

Setting up an Amazon Web Services (AWS) and RStudio Data Connective Environment

Robert P. Schumaker*

Professor - Computer Science Dept., University of Texas at Tyler

Director - Data Analytics Lab, University of Texas at Tyler

Consultant - ORS Research Design and Data Analysis Lab

December 3, 2021

Contents

1	Process Overview	2
2	Amazon AWS Overview	2
2.1	Virtual Machines	3
2.2	File Storage	3
2.3	Database	5
3	Tutorial - Setting up an AWS and RStudio Data Connective Environment	5
3.1	Setting up an Amazon AWS Account	5
3.2	Creating an Amazon RDS Instance	5
3.3	Configuring Security and Permissions	6
3.3.1	Security Group Rules	6
3.3.2	User Accounts and Permissions	7
3.4	Installing R and RStudio	9
3.5	Example R Code	9

*rob.schumaker@gmail.com

1 Process Overview

This document will demonstrate how to set up a data storage environment in Amazon Web Services (AWS) and how to connect to it using RStudio. Completion of this tutorial will require a creditcard for Amazon AWS account creation. A typical research project using the options described within this document will cost approximately \$20/month, however, be aware that different options/activities could incur significant cost.

An overview of the process architecture used in this document is shown in Figure 1.

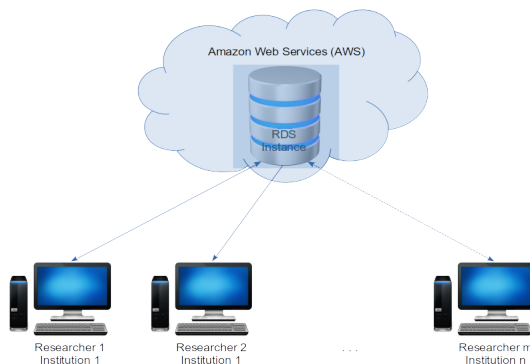


Figure 1: AWS RDS Overview

From this figure, a Relational Database Service (RDS) instance can share data amongst an almost unlimited number researchers from an almost unlimited number of institutions. RDS-level account permissions can also be set to allow/disallow researcher access to specific data as well as control the direction of data transfer (read/write vs read-only).

The rest of this document will focus on setting up an AWS account, creating an RDS instance, configuring security and permissions, testing against `mysql` Workbench, installing RStudio and working with some example R code. Data collection is beyond the scope of this tutorial, however, later sessions will explore this added functionality.

2 Amazon AWS Overview

Amazon Web Services (AWS) is the computational engine behind the eCommerce giant Amazon. Prior to being made available to the public, Amazon

datacenters were sized solely to handle the online traffic for one day of the year only, Black Friday. The rest of the year Amazon datacenters were mostly idle. In the early 2000s several Amazon engineers pitched the idea of selling excess capacity to business and private individuals. In 2002 AWS was opened to the public. Demand for AWS services skyrocketed and helped propel their project leader Andy Jassy into the role of Amazon's CEO in 2021.

AWS is an ever-growing collection of web-related services. From virtual machines that allow for scalable remote computing, remote file storage capabilities, database storage, machine learning, mobile app development, blockchain, satellite control, networking, etc.

For the purposes of this tutorial we will focus lightly on the virtual machine environment and file storage capability. We will spend more time on database and security/permission configuration.

2.1 Virtual Machines

Virtual machines (VM) are remote computing instances. Each VM operates similarly to your laptop/desktop computer with an operating system and applications. Students like VMs because they can share a VM instance with group members to work on projects remotely and when the semester is over, terminate the instance only paying for what was used.

Within the AWS ecosystem, VMs are the backbone of many web-related services, databases included. While we don't need to pay specific attention to VMs for the purpose of this tutorial, for those with specific data or hardware needs, be aware that you will need to pay more attention to correctly sizing your VM to your RDS needs. Examples include more robust processing, memory or networking requirements. For the majority of research projects, using the defaults within this document will be sufficient. If a research project's VM needs change, a new RDS instance with the new hardware requirements can be created and data migrated to the new instance. While beyond the scope of this tutorial, be aware that it is possible.

2.2 File Storage

Another important aspect in AWS is file storage. Files, photos, video, and data all require storage. Students like remote file storage because they can share files between group members or back up their computers for relatively

cheap. As of December 2021, Simple Storage Services (S3) costs \$0.023 per GB up to 50TB. Researchers with 1TB of data with frequent access could expect to pay \$23.52 monthly for data storage. This is in addition to your RDS instance costs which are separate. If data access is less frequent, costs can decrease substantially. For long-term data storage, primarily for data backups, S3 Glacier costs \$0.004 per GB and S3 Glacier Deep Archive is \$0.00099 per GB.

Storage is an integral part of an RDS instance. While we do not need to configure it separately, we will need to have an idea of our data environment needs to estimate costs and budget appropriately.

Figure 2 shows an example monthly AWS bill for the OceanPlatform RDS instance we will be testing against.

