



Global Knowledge®

Expert Reference Series of White Papers

Introduction to Amazon Relational Database Service (Amazon RDS)

Introduction to Amazon Relational Database Service (Amazon RDS)

Jon M. Gallagher, Global Knowledge Instructor, Certified AWS Solutions Architect, Certified AWS SysOps Administrator, Authorized Amazon Instructor

Introduction

Database Management Systems (DBMS) are an integral part of almost every large-scale software system. DBMS are large, complex software suites that not only store and retrieve data, but also secure the data, allow backups of the data, replicate the data across multiple systems for greater reliability, and cache the data for faster access.

Such large database systems are difficult to set up, maintain, and expand. They are also challenging to clone, which is a critical step that allows Development (Dev) and Quality Assurance (QA) departments to use the same environment for developing and testing an organization's products.

Meanwhile, companies are finding that the data they capture is becoming more and more valuable to their business, either as a way of measuring and improving their own operations, or as a product that can be sold. The process of extracting value from the data becomes complicated, however. This is because as the data becomes more valuable, more care should be taken with the DBMS infrastructure—and yet the people who are involved in running and maintaining the DBMS are the ones who can best help with exploiting the data for the business. Amazon Relational Database Service (Amazon RDS) was created by Amazon Web Services (AWS) out of its own experience with these DBMS complications. Amazon RDS provides cost-effective DBMS deployment, quick and efficient scaling, and easy support of development and QA.

This paper describes how you can set up Amazon RDS and use it as a drop-in replacement for traditional DBMS, with all the cost, scaling, and agility advantages of deploying software in the cloud.

Problems with Traditional DBMS

Because traditional DBMS are so large and complex, you can run into problems with them at any stage of the software lifecycle: Production, Dev, and QA. This section first describes some of these problems in the order in which you're most likely to encounter them (Production, then in the QA-Dev cycle), and then discusses how Amazon RDS can help mitigate them.

Problems with Production DBMS

Databases—particularly large-scale databases—are at the heart of successful web applications and critical enterprise software. Any company that uses a database, or provides database services, finds itself depending on the database either for critical support of its operations or for its competitive business advantage. But there is a usually a long and involved process in setting up the DBMS that support these databases, particularly if the databases need to scale across hundreds, thousands, or even millions of users, and/or across continents or time zones, all while guaranteeing reliability.

Traditionally, DBMS have been monolithic structures with their own dedicated hardware, storage arrays, and consoles. Proper sizing of the hardware infrastructure has been a capital-intensive—but largely opaque—process. Initial configurations are based on educated guesses about system load and user needs, and lock in thousands of dollars of capital budget. Of course, these guesses almost always get at least one aspect of the deployment wrong, meaning, it requires more design time and expenditures on additional memory, bigger CPUs, and/or more disk space.

In growing its own infrastructure and then developing AWS, Amazon realized that while each company can use unique methods of collecting and using data, the actual processes of building the management infrastructure—and the problems inherent in growing that infrastructure—are almost always the same. Werner Vogels, the CTO of AWS, calls these processes “undifferentiated heavy lifting,” meaning, tasks that every organization must perform but they impart no business advantage.

These processes encompass the whole spectrum of database management: you must choose the right software and hardware to run, install the latest version of software, configure the right security levels to access the hardware and software, get the entire system onto the network, and finally, make the system accessible as a data store.

Once the DBMS is running, the operational challenges start: you must make sure the database is backed up while running read replicas of the database to increase access speed and increasing capacity as the system grows. If you decide that you need high availability, there are additional complications: you must replicate the data onto separate hardware platforms, and you must detect the failure of the main server and re-route traffic to the replicated server.

These processes are not everyday tasks; in fact, setting up a new database on a new machine is a relatively rare occurrence for most organizations, as would be setting up replication, and so on. Because these processes happen rarely, they cause a different sort of problem: when it is time to perform them again, you or your team needs to re-learn them and/or update their skill set to the latest release. This means that critical configuration tasks—which can cause their own problems, and sometimes ones that do not manifest themselves until well down the road—are being done by relative rookies, every time.

Problems with Dev and QA DBMS

The previous section described problems frequently encountered when running a production database using a DBMS. What about databases that support Dev or QA departments? If Production support people are necessary to support DBMS for Dev and QA, then the priority for this support may fall behind that of keeping Production running. But if Production support doesn't help with Dev and QA, when (and how) will the Production team learn what changes need to be made to support new software releases? How can the smooth transition of database changes made by Dev to QA to Production be assured?

Amazon RDS Remedies These Problems

AWS provides an answer to these complicated questions with its Amazon RDS. Basically, Amazon RDS is just a replacement to running your own database server. With a simple sequence of commands at the console, you can choose from two commercial DBMS (Oracle or Microsoft SQL Server), or two open source DBMS (MySQL or PostgreSQL). You choose the size of computer to run the software on, the size of storage for the system, and whether you want AWS to run a replicated server in another Availability Zone (AZ).

For non-production purposes, Amazon RDS provides an interface to quickly clone Production databases and safely run Dev and QA versions of production data.

At its core, the strength of Amazon RDS is the flexibility it provides you to change the system as you learn more about it. Your initial guesstimates of storage, capacity, and usage can be validated quickly, and if the infrastructure needs to be modified based on what you've learned, the changes can be made quickly and will leave no expensive capital equipment idle.

The rest of this white paper describes how to deploy with Amazon RDS, how using Amazon RDS can help with challenges that arise in Production, and how Amazon RDS can help with the Dev and QA of your software.

Deploying with Amazon RDS

The normal process for deploying a DBMS is:

- Choose the system
- Obtain the software:
 - If commercial software, purchase license for estimated usage
 - If open source software, choose the version and download the software
- Specify, purchase, install, and configure the platform (hardware and disk storage)
- Install the DBMS
- Configure the administrative user and the initial database
- Secure the system

Amazon RDS makes this process easier, cheaper, and much more scalable, as described in the rest of this section.

Choosing the System

As shown in the following screenshot, Amazon RDS has four DBMS to choose from:










- Oracle (commercial)
- Microsoft SQL Server (commercial)
- MySQL (open source)
- PostgreSQL (open source)

Amazon RDS provides a Free Usage Tier that allows you to test these products. This free tier lets you use, per month:

- Up to 750 hours of a micro server (the smallest)
- 10 million I/Os
- 20GB of storage
- 20GB of backups

Engine Selection

To get started, choose the DB Instance details below and click Select

	mysql MySQL Community Edition	Select
	postgres PostgreSQL	Select
	oracle-se1 Oracle Database Standard Edition One	Select
	oracle-se Oracle Database Standard Edition	Select
	oracle-ee Oracle Database Enterprise Edition	Select
	sqlserver-ex Microsoft SQL Server Express Edition <i>Note that SQL Server Express Edition limits the storage of per database to a maximum of 10GB. Refer to this link for more details.</i>	Select
	sqlserver-web Microsoft SQL Server Web Edition <i>Note that in accordance with Microsofts licensing policies, SQL Server Web Edition can only be used to support public and internet accessible Web pages, Websites, Web applications and Web services. Refer to the AWS Service Terms for more details.</i>	Select
	sqlserver-se Microsoft SQL Server Standard Edition	Select
	sqlserver-ee Microsoft SQL Server Enterprise Edition	Select

[Cancel](#)

Obtaining the Software

Amazon RDS maintains multiple versions of all DBMS that it supports, and allows you to specify the particular major and minor version of currently supported software to run.

As shown in the following screenshot, each commercial DBMS that Amazon RDS supports has two license models:

- **Bring Your Own License (BYOL).** If you already have a license for an equivalent server, you can specify that the server run under BYOL. When you configure the server in Amazon RDS, you are asked for the key.
- **License Included (Pay as you go).** The cost of the license is folded into the hourly charge for running the server. You pay for the server only when the server is turned on and running. Stopping your server stops all charges.

DB Instance Details

To get started, choose a DB engine below and click Next Step

DB Engine: oracle-se1

License Model: ☒ - Select One -
bring-your-own-license
license-included

DB Engine Version:

DB Instance Class: - Select One -

Multi-AZ Deployment: - Select One -

Auto Minor Version Upgrade: ☒ Yes ☐ No

Provide the details for your RDS Database Instance.

