

Database:

In AWS we have wide range of database services to fit our application requirements. These database services are fully managed and can be launched in minutes with just a few clicks.

AWS database services include:

- Amazon Relational Database Service (Amazon RDS) support for six commonly used database engines
 - Amazon Aurora,
 - MySQL,
 - PostgreSQL
 - Oracle
 - MS SQL
 - Maria DB
- Amazon DynamoDB, a fast and flexible NoSQL database service,
- Amazon Redshift, a petabyte-scale data warehouse service, and
- Amazon ElastiCache, an in-memory cache service with support for Memcached and Redis.
- AWS also provides the AWS Database Migration Service, a service which makes it easy and inexpensive to migrate your databases to AWS cloud.

Amazon Relational Database Service (Amazon RDS)

The most common type of database in use today is the relational database. The relational database has roots going back to the 1970s when Edgar F. Codd, working for IBM, developed the concepts of the relational model. Today, relational databases power all types of applications from social media apps, e-commerce websites, and blogs to complex enterprise applications.

Amazon Relational Database Service (Amazon RDS) makes it easy to set up, Operate, and scale a relational database in the cloud. It provides cost-efficient and resizable capacity while managing time-consuming database administration tasks such as hardware provisioning, database setup, patching and backups.

- A relational database consists of one or more tables, and a table consists of columns and rows similar to a spreadsheet.
- A database column contains a specific attribute of the record, such as a person's name, address, and telephone number.
- Each attribute is assigned a data type such as text, number, or date, and the database engine will reject invalid inputs.

StudentID	FirstName	LastName	Gender	Age
-----------	-----------	----------	--------	-----

101	Avinash	Reddy	M	28
102	Anudeep	Thi	M	26
103	Aravind	Reddy	M	24
104	Vikas	Ch	M	24

Here is an example of a basic table that would sit in a relational database. There are five fields with different data types:

StudentID = Number or integer

FirstName = String

LastName = String

Gender = String (Character Length = 1)

Age = Integer

This sample table has four records, with each record representing an individual student. Each student has a *StudentID* field, which is usually a unique number per student. A unique number that identifies each student can be called a **primary key**.

A relational database can be categorized as either an Online Transaction Processing (OLTP) or Online Analytical Processing (OLAP) database system, depending on how the tables are organized and how the application uses the relational database.

OLTP refers to transaction oriented applications that are frequently writing and changing data (for example, data entry and e-commerce).

OLAP is typically the domain of data warehouses and refers to reporting or analyzing large data sets.

Data Warehouses: A data warehouse is a central repository for data that can come from one or more sources. This data repository is often a specialized type of relational database that can be used for reporting and analysis via OLAP. Organizations typically use data warehouses to compile reports and search the database using highly complex queries.

NoSQL Databases

NoSQL databases have gained significant popularity in recent years because they are often simpler to use, more flexible, and can achieve performance levels that are difficult or impossible with traditional relational databases.

Traditional relational databases are difficult to scale beyond a single server without significant engineering and cost, but a NoSQL architecture allows for horizontal scalability on commodity hardware.

- NoSQL databases are non-relational and do not have the same table and column semantics of a relational database.
- NoSQL databases are instead often key/value stores or document stores with flexible schemas.

Advantages if you with RDS over On-Premise or EC2

Responsibility	Database On-Premise	Database on Amazon EC2	Database on Amazon RDS
App Optimization	You	You	You
Scaling	You	You	AWS
High Availability	You	You	AWS
Backups	You	You	AWS
DB Engine Patches	You	You	AWS
Software Installation	You	You	AWS
OS Patches	You	You	AWS
OS Installation	You	AWS	AWS
Server Maintenance	You	AWS	AWS
Rack and Stack	You	AWS	AWS
Power and Cooling	You	AWS	AWS

Database Engines

Amazon RDS supports six database engines: MySQL, PostgreSQL, MariaDB, Oracle, SQL Server, and Amazon Aurora.

MySQL: MySQL is one of the most popular open source databases in the world, and it is used to power a wide range of applications, from small personal blogs to some of the largest websites in the world. Amazon RDS MySQL allows you to

connect using standard MySQL tools such as MySQL Workbench or SQL Workbench/J.

PostgreSQL: PostgreSQL is a widely used open source database engine with a very rich set of features and advanced functionality. Amazon RDS PostgreSQL can be managed using standard tools like pgAdmin and supports standard JDBC/ODBC drivers.

MariaDB: MariaDB is a popular open source database engine built by the creators of MySQL and enhanced with enterprise tools and functionality.

Oracle: Oracle is one of the most popular relational databases used in the enterprise and is fully supported by Amazon RDS. Amazon RDS supports access to schemas on a DB Instance using any standard SQL client application, such as Oracle SQL Plus.

Microsoft SQL Server

Microsoft SQL Server is another very popular relational database used in the enterprise. Amazon RDS allows Database Administrators (DBAs) to connect to their SQL Server DB Instance in the cloud using native tools like SQL Server Management Studio.

Amazon RDS SQL Server also supports four different editions of SQL Server: Express Edition, Web Edition, Standard Edition, and Enterprise Edition.

Licensing: AWS offers two licensing models: **License Included** and **Bring Your Own License (BYOL)** for Amazon RDS Oracle and Microsoft SQL Server as they are commercial software products.

Amazon Aurora: Amazon Aurora is a fully managed service, is MySQL compatible, and provides for increased reliability and performance over standard MySQL deployments. Amazon Aurora can deliver up to five times better performance compared to MySQL. We can use the same code, tools, and applications that we use with existing MySQL databases with Amazon Aurora.