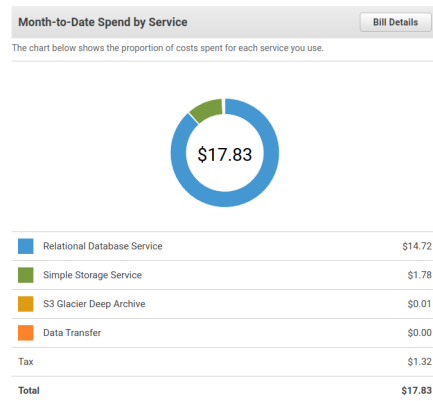


Figure 2: An Example AWS Monthly bill

Note the bill is broken into several helpful parts. The RDS (Relational Database Service) instance consumes the majority of the monthly budget. Simple Storage Service (S3) is the instance that handles all of the data storage needs of the RDS instance. S3 Glacier Deep Archive is for data from past research projects that are currently inactive.

Data Transfer is another charge that can be incurred if data transfers exceed the monthly allotment of 15 GB per month for the free-usage tier. This can occur if the RDS and S3 are in different geographic regions or if data is transferred out of AWS. Data moving into AWS is typically free. Data Transfer costs are typically \$0.16 per GB between geographic regions and \$0.114 per GB out of AWS (up to 10TB/month).

2.3 Database

In order to create and store data, you will need a database. There are many different types of RDS platforms to choose from including NoSQL instances like MongoDB, graph-based instances like Neptune, big-data instances like Apache Hadoop and relational database instances like MariaDB, MySQL, Microsoft SQL Server, Oracle, Postgre, etc.

This tutorial will focus on setting up an Amazon RDS instance using MySQL. However, with some intuition other database types could be used as well.

3 Tutorial - Setting up an AWS and RStudio Data Connective Environment

These instructions were created December 2021 and are specific to the current AWS layout. AWS is known to change their layouts frequently so you may need to be resourceful in order to complete this tutorial.

3.1 Setting up an Amazon AWS Account

1. Go to <https://aws.amazon.com>
2. Click on **Sign in to the Console**
3. Click on **Create a new AWS account**
4. Enter your personal details. AWS will ask for creditcard information
5. Account approval could take anywhere from instantly to several days

3.2 Creating an Amazon RDS Instance

1. Go to <https://aws.amazon.edu> and login to the *AWS Management Console*
2. In the **Search bar** type *RDS*. You should see RDS under *Services*. Click it
3. Under the box *Create Database* is a button **Create Database**. Click it
4. Choose **Easy Create, MySQL, Free tier**

5. Under *DB Instance Identifier*, choose a name for your RDS instance
6. Complete *Master username*, *Master password* and *Confirm password* boxes
7. Click **Create database**
8. AWS will now spin up an RDS instance. When ready the *Status* column will say "Available"

3.3 Configuring Security and Permissions

Although your new RDS instance may say Available, it isn't ready to connect remotely yet. You can connect from within the AWS ecosystem, for example with an EC2, Lightsail or other AWS product, but not from your personal desktop/laptop. These instructions will allow for remote access anywhere in the world. If IP-based restrictions are necessary (e.g., only certain computers or institutions should have access), some knowledge of CIDR networking can be beneficial.

Security and Permissions are two separate concepts. Security deals with firewall access and is at the perimeter of the AWS ecosystem. *Can another computer be allowed inside AWS to interact with the database?* Permissions are at the database-level. These include account creation and specifying what access a user has inside the RDS. We will tackle both of these separately. Once setup, you almost never have to deal with these configurations again.

3.3.1 Security Group Rules

1. Within the *Amazon RDS service*, click on your **DB identifier**
2. Under the box *Security Group Rules* is a security group type that contains *Inbound*. Click that **Security Group**
3. Click on the tab **Inbound Rules**
4. Click on the box **Edit inbound rules**
5. Click on the box **Delete**
6. Click on the box **Add rule**

7. Under *Type*, select **MYSQL/Aurora**. This will choose the TCP protocol and port number 3306 (grayed out)
8. Under *Source*, select **Anywhere-IPv4**. This will choose the CIDR address of 0.0.0.0/0
9. Click **Save rules**
10. Under *Details*, make note of the *Security group name* (e.g., default)
11. Return to the *Amazon RDS service*, click on your **DB identifier**
12. Click the box **Modify**
13. Under *Connectivity, Security group*, you should see the security group you made note of earlier. If not, select it
14. Click on **Additional configuration**
15. Click on **Publicly accessible**
16. Click **Continue**
17. Click **Apply immediately**
18. Click **Modify DB instance**

A note for the advanced. If you wanted to restrict traffic to only allow connections from UT Tyler, you would choose *Source*, **Custom** and enter the CIDR of 129.114.0.0/16

You can enter as many security group rules as needed for peer institutions. If your co-researchers were able to connect but suddenly can no longer, check the their IP against the security groups.

If you use 0.0.0.0/0 you do not need to worry about setting additional security group rules, however, your instance is more susceptible to attack. You will need to decide the balance between security and accessibility for your specific data needs.

3.3.2 User Accounts and Permissions

1. Within the *Amazon RDS service*, click on your **DB identifier**
2. Under the *Connectivity & security* tab is a string of text under *Endpoint*

3. Copy that text string to your clipboard (e.g., oceanplatform0.cdb7tnix15tