## **RDS AWS Tutorial**

Today in this RDS AWS Tutorial we shall be discussing in detail about Amazon’s Relational Database Management service RDS AWS and shall also do a hands-on, but first let us understand why it came into existence.

The world is changing, with every idea being converted into an application, millions of new applications go online every day. Now for any application or project to be successful, it should have a unique idea behind it.

Let’s talk about you, you just had the world’s most amazing idea, and you want to create an application around it.

Now imagine yourself 10 years back, when to have the application up and ready, you’d have to setup a back end server, research and install various softwares to support your application, after all of these tiring tasks you would have started developing your application.

Hey, wait! What about its maintenance? You would have to install all the latest security patches and updates for your backend server and also make sure that it stays in a healthy state.

Now, while you were taking care of all that, your app becomes famous overnight, with tons of traffic directed at your application, the need to scale becomes your top most priority, now let’s not even think about the investment you will be making on this task, how will you accomplish this task of scaling up and configuring all these extra servers quickly?

Scary, right? What if i told you, someone will do all these tasks for you, and you just have to focus on your application. Also, at a fraction of the cost that you were investing earlier.

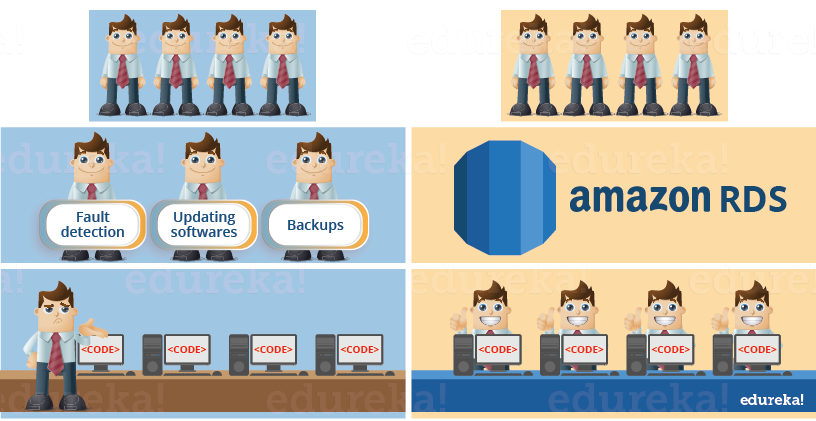
Wouldn’t it be amazing?

Well amazing is, sorry **amazon** is here, Amazon Web Services (AWS) offers a service called RDS AWS (Relational Database Service), which does all these tasks (i.e. setup, operate, update) for you automatically

You just have to select the database that you want to launch, and with just one click you have a back end server at your service which will be managed automatically!

Let’s take an example here, suppose you start a small company.

You want to launch an application which will be backed by a MySQL database and since there is a lot of Database work, there are chances that the development work will fall behind.



Imagine this scene again, with Amazon RDS, the image is self explanatory!

Now this was just an example. For bigger companies where you have a bigger team, which manages your database servers; using RDS, that team can be reduced to a significant number and perhaps be optimally deployed!

Let’s move further in this RDS AWS Tutorial and see how amazon defines their service:

**The Amazon Relational Database Service (RDS AWS)** is a web service that makes it easier to set up, operate, and scale a relational database in the cloud. It provides cost-efficient, re-sizable capacity in an industry-standard relational database and manages common database administration tasks.

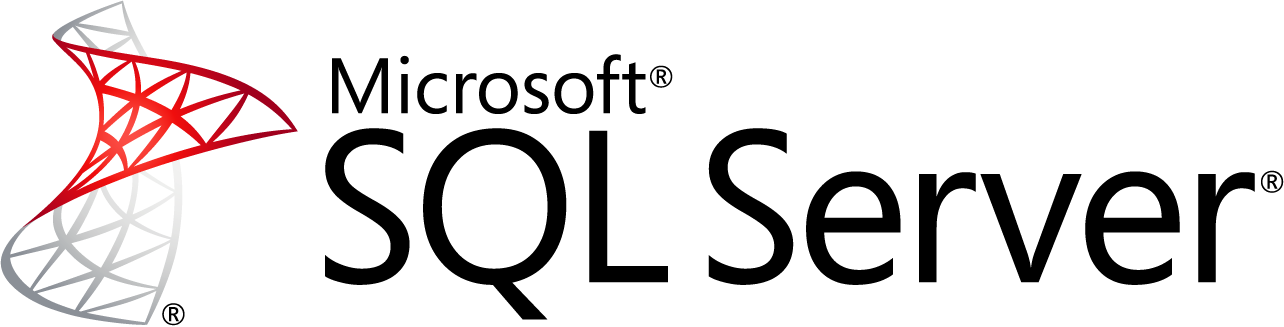
So people often develop a misconception, when they confuse RDS with a database.