- 2 copies of your data is contained in each availability zone, with minimum of 3 availability zones. 6 copies of your data.
- Aurora is designed to transparently handle the loss of up to two copies of data without affecting database write availability and up to three copies without affecting read availability.
- Aurora storage is also self-healing. Data blocks and disks are continuously scanned for errors and repaired automatically.
- We can create two types of replications for Aurora
 - Aurora Replicas (currently 15)
 - MySQL Read Replicas (currently 5)

Storage Options

Amazon RDS uses Amazon Elastic Block Store (Amazon EBS). Based on your performance and cost requirements we can select Magnetic, General Purpose (Solid State Drive [SSD]), or Provisioned IOPS (SSD). Depending on the database engine and workload, you can scale up to 4 to 6TB in provisioned storage and up to 30,000 IOPS.

Backup and Recovery

Amazon RDS provides two mechanisms for backing up the database:

1. Automated backups and
2. Manual snapshots.

Automated Backups: An automated backup is an Amazon RDS feature that continuously tracks changes and backs up your database.

- You can set the backup retention period when you create a DB Instance. Default of 7 days, but you can modify the retention period up to a maximum of 35 days.
- When you delete a DB Instance, all automated backup snapshots are deleted and cannot be recovered.
- Automated backups will occur daily during a configurable 30-minute maintenance window called the backup window.
- Automated backups are kept for a configurable number of days, called the backup retention period.
- You can restore your DB Instance to any specific time during this retention period, creating a new DB Instance.

Manual DB Snapshots: This is a manually initiated task. We have to perform this backup manually.

- A DB manual snapshot is initiated by us and can be created as frequently as we want.
- We can then restore the DB Instance to the specific state in the DB snapshot at any time.
- Manual DB snapshots are kept until you explicitly delete them with the Amazon RDS console.

Recovery: We can use automated backup or manual snapshot to recovery the database.

- Amazon RDS allows you to recover your database using automated backups or manual DB snapshots.
- You cannot restore from a DB snapshot to an existing DB Instance; a new DB Instance is created when you restore.

- When using automated backups, Amazon RDS combines the daily backups performed during your predefined maintenance window in conjunction with transaction logs to enable you to restore your DB Instance to any point during your retention period, typically up to the last five minutes.

Multi-AZ: By using Multi-AZ we can increase the availability of the database using replication. We will get a same copy of production database in another availability zone for DR purpose (Disaster Recovery).

- Multi-AZ allows you to place a secondary copy of your database in another Availability Zone for disaster recovery purposes.
- Multi-AZ deployments are available for all types of Amazon RDS database engines.
- When you create a Multi-AZ DB Instance, a primary instance is created in one Availability Zone and a secondary instance is created in another Availability Zone.
- Amazon will takes care about the replication between primary Database and Secondary database.
- Amazon RDS detects and automatically recovers from most common failures for Multi-AZ deployments so that we will not get any downtimes and recovers without administrative intervention.
- Multi-AZ deployments are for disaster recovery only; they are not meant to enhance database performance.
- To improve database performance/Scaling we have to use read replicas or ElastiCache.

Read replicas:

Read replica's allow you to have a read only copy of your production database. This is achieved by using Asynchronous replication from the primary RDS instance to the read replica. You use read replica's primarily for very read-heavy database workloads.

Read replicas are currently supported for:

- MySQL,
- PostgreSQL,
- MariaDB, and
- Amazon Aurora.

- Updates made to the source DB Instance are asynchronously copied to the read replica.
- You can create one or more replicas of a database within a single AWS Region or across multiple AWS Regions.
- We can use Read replicas for Scaling!!! Not for DR!
- Must have automatic backups turned on in order to deploy a read replica.
- You can have up to 5 read replicas copies of any databases
- You can have read replicas of read replicas and each read replica will have its own DNS end point.
- You cannot have Read Replicas that have Multi-AZ
- You can create Read Replica's of Multi-AZ source databases however.
- Read Replicas can be promoted to be their own databases. This breaks the replication.

Amazon DynamoDB:

Amazon DynamoDB is a fully managed NoSQL database service that provides fast and low latency performance that scales with ease. Amazon DynamoDB significantly simplifies the hardware provisioning, setup and configuration, replication, software patching, and cluster scaling of NoSQL databases.

Amazon DynamoDB can provide consistent performance levels by automatically distributing the data and traffic for a table over multiple partitions. After you configure a certain read or write capacity, Amazon DynamoDB will automatically add enough infrastructure capacity to support the requested throughput levels. As your demand changes over time, you can adjust the read or write capacity after a table has been created, and Amazon DynamoDB will add or remove infrastructure and adjust the internal partitioning accordingly.

- All table data is stored on high performance SSD disk drives.
- Applications can connect to the Amazon DynamoDB service endpoint and submit requests over HTTP/S to read and write items to a table or even to create and delete tables.

Provisioned Capacity

When you create an Amazon DynamoDB table, you are required to provision a certain amount of read and write capacity to handle your expected workloads.

Amazon Redshift

Amazon Redshift is a fast, powerful, fully managed, petabyte-scale data warehouse service in the cloud. Amazon Redshift is a relational database designed for OLAP scenarios and optimized for high-performance analysis and reporting of very large datasets. Traditional data warehouses are difficult and expensive to manage, especially for large datasets. Amazon Redshift not only significantly lowers the cost of a data warehouse, but it also makes it easy to analyze large amounts of data very quickly.

Amazon Redshift gives you fast querying capabilities over structured data using standard SQL commands to support interactive querying over large datasets. With connectivity via ODBC or JDBC, Amazon Redshift integrates well with various data loading, reporting, data mining, and analytics tools. Amazon Redshift is based on industry-standard PostgreSQL, so most existing SQL client applications will work with only minimal changes.

Amazon Redshift manages the work needed to set up, operate, and scale a data warehouse, from provisioning the infrastructure capacity to automating ongoing administrative tasks such as backups and patching. Amazon Redshift automatically monitors your nodes and drives to help you recover from failures.

Clusters and Nodes

The key component of an Amazon Redshift data warehouse is a cluster. A cluster is composed of a leader node and one or more compute nodes. The client application interacts directly only with the leader node, and the compute nodes are transparent to external applications.

Launching RDS instance:

1. Log on to AWS account using the IAM credentials, and from the AWS Management Console, select the RDS option under Database.



2. Select the **Engine**.

The screenshot displays the Amazon RDS console's 'Select Engine' step. On the left, a vertical list of engine logos is shown: Amazon Aurora, MySQL, MariaDB, PostgreSQL, ORACLE, and Microsoft SQL Server. The MySQL engine is currently selected, indicated by a blue highlight. To the right of the MySQL logo, the text 'MySQL' and 'MySQL Community Edition' is displayed, followed by a blue 'Select' button. Below this, a description states: 'MySQL is the most popular open source database in the world. MySQL on RDS offers the rich features of the MySQL community edition with the flexibility to easily scale compute resources or storage capacity for your database.' A bulleted list of features follows: 'Supports database size up to 6 TB.', 'Instances offer up to 32 vCPUs and 244 GiB Memory.', 'Supports automated backup and point-in-time recovery.', 'Supports cross-region read replicas.', and 'Free tier eligible'. Below the MySQL section, the Amazon Aurora MySQL-compatible edition is also visible, with its own 'Select' button and a list of features including throughput, storage, replication, and failover times.

3. As we discussed earlier, we have six relational db engines are available with amazon RDS, Now am going to launch MySQL.

- If you want to use **Free Tier eligibility** make sure you select the tick mark under Select engine option.

Step 1: Select Engine

☒ Free tier eligible only ⓘ

4. If you don't want to get charged or want to use free tier eligibility make sure you select this option **“Only show options that are eligible for RDS Free Tier”**

Free Tier


The Amazon RDS Free Tier provides a single db.t2.micro instance as well as up to 20 GB of storage, allowing new AWS customers to gain hands-on experience with Amazon RDS. Learn more about the RDS Free Tier and the instance restrictions [here](#).

☒ Only show options that are eligible for RDS Free Tier

5. I want to use free tier for the DB instance, so selecting MySQL community edition, in next we have to specify the DB Details.

Instance Specifications

DB Engine	mysql
License Model	general-public-license ▼
DB Engine Version	mysql 5.6.37 ▼

 Review the **Known Issues/Limitations** to learn about potential compatibility issues with specific database versions.

DB Instance Class	db.t2.micro — 1 vCPU, 1 GiB RAM ▼
Multi-AZ Deployment	No ▼
Storage Type	General Purpose (SSD) ▼
Allocated Storage*	20 GB

Settings

DB Instance Identifier*	mysqldatabase
Master Username*	masteruser
Master Password*
Confirm Password*

- **DB Engine:** We have selected the Db Engine as MySQL.

- **License Model:** MySQL databases have only one license model; that is, generalpublic-license. AWS provides the required license keys for your databases, so you don't have to separately purchase one.
- **DB Engine Version:** Select the appropriate DB Engine Version as per your requirements. RDS provides and supports a variety of database engine versions that you can choose from.
- **DB Instance Class:** We have multiple DB Instance Classes with various configurations (vCPU & RAM), Select the appropriate one as per requirement.
- **Multi-AZ Deployment:** Select "Yes/No" for Multi-AZ based on requirement.
- **Storage Type:** Select the Storage Type between **"General purpose SSD"** and **"Provisioned IOPS"**.
- **Allocated Storage:** We can allocate the storage for db instance. We can select from **20 GB to 6TB**.
- **DB Instance Identifier:** Give a valid name for the DB instance and this must be unique in the selected region.
- **Master Username:** Give a valid username to login to the Db instance.
- **Master Password:** Give a valid password for the master username.

6. In Step 3, we need to Configure Advanced Settings.

Configure Advanced Settings

Network & Security



VPC*	Default VPC (vpc-7d7ab214) ▼
Subnet Group	default ▼
Publicly Accessible	Yes ▼
Availability Zone	No Preference ▼
VPC Security Group(s)	<div>Create new Security Group My-SG (VPC) default (VPC) mysecuritygroup (VPC) ▲▼</div>

- **VPC:** Select the name of the VPC that will host your MySQL DB instance. Here am selecting Default VPC to host this instance.
- **Subnet Group:** Selecting the default Subnet Group.

- **Public Accessible:** Select “**Yes**” if you want EC2 instances and devices outside of the VPC hosting the DB instance to connect to the DB instance. If you select No, Amazon RDS will not get a public IP address to the DB instance, so we cannot connect over internet.
- **Availability Zone:** We can select the desired AZ based on the region.
- **VPC Security Groups:** We have to attach a security group to the Db instance. It works same as the EC2 instance security group, As we are launch **MySQL port number 3306** must be opened. For **MsSQL port number is 1433**.

Database Options

Database Name	<input type="text"/>
<small>Note: if no database name is specified then no initial MySQL database will be created on the DB Instance.</small>	
Database Port	<input type="text" value="3306"/>
DB Parameter Group	<input type="text" value="default.mysql5.6"/>
Option Group	<input type="text" value="default:mysql-5-6"/>
Copy Tags To Snapshots	<input type="checkbox"/>
Enable IAM DB Authentication	<input type="text" value="No Preference"/>
Enable Encryption	<input type="text" value="No"/>

- **Database Name:** Provide a suitable database name here. RDS will not create and initialize any database unless you specify a name here.
- **Database Port:** Provide the port number using which you wish to access the database. MySQL’s default port number is 3306. We cannot change the default port number after db instance launch.
- **DB parameter Group:** DB parameter groups are logical groupings of database engine configurations that you can apply to one or more DB instances at the same time. Go with the default option here.
- **Option Group:** This option is similar to DB parameter groups in that they too provide and support few additional configuration parameters that make it easy to manage databases
- **Copy Tags To Snapshots:** Give a tick on checkbox if you want to copy the tags to created snapshots of the db instance.

- **Enable IAM DB Authentication:** We can use IAM users to use the db, but the IAM user need to have appropriate permissions. Select **“Yes”** to manage your database user credentials through AWS IAM users and roles.
- **Enable Encryption:** RDS provides standard AES-256 encryption algorithms for encrypting data at rest. T2.micro will not support the encryption.

