influenced by process, staff, tech and documentation.

Backups:

- First snap is full size of consumed data.
- After first snapshot it will only capture the latest data written to db.
- Manual snapshots will remain in your aws account even after the life of snapshot, these needs to be deleted manually.
- Automatic Snapshots.

- Automatic cleanups can be anywhere from 0 to 35 days.
- When you delete the db, they can be retained but they will
- expire based on their retention period.

Points to Remember:

- When performing a restore, RDs creates new endpoint address.
- Restores aren't fast, think about RTO.

RDS Read-Replicas:

Read Replicas is a copy of the primary instance.

Use case: Let’s assume we have a primary RDS instance that serves both read and write traffic.

Due to the size of the instance and the amount of read-intensive traffic being directed to the

database for queries. the performance of the instance is taking a hit.

To help resolve this, you can create a read replica. A snapshot will be taken of your database,

and if you are using Multi-AZ, then this snapshot will be taken of your secondary database instance

to ensure that there are no performance impacts during the process. Once the snapshot is completed,

a read replica instance is created from this data.

1) Asynchronous replication.

2) It is written fully to the primary instance.

once its stored-on disk it is then push to the replica db.

This means there could be small log.

3) These can be created in the same region or different region, it is a cross region replication.

why we need Read Replicas:

- We can create 5 read-replicas per DB instance.
- Each of these provides an additional instance of Read performance.
- This allows you to scale out read operations for an instance.
- Can provide global performance improvements.