RDS is **not** a database, it’s a service that manages databases, having said that, let’s discuss the databases that RDS can manage as of now:

It is a relational database engine made by amazon which combines the speed and reliability of high-end commercial databases with the simplicity and cost-effectiveness of open source databases. Amazon claims that Aurora is 5x faster than RDS MySQL.

It is an open source database management system which uses SQL (Structured Query Language) to access the data stored in its system.

PostgreSQL is yet another open source database management system which uses SQL to access the data.

SQL Server is a Relational Database Management System, which was developed by Microsoft in 2005 for the enterprise environment.

It is object-relational database management system which was developed by Oracle Inc.

MariaDB is a community developed **fork** of MySQL DBMS. The reason for its fork, was the concern over the acquisition of Oracle over MySQL

**Fork** means copying the source code of the original application, and starting development over the new application.

The interesting part is, DB engines that RDS support are existing Relational Databases therefore, you don’t have to change the code of your application or learn a new querying language for using RDS in your already existing application.

Now you may be wondering what is the difference between, say a normal MySQL and a MySQL which is managed by RDS.

Therefore, in terms of usage you will be using it as if, you were using your own database, but now, you as a developer will not be worried about the underlying infrastructure or the administration of the database. The updation, the health monitoring of the system on which your SQL is installed, taking regular backups, etc., all these tasks will be managed by RDS AWS.

AWS also offers EC2 relational database AMIs, now you may ask why one more relational database service when we already have AWS RDS?

So, EC2 relational database AMIs allow you to fully manage your own relational databases on AWS Infrastructure, where as RDS manages them for you. So, depending on your use case you can choose an AWS service. Hope, it is clear to you now!

Moving along in this RDS AWS Tutorial, let’s discuss the components of RDS.

## **RDS AWS Components:**

* DB Instances
* Regions and Availability Zones
* Security Groups
* DB Parameter Groups
* DB Option Groups

Let’s discuss each one of them in detail:

**DB Instances**

* They are the building blocks of RDS. It is an isolated database environment in the cloud, which can contain multiple user-created databases, and can be accessed using the same tools and applications that one uses with a stand-alone database instance.
* A DB Instance can be created using the AWS Management Console , the Amazon RDS API, or the AWS Command line Interface .
* The computation and memory capacity of a DB Instance depends on the DB Instance class. For each DB Instance you can select from 5GB to 6TB of associated storage capacity.
* The DB Instances are of the following types:
  + Standard Instances (m4,m3)
  + Memory Optimised (r3)
  + Micro Instances (t2)

**Regions and Availability Zones**

* The AWS resources are housed in highly available data centers, which are located in different areas of the world. This “area” is called a region.
* Each region has multiple Availability Zones (AZ), they are distinct locations which are engineered to be isolated from the failure of other AZs.
* You can deploy your DB Instance in multiple AZ, this ensures a failover i.e. in case one AZ goes down, there is a second to switch over to. The failover instance is called a standby, and the original instance is called the primary instance.

**Security Groups**

* A security group controls the access to a DB Instance. It does so by specifying a range of IP addresses or the EC2 instances that you want to give access.
* Amazon RDS uses 3 types of Security Groups:
* VPC Security Group
  + It controls the DB Instance that is inside a VPC.
* EC2 Security Group
  + It controls access to an EC2 Instance and can be used with a DB Instance.
* DB Security Group
  + It controls the DB Instance that is not in a VPC.