Backup

Please note that automated backups are currently supported for InnoDB storage engine only. If you are using MyISAM, refer to detail [here](#).

Backup Retention Period days

Backup Window

Start Time : UTC

Duration hours

- We can set the Backup Retention Period as well as the Backup window's Start Time and Duration. As discussed above if we enable amazon creates automated backups.

Monitoring

Enable Enhanced Monitoring

Maintenance

Auto Minor Version Upgrade

Maintenance Window

Start Day

Start Time : UTC

Duration hours

- **Enable Enhanced Monitoring:** We can use Cloudwatch to monitor the db instances, give yes if you want to change the default monitoring period to detailed monitoring.
- **Auto Minor Version upgrade:** Specify Yes to enable automatic upgrades to new minor versions as they are released. The automatic upgrades occur during the maintenance window for the DB instance.
- **Maintenance Window:** We can select the period in which you want pending modifications or patches applied to the DB instance by Amazon RDS. Any such maintenance should be started and completed within the

selected period. If you do not select a period, Amazon RDS will assign a period randomly.

7. After configuring all the above steps, choose Launch DB instance option. DB instance creation will be initiate now.

Step 1: [Select Engine](#)

Step 2: [Specify DB Details](#)

Step 3: [Configure Advanced Settings](#)

✓ **Your DB Instance is being created.**

Note: Your instance may take a few minutes to launch.

Launch DB Instance Show Monitoring Instance Actions

Filter: All Instances Search DB Instances...

	Engine	DB Instance	Status	CPU	Current Activity	Maintenance	Class
MySQL	mysqldb	creating			None	db.t2.micro	

Filter: All Instances Search DB Instances...

Engine	DB Instance	Status	CPU
MySQL	mysqldb	modifying	1.69%

Engine	DB Instance	Status	CPU	Current Activity
MySQL	mysqldb	backing-up	12.50%	0 Connections

Filter: All Instances Search DB Instances...

Engine	DB Instance	Status	CPU	Current Activity
MySQL	mysqldb	available	1.53%	0 Connections

We have four steps for instance launch stage: Creating, Modifying, Backing-Up and Available.

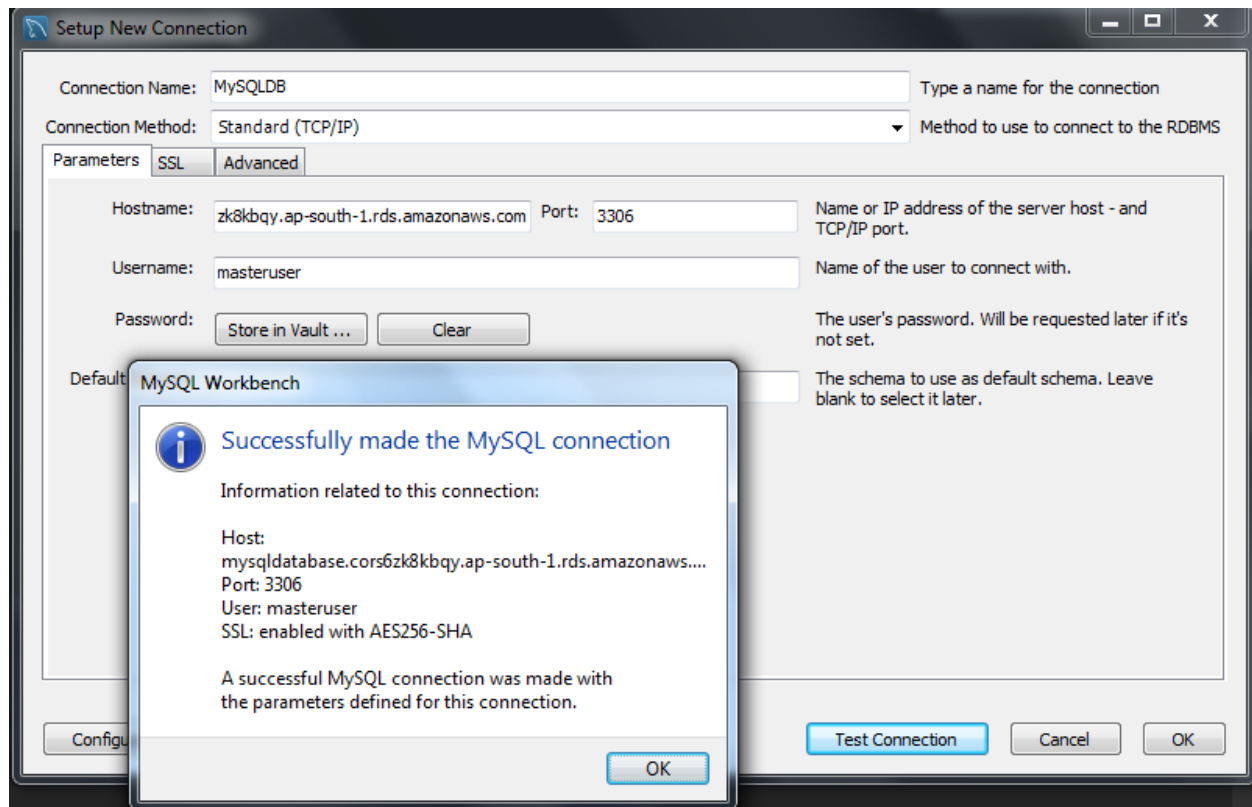
Creating: This is the first stage of any DB instance's lifecycle where the instance is actually created by RDS. During this time, your database will remain inaccessible.

Modifying: This state occurs whenever the DB instance enters any modifications either set by you or by RDS itself.