Allocated Storage:* GB (Minimum: 10 GB, Maximum: 3072 GB) Higher allocated storage may improve IOPS performance.

Use Provisioned IOPS: ☐

DB Instance Identifier:* (e.g. mydbinstance)

Master Username:* (e.g. awsuser)

Master Password:* (e.g. mypassword)

[Cancel](#) [Previous](#) [Next Step](#)

Specifying the Platform

Specifying the platform is an expensive process for a traditional DBMS. Capital gets tied up in hardware that is based on best estimates of initial usage. Because the procurement process for many organizations is lengthy, the estimate for initial usage must also account for any growth that might occur before upgrades can be obtained.

With Amazon RDS, you customize the size of machine to run the server on, and the size of disk to store the data on.

First, you pick the server type. Each server type offered by AWS is optimized for data throughput. The following screenshot shows the current choices.

DB Instance Details

To get started, choose a DB engine to create, and then select a DB instance class.

DB Engine:	- Select One -
License Model:	db.t1.micro
DB Engine Version:	db.m1.small
DB Instance Class:	db.m1.medium
Multi-AZ Deployment:	db.m1.large
Auto Minor Version Upgrade:	db.m1.xlarge
	db.m2.xlarge
	db.m2.2xlarge
	db.m2.4xlarge
	db.cr1.8xlarge

Provide the details for your RDS Database Instance.

Allocated Storage: * 100 GB (Minimum: 100 GB, Maximum: 3072 GB)

Use Provisioned IOPS: ☒ Use m1.large or [larger](#) instances for best results.

Provisioned IOPS: 1000 RDS MySQL supports IOPS / GB ratios between 3 and 10

For a workload with 50% writes and 50% reads running on a cr1.8xlarge instance, you can realize up to 21,000 IOPS. However, by provisioning more than this limit, you may be able to achieve lower latency and higher throughput. Your actual realized IOPS may vary from the amount you provisioned based on your database workload and instance type. Refer to the **Factors That Affect Realized IOPS** section to learn more.

DB Instance Identifier: * (e.g. mydbinstance)

Master Username: * (e.g. awsuser)

Master Password: * (e.g. mypassword)

[Cancel](#) [Previous](#) [Next Step](#)

Next, you pick the size of storage for the data. This storage is Amazon's Elastic Block Store (Amazon EBS), which is a highly available network-attached storage system. This means that the data is highly available and that input/output (I/O) operations can be provisioned on a per-second basis (PIOPS). You can configure the rate at which the Amazon RDS storage responds to read and write operations to match the rate your software requires.

Configuring the Administrative User and Initial Database

In the last part of the DBMS Server Size screen (see above), you are prompted to name the instance you are using, create an administrative user, and specify the password for the user.

Configuring Access to the Database

Next, you set the initial database, the communications protocol to use with the DBMS, and the security groups that specify who may access the DBMS, as shown in the following screenshot.

Additional Config

Provide the optional additional configuration details below.

Database Name: (e.g. mydb)

Note: if no database name is specified then no initial MySQL database will be created on the DB Instance.

Database Port:

Choose a VPC: Only VPCs with a DB Subnet Group(s) are allowed

Availability Zone: MultiAZ deployment selected

Option Group:

If you have custom DB Parameter Groups or DB Security Groups you would like to associate with this DB Instance, select them below, otherwise proceed with default settings.

Parameter Group:

DB Security Group(s):

default

ndhtest

[Cancel](#) [Previous](#) [Next Step](#)

Securing the System

AWS includes a thorough system of firewalls that are configured via security groups. Security groups define how, and from where, the servers in the group can be accessed.