**DB Parameter groups**

* It contains the engine configuration values that can be applied to one or more DB Instances of the same instance type.
* If you don’t apply a DB Parameter group to your instance, you are assigned a default Parameter group which has the default values.

**DB Option groups**

* Some DB engines offer tools that simplify managing your databases.
* RDS makes these tools available with the use of Option groups.

**RDS AWS Advantages**

Let’s talk about some interesting advantages that you get when you are using RDS AWS,

* So usually when you talk about database services, the CPU, memory, storage, IOs is bundled together, i.e. you cannot control them individually, but with AWS RDS, each of these parameters can be tweaked individually.
* Like we discussed earlier, it manages your servers, updates them to the latest software configuration, takes backup, everything automatically.
* The backups can be taken in two ways
  + The automated backups where in you set a time for your backup to be done.
  + DB Snapshots, where in you manually take a backup of your DB, you can take snapshots as frequently as you want.
* It automatically creates a secondary instance for a failover, therefore provides high availability.
* RDS AWS supports **read replicas** i.e. snapshots are created from a source DB and all the read traffic to the source database is distributed among the read replicas, this reduces the overall overhead on the source DB.
* RDS AWS can be integrated with IAM, for giving customized access to your users who will be working on that database.

The updates to your database in RDS AWS are applied in a **maintenance window**. This maintenance window is defined during the creation of your DB Instance, the way it functions is like this:

* When an update is available for your DB you get a notification in your RDS Console you can take one of the following actions
  + Defer the maintenance items.
  + Apply maintenance items immediately.
  + Schedule a time for those maintenance items.
* Once maintenance starts, your instance has to be taken offline for updating it, if your instance is running in Multi-AZ, in that case the standby instance is updated first, it is then promoted to be a primary instance, and the primary instance is then taken offline for updating, this way your application does not experience a downtime.
* If you want to scale your DB instance, the changes that make to your DB instance also happen during the maintenance window, you can also apply them immediately, but then your application will experience a downtime if its in a Single-AZ.

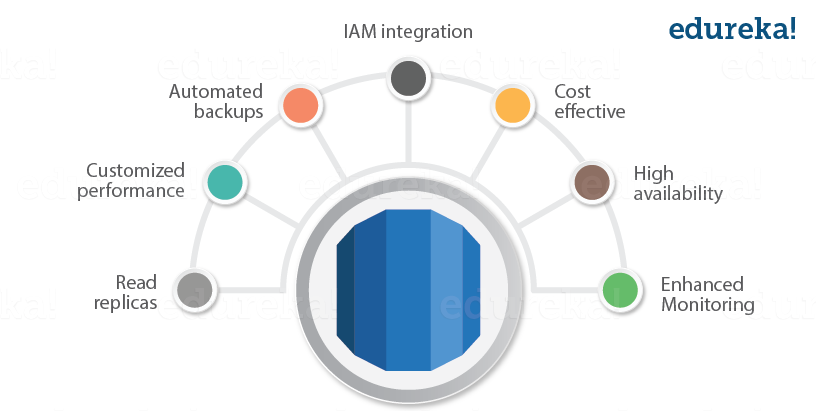


Fig. RDS AWS Advantages

## **Pricing**

RDS AWS is billed based on the following parameters:

* **Instance Class** i.e. the type of instance that you are choosing.
* **Running Time** i.e. the amount of time an instance is running, partial hours are billed as full hours.
* **Storage** i.e. the amount of storage that you have provisioned to your DB Instance
* **I/O Requests per Month** i.e. the I/O requests that are made to your DB Instance per month
* **Data Transfer**: Data transfer in and out of your DB Instance.

Another way of getting billed for AWS RDS is by reserving some instances.

**Reserved Instance** is also a way of using AWS RDS, in this you reserve an RDS Instance for a term, which can be for one or three years by making a one time payment, it is a less expensive way compared to the monthly bill that one pays.

### **Free Tier**

AWS has an amazing free tier usage for most of its services, so that the customer can first use the service and then do the needful.