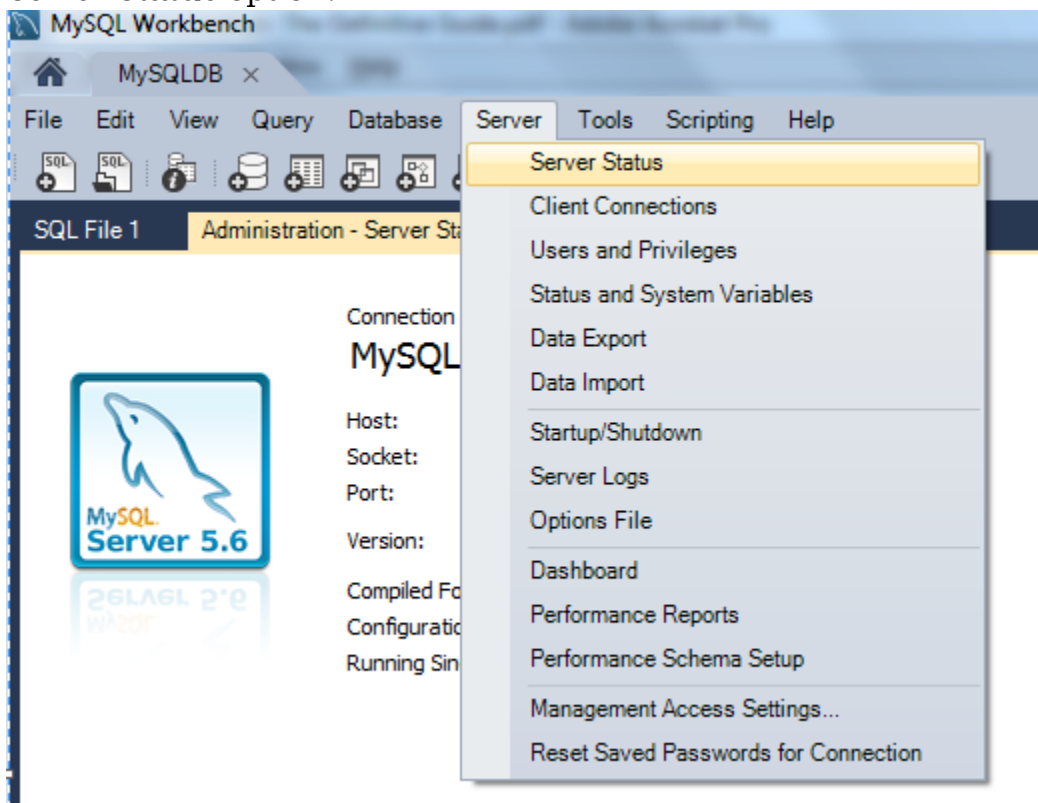
Backing-up: RDS will automatically take a backup of your DB instance when it is first created. You can view all your DB instance snapshots using the Snapshots option on the navigation pane.

Available: This status indicates that your DB instance is available and ready for use. You can now access your database remotely by copying the database's endpoint.

8. To test the connectivity we are going to use MySQL Workbench application, Download and install on any of the local machine or EC2 instance if you want to test it in graphical manner. You can download the MySQL workbench from the following URL:
<https://dev.mysql.com/downloads/workbench/>
9. I've copied the Endpoint URL of my DB instance and opened the installed MySQL workbench application and add a connection and give a name for the connection, Enter the **Endpoint name in Hostname field**, port number is **3306**, Enter **username** and click on **Test Connection** and Give the password, you should get a Successful test result.



10. We can verify the Server Status by navigating to Server and selecting the Server Status option.



11. By using the workbench, we can create databases, schemas and we can manage the database graphically.

To test the MySQL from Linux machine, Launch a Linux instance and install the mysql package by running **yum install mysql** option.

After launching the Linux instance, Install mysql package by running

yum install mysql

Then run # **mysql -u <USERNAME> -h <DATABASE_ENDPOINT> -p** and press enter, It'll ask you to enter the password of connecting user, then you can access the mysql database.

```
[root@ip-172-31-24-253 ec2-user]# mysql -u masteruser -h mysqlatabase.cors6zk8k
bqy.ap-south-1.rds.amazonaws.com -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 36
Server version: 5.6.37-log MySQL Community Server (GPL)

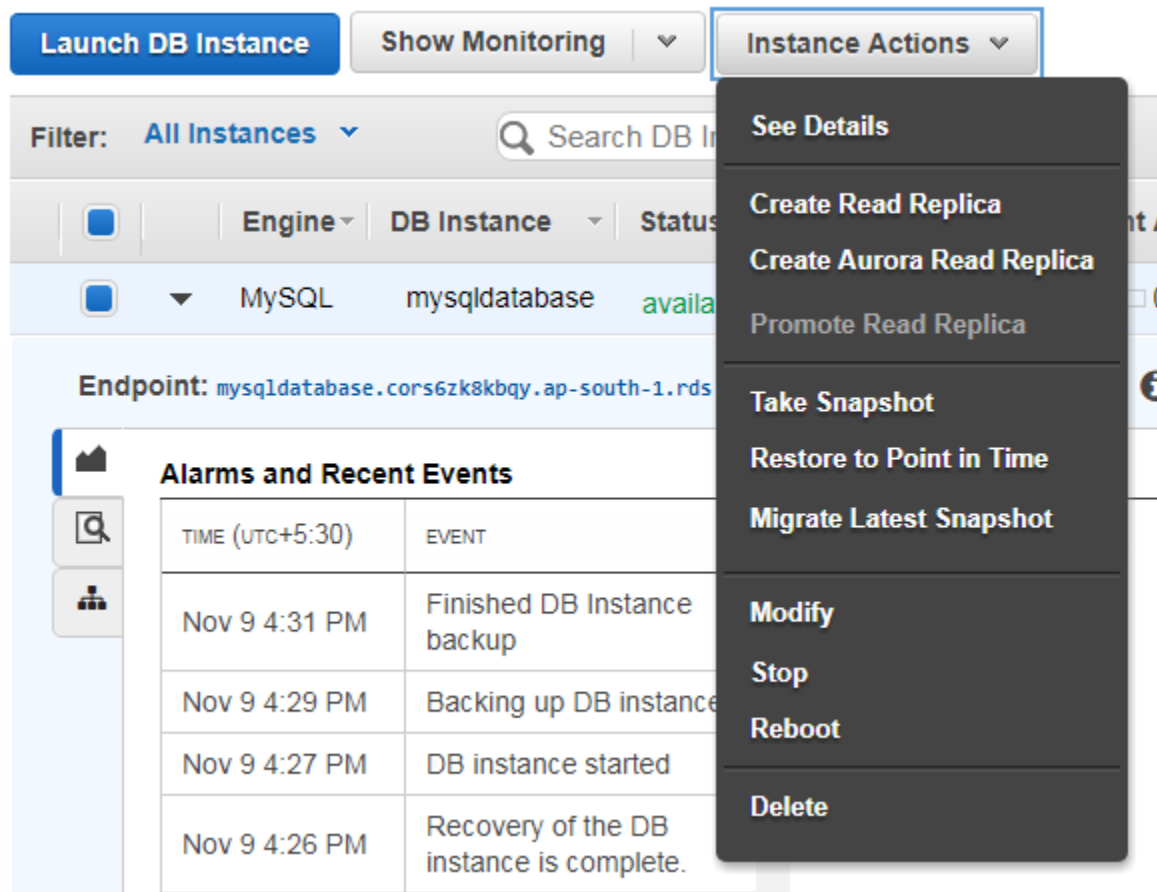
Copyright (c) 2000, 2017, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases
->
```