Security groups are created in a separate part of the Amazon RDS console. In the following screenshot, specific IP addresses are called out (via CIDR notation), and a security group for web servers is being added:

[DB Security Groups](#) > ndhtest

Security Group Details

Connection Type	Details	Status	Actions
CIDR/IP	CIDR/IP: 209.203.71.2/32	authorized	Remove
CIDR/IP	CIDR/IP: 76.196.233.30/32	authorized	Remove
CIDR/IP	CIDR/IP: 76.88.85.72/32	authorized	Remove
AWS Account Id: 323826331358 (change)			
EC2 Security Group <input type="button" value="v"/>	EC2 Security Group: <input type="text" value="NDH-Web Servers (sg-dda476b6)"/>		Add

[Refresh Security Groups](#)

[Recent Events](#)

[Tags](#)

Security group access is determined by AWS itself, meaning that network and access security can be kept separate from the server layer.

Bonus! Setting up Automatic Management

Amazon RDS automatically backs up your data. You can set up additional backups, retention policies, and maintenance windows as part of the Amazon RDS installation, as shown in this screenshot of the final page in the Amazon RDS setup process:

Management Options

Enabled Automatic Backups: ☒ Yes ☐ No

The number of days for which automated backups are retained.

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

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

Backup Window: ☐ Select Window ☒ No Preference

The weekly time range (in UTC) during which system maintenance can occur.

Maintenance Window: ☐ Select Window ☒ No Preference

[Cancel](#)

[Previous](#)

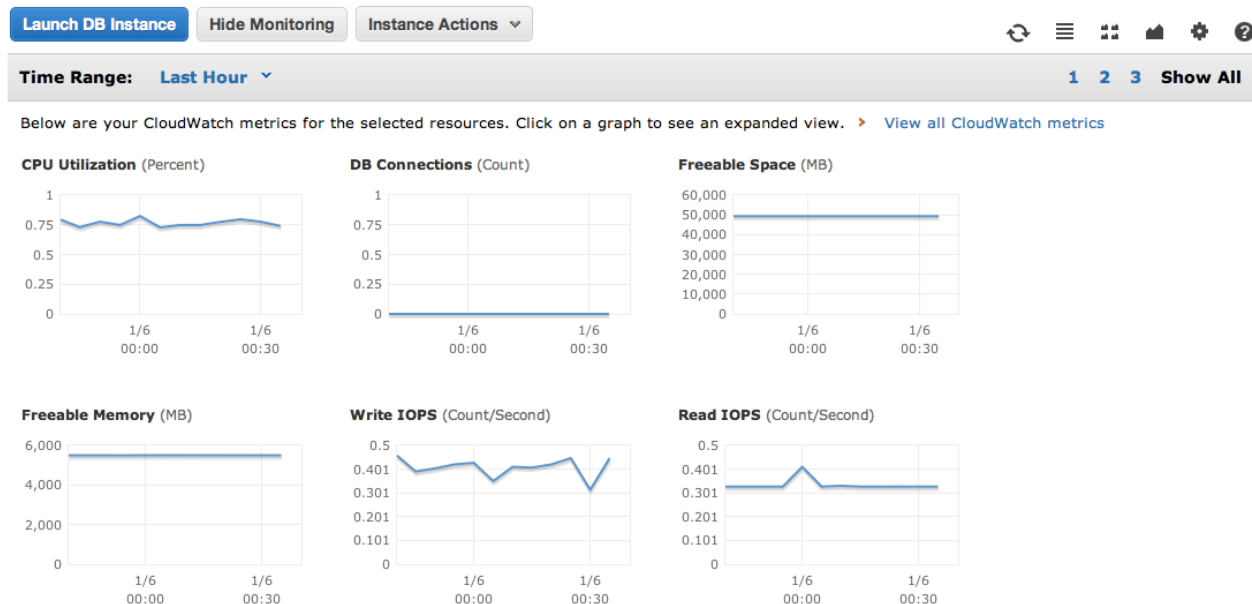
[Next Step](#)

Running Your System Production with Amazon RDS

Once everything is configured and running, Amazon RDS provides a spectrum of tools and capabilities to monitor the running systems, increase capacity, create high availability, and scale up to meet customer needs.