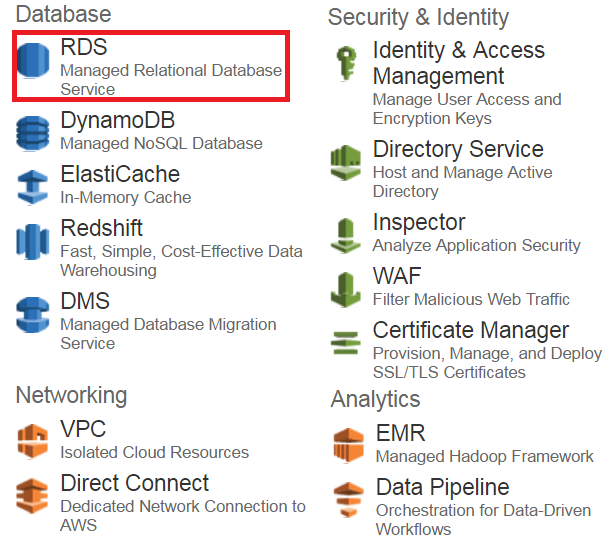
Similarly it offers free tier usage for RDS AWS, which includes the following benefits:

* 750 hours of Amazon RDS usage in single-AZ for db.t2.micro instance, every month for one year from signup.
* 20 GB of DataBase Storage: any combination of General Purpose (SSD) or Magnetic storage.
* 10 million IOs
* 20GB of backup storage

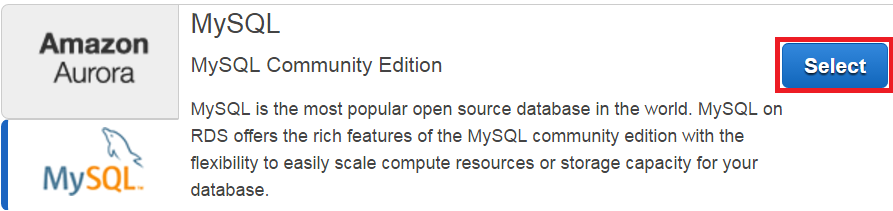
Enough of theory, lets make this RDS AWS Tutorial more interesting, let’s now launch a MySQL DB in RDS.

## **Hands-On**

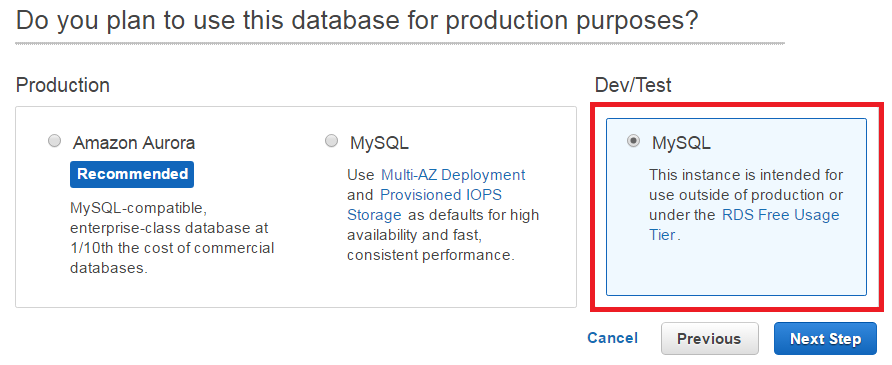
**Step1:** First select the RDS service from the AWS Management Console.



**Step 2:** Since we will be launching a MySQL instance, select the MySQL instance from the list of Dbs. Moving forward in this RDS AWS Tutorial, let’s go to Step 3.