DB Instance Actions: We can find the below options when you select the **db instance** and choose **Instance Actions** option.



Create Read Replica: As we discussed above, we can create read replicas of the primary db instance for scaling purpose, We'll get a new endpoint for read replicas and the launch wizard is almost same new db instance launch.

Create Aurora Read Replica: If we need a replica with aurora db engine, we can choose this option and follow wizard. Read replica will create with aurora db engine.

Promote Read Replica: If you want to promote read replica to a standalone db instance, we can select this option, But the replication between primary db and read replica will breakdown.

Take Snapshot: For backups of the db instance we can use the snapshots.

Restore to Point in Time: With this option we can create a new DB Instance from a source DB Instance at a specified time. This new DB Instance will have the default DB Security Group and DB Parameter Groups.

Use Latest Restorable Time ☒ November 9, 2017 at 5:30:00 PM UTC+5:30

Use Custom Restore Time ☐ : : UTC+5:30

Migrate Latest Snapshot: We can migrate the selected database to a new DB Engine by selecting desired options for the migrated instance. For mysql “Aurora” and “mariadb”.


Modify: By using modify option, we can change the db instance properties i.e; DB engine version, instance class, storage options, master password, backup retention period and maintenance periods.


Stop: Instance will changes it status to Stopped state, we can start at anytime.

Reboot: underlying instances operating system will reboot.

Delete: Db instance will delete. When you perform delete option AWS will ask you to create a final snapshot. If the data in the db is important, we can take a final snapshot to launch it in future, otherwise we can select No and delete the db instance.

Are you sure you want to Delete the mysqldb database DB Instance?

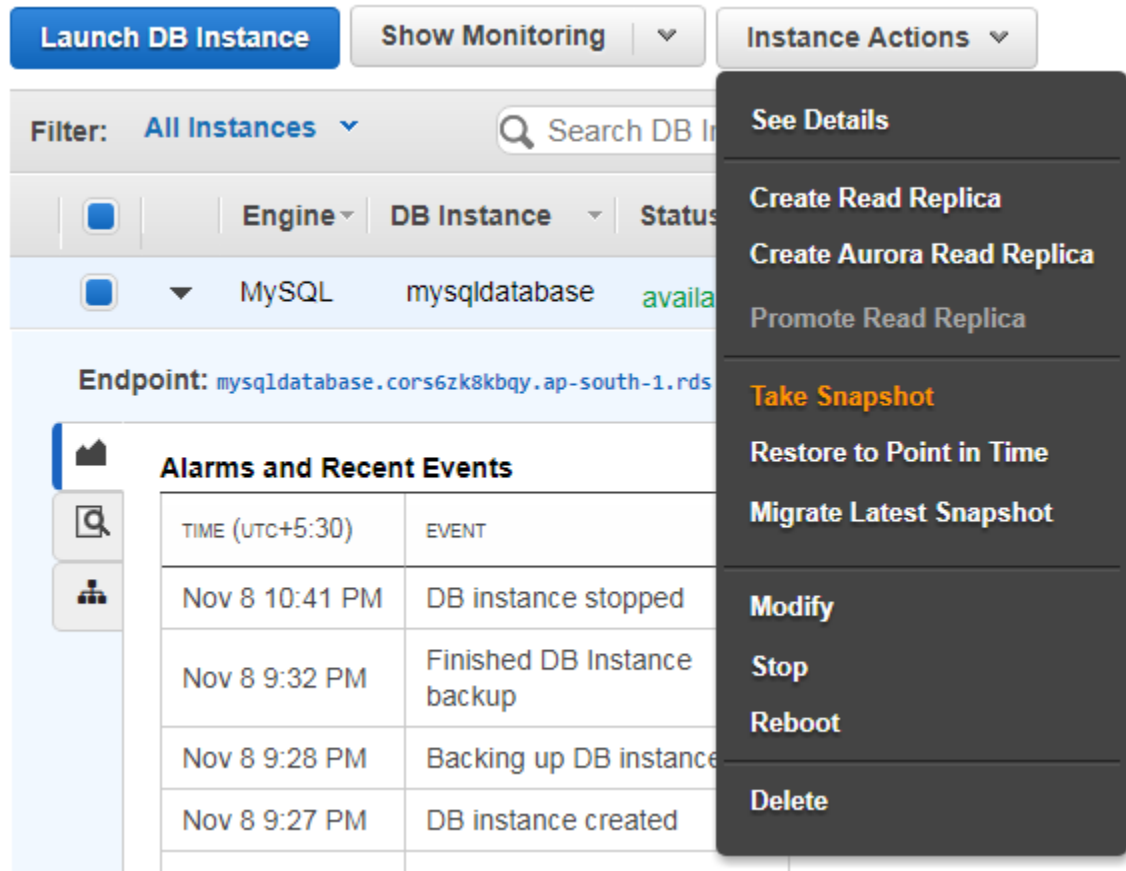
Create final Snapshot? 