Automatic Amazon RDS Monitoring

AWS provides a free service called CloudWatch with all AWS products and services. CloudWatch has a set of metrics that it monitors automatically for each AWS offering. The following screenshot shows an example of a CloudWatch screen as it monitors an active database:



Any of the metrics that CloudWatch monitors for Amazon RDS can have an alarm set on them. This means that when a metric reaches a certain level, CloudWatch will perform an alerting action, such as sending an email or a text, executing an HTTP call, or performing other programmatic actions. This allows Amazon RDS to tie into whatever network monitoring system you are using, and you can also tie actions such as increasing storage space to an alarm.

Increasing Capacity

If you are running the Oracle, MySQL, or PostgreSQL DBMS in Amazon RDS, you can increase or decrease the amount of storage while the DBMS is running (Microsoft SQL Server cannot do this due to its internal representation of storage).

Creating High Availability

Every AWS region has at least two Availability Zones (AZ). Each AZ is configured to be completely independent and isolated, meaning that a failure in one AZ will have no effect on another AZ in the same region. When you set up Amazon RDS, you can choose to have it run in Multi-AZ mode, meaning that the master database server is replicated in another AZ. If the master fails, the replica takes over with no intervention on your part.

If you chose to install Amazon RDS without Multi-AZ, you can add it to your configuration later, even while the system is running. Adding Multi-AZ has no effect on the existing server.

Scaling Amazon RDS

As use of your system increases, you can scale up the performance of Amazon RDS. If database reads are becoming a problem, you can create Read Replicas of your database. The Read Replicas assist with read access to the database, while write access traffic is routed to the main server. Read Replicas can be created or destroyed at any time, even when the DBMS is running. This means that Read Replicas can be created when CloudWatch detects high levels of activity, and destroyed when activity has moderated.

If the DBMS itself is the bottleneck, then Amazon RDS can upgrade the instance type the server runs on. During the period that the new instance is starting the DBMS will be unavailable. Thus, the command to upgrade the instance has its delay flag set, meaning that the command will run at the next maintenance interval. The delay can be overridden, however, so that the upgrade is invoked immediately.

Using Amazon RDS in Development and Quality Assurance

Because the production Amazon RDS system is just a series of settings in AWS, it is simple to copy the configuration for use by Dev. This means that Dev does not have to wait to get production resources to help with setting up the development environment. And because Amazon RDS can run on a wide variety of instance types, Dev can run its server on the lowest-cost feasible instance, typically a micro that runs about 2.5 cents an hour for MySQL.

The same is true for QA: it can use a snapshot of the production system to create its test environment. In addition, it is easy for QA to scale up to full production size, conduct any performance or stress testing, and then turn off all the big instances. In other words, QA can completely duplicate a running production environment for whatever testing it needs, and pay only for the time spent testing. No hardware is sitting unused on racks, gathering dust and going out of date, waiting for your organization's next software release.

Conclusion

Databases are becoming increasingly important as companies realize that the data they've been collecting contains countless business opportunities. But the freeing up of company personnel to explore and exploit these opportunities means that the burdens of database management need to fall elsewhere. Amazon RDS was built for exactly this situation: unloading the drudgery of traditional DBMS from a company's data experts and allowing them to focus on adding value to their data and their company.

Learn More

Global Knowledge offers several courses that introduce Amazon RDS, teach users how to set up and use Amazon RDS, and how to maintain system operations that include Amazon RDS.

[AWS Essentials](#)
[Architecting on AWS](#)
[Systems Operations on AWS](#)

Visit www.globalknowledge.com or call **1-800-COURSES (1-800-268-7737)** to speak with a Global Knowledge training advisor.

About the Author

Jon M. Gallagher has decades of experience creating and running large-scale software systems for the web and for enterprises. He has been using Amazon Web Services since its introduction in 2006.

Sources/References/Resources

- Overview of Amazon Web Services (AWS):
<http://aws.amazon.com/about-aws/>
- Overview of Amazon Relational Database Service (Amazon RDS):
<http://aws.amazon.com/rds/>
- Amazon RDS Documentation:
<http://aws.amazon.com/documentation/rds/>
- Amazon RDS Cost Calculator:
<http://aws.amazon.com/rds/amazon-rds-tco/rds-tco-tool-request-form/>
- Amazon RDS Free Usage Tier:
<http://aws.amazon.com/rds/free/>