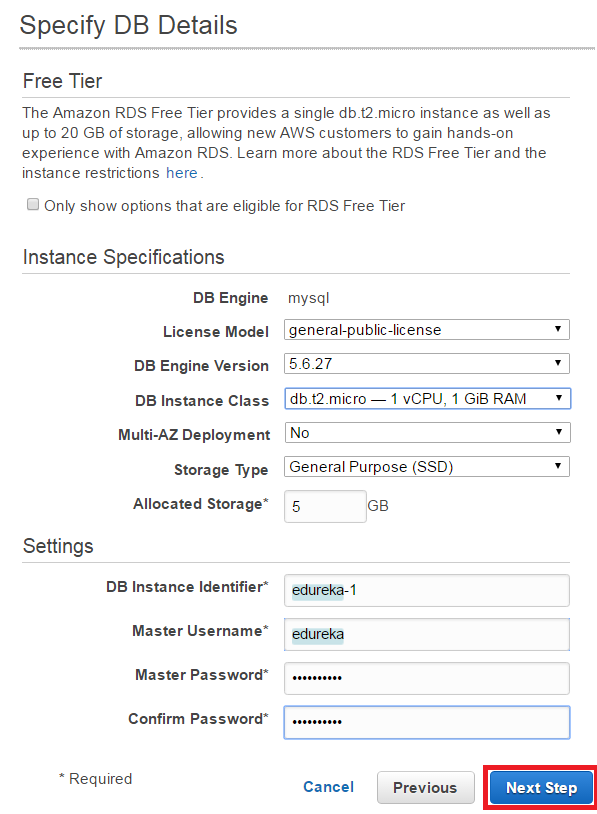


**Step 3:** Since we are creating this instance for demo purposes, we will be selecting Dev/Test option and click on Next Step.



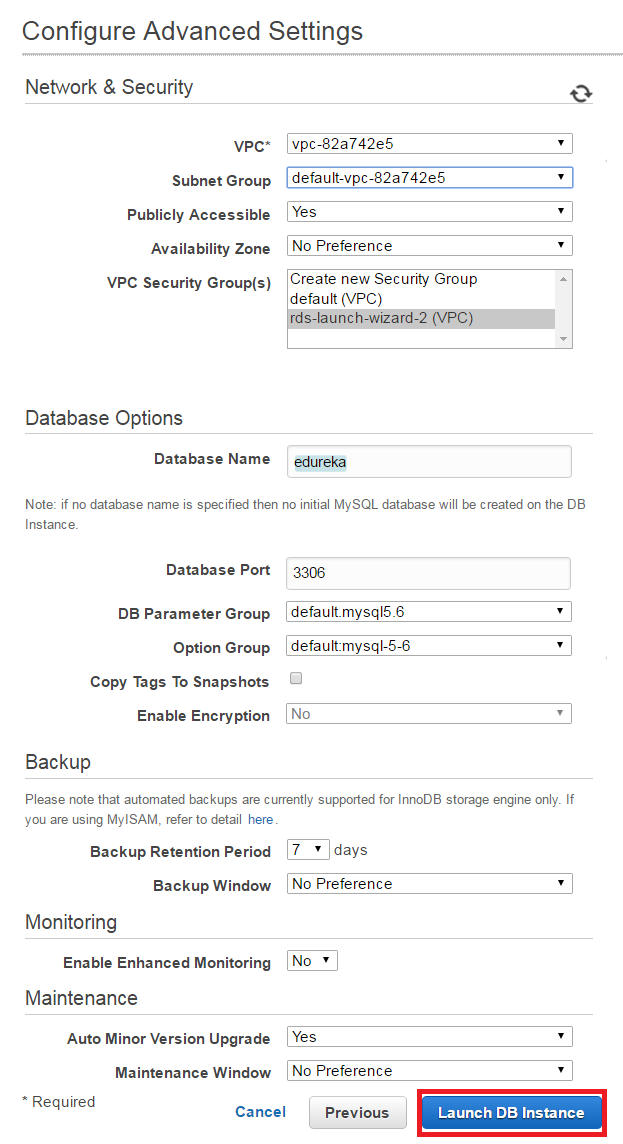
**Step 4:** On the next page you will be filling the following details:

* You can select your desired Db instance here
* You can select whether you want Multi-AZ enabled in your MySQL Db.
* You can select how much space you want to allocate to your Db instance, it can vary from 5GB to 6TB.
* In the end you will be setting your username and password for your Db Instance



**Step 5:** In the next step, you will be configuring Advanced Settings for your DB

* You will be selecting the VPC here, if you do not wish to launch your instance in a VPC you can leave the default settings and move ahead.
* In the next section you can select which version of the Db you want to use, for our example we are using MySQL 5.6
* In the next section you can set your backup preferences, like the retention period etc.
* After that we will be setting the maintenance window, this is the time frame during which your Db instances will be updated.
* Once you fill all the details, you will be launching the Db instance!



**Congratulations!** You have successfully launched your first RDS Db Instance!