Final snapshot name 

DB INSTANCE BACKUP OPTIONS:

As we discussed above, we have two options for backing up 1. Automated backup and 2. Manual Snapshots.

To create a manual snapshot, select the “**instance Actions**” and choose “**Take Snapshot**” Option.



The screenshot shows the AWS Management Console interface for a MySQL DB instance. At the top, there are buttons for 'Launch DB Instance', 'Show Monitoring', and 'Instance Actions'. Below these, there's a filter set to 'All Instances' and a search bar. The instance details show it's a MySQL instance named 'mysqldatabase' with a status of 'available'. The 'Endpoint' is 'mysqldatabase.cors6zk8kbqy.ap-south-1.rds.amazonaws.com'. Below this, there's a section for 'Alarms and Recent Events' with a table of events. The 'Instance Actions' dropdown menu is open, showing options like 'See Details', 'Create Read Replica', 'Create Aurora Read Replica', 'Promote Read Replica', 'Take Snapshot' (highlighted in orange), 'Restore to Point in Time', 'Migrate Latest Snapshot', 'Modify', 'Stop', 'Reboot', and 'Delete'.

TIME (UTC+5:30)	EVENT
Nov 8 10:41 PM	DB instance stopped
Nov 8 9:32 PM	Finished DB Instance backup
Nov 8 9:28 PM	Backing up DB instance
Nov 8 9:27 PM	DB instance created

Take DB Snapshot

To take a snapshot of this DB instance you must provide a name for the snapshot. This feature is currently supported for InnoDB storage engine only. If you are using MyISAM, refer to details [here](#).

DB Instance **mysqldatabase** ⓘ

Snapshot Name ⓘ

Give a name for the newly creating Snapshot and here is the status of Snapshot creation.

Create Snapshot

Snapshot Actions

Filter: Owned by Me

Search...

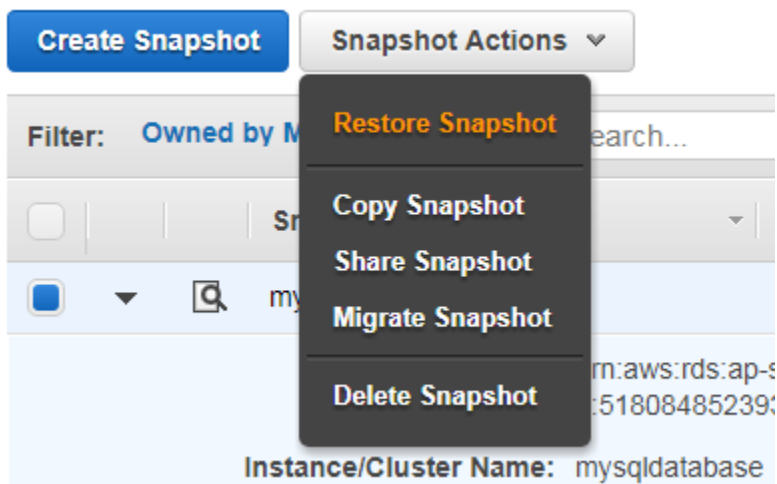
X

Viewing 2 of

<input type="checkbox"/>		Snapshot	DB Instance or Cluster	Snapshot Creation Time	Status	Progress
<input checked="" type="checkbox"/>		mysqlsnapshot	mysqldatabase		creating	0%
<input type="checkbox"/>		rds:mysqldatabase-2017-11-08-15-58	mysqldatabase	Nov 8, 2017, 9:28:19 PM	available	Completed

Launching Instance from the Snapshot.

We can use either automated backups or manual snapshots to launch a new instance, but remember we'll get a new endpoint. To create a snapshot, select the snapshot and choose the **"Snapshot Actions"** and choose **"Restore Snapshot"** option, Then automatically an instance launch wizard will launch. You'll find almost all same as the regular instance launch.



Copy Snapshot: We can make a copy the snapshot in another region. Choose the **"Destination region"** give a name for the new DB snapshot identifier, we can enable the encryption, if required while copying the snapshot.

Make Copy of DB Snapshot?

Source DB Snapshot **mysqlsnapshot** ⓘ

Destination Region **Asia Pacific (Mumbai)** ▼ ⓘ

New DB Snapshot Identifier ⓘ

Target Option Group (Optional) **default:mysql-5-6** ▼ ⓘ

Copy Tags ☒ ⓘ

Enable Encryption **No** ▼ ⓘ

Share Snapshot: We can share the snapshot with any other AWS account user or make it available for public by selecting the Share Snapshot option.

Manage Snapshot Permissions

You are sharing an unencrypted DB Snapshot. When you share an unencrypted DB Snapshot, you can restore a database from your DB Snapshot.

DB Snapshot **mysqlsnapshot**

DB Snapshot Visibility ☒ Private ☐ Public

AWS Account ID

AWS Account ID	Delete
Please add AWS Account ID	

Migrate Snapshot: We can migrate the snapshot to a different db engine by using this option. Choose the Migrate snapshot option and select the **Aurora or Marisdb** and follow the wizard, we'll get a new endpoint with the selected db engine.

Instance Specifications

Migrate to DB Engine

DB Instance Class

Settings

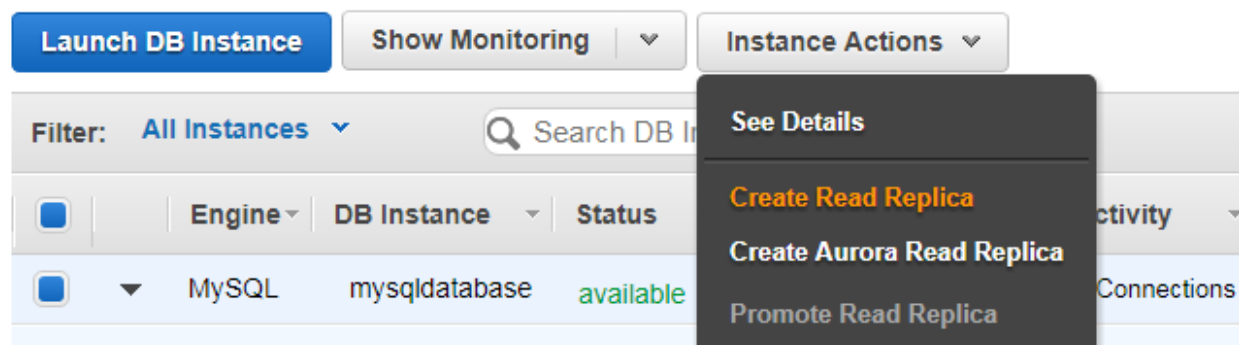
DB Snapshot ID

DB Instance Identifier*

Creating Read Replicas and promoting them

Read replica's allow you to have a read only copy of your production database. This is achieved by using Asynchronous replication from the primary RDS instance to the read replica. You use read replica's primarily for very read-heavy database workloads.

To create a read replica select the Instance Actions tab and select the Create Read Replica option.



Instance Specifications

DB Instance Class db.t2.micro — 1 vCPU, 1 GiB RAM ▼

Storage Type General Purpose (SSD) ▼

Settings

Read Replica Source mysqldatabase ▼

DB Instance Identifier* mysql_readreplica

Network & Security

Destination Region Asia Pacific (Mumbai) ▼

Destination DB Subnet Group default ▼

Publicly Accessible Yes ▼

Availability Zone No Preference ▼

Select the Read replica source and give a name for the replica and choose in what availability zone you want to deploy and even we can select the desired availability zone in the destination region also.

Database Options

Database Port	<input type="text" value="3306"/>
Copy Tags To Snapshots	<input type="checkbox"/>
Enable IAM DB Authentication	<input type="text" value="No"/>

Monitoring

Enable Enhanced Monitoring	<input type="text" value="No"/>
----------------------------	---------------------------------

Maintenance

Auto Minor Version Upgrade	<input type="text" value="Yes"/>
----------------------------	----------------------------------

- Select the appropriate options and click on Create read replica option. Read replica creation will start and we can see the status in dashboard.

Filter: All Instances		Search DB Instances...			
		Engine	DB Instance	Status	CPU
		MySQL	mysqldb	modifying	1.69%

Filter: All Instances ▼

	<input type="checkbox"/>	Engine ▼	DB Instance ▼	Status ▼	CPU	Current Activity ▼
	<input type="checkbox"/>	MySQL	mysqldb	modifying	<div><div></div></div> 1.50%	<div><div></div></div> 0 Connections
	<input checked="" type="checkbox"/>	MySQL	mysqlreadreplica	creating		

Now read replica is created. To verify the master and slave status we can go to details and verify.

Filter: All Instances Search DB Instances...

Engine	DB Instance	Status	CPU	Current Activity	Maintenance
MySQL	mysqldatabase	available	1.33%	1 Connections	None
MySQL	mysqlreadreplica	available	1.48%	0 Connections	None

Endpoint: mysqlreadreplica.cors6zk8kbqy.ap-south-1.rds.amazonaws.com:3306 (authorized)

Endpoint: mysqlreadreplica.cors6zk8kbqy.ap-south-1.rds.amazonaws.com:3306 (authorized)

Replication Details

Read Master: mysqldatabase

Role: replica

DB INSTANCE	ROLE	ZONE	REPLICATION SOURCE	REPLICA LAG
mysqldatabase	master	ap-south-1b	-	-
mysqlreadreplica	replica	ap-south-1b	mysqldatabase	0 Milliseconds

- And we can promote the read replica to a standalone db instance, but this breaks the replication.

To promote a read replica, choose the **“Promote Read Replica”** option from “Instance Actions”

Launch DB Instance Show Monitoring Instance Actions

Filter: All Instances Search DB Instances...

Engine	DB Instance	Status
MySQL	mysqldatabase	available
MySQL	mysqlreadreplica	available

Endpoint: mysqlreadreplica.cors6zk8kbqy.ap-south-1.rds.amazonaws.com:3306 (authorized)

Replication Details Deployment DB Instances

See Details

Create Read Replica

Create Aurora Read Replica

Promote Read Replica

Take Snapshot

Restore to Point in Time

Migrate Latest Snapshot


Promote Read Replica: mysqlreadreplica

Enable Automatic Backups ☒ Yes ☐ No 

The number of days for which automated backups are retained.

Backup Retention Period days 

The daily time range during which automated backups are created if automated backups are enabled

Backup Window ☐ Select Window ☒ No Preference 

- If we want to promote a read replica, we must enable the automated backups and need to select the backup retention period, and you can select the backup window.

We will get a note with the following information:

“Before you promote this Read Replica, we recommend that you stop any transactions on the master and wait for the Read Replica lag to be zero. Otherwise, there is a high likelihood that the Read Replica does not have all the transactions committed to the master DB Instance.

Note that the promotion process takes a few minutes to complete. When you promote a Read Replica, replication is stopped and the Read Replica is rebooted as part of the promotion. In addition, the promotion process is irreversible and you cannot restart the replication with the promoted DB Instance as a replication target.”

Filter: All Instances ▾ Search DB Instances...

	Engine ▾	DB Instance ▾	Status ▾	CPU
	MySQL	mysqldatabase	available	<div><div></div></div> 1.50%
	MySQL	mysqlreadreplica	available	<div><div></div></div> 1.67%

Endpoint: mysqlreadreplica.cors6zk8kbqy.ap-south-1.rds.amazonaws.com:3306 (auth)

No replication details to display.

Now the read replica is promoted as an individual db instance and no replication is enabled with any other db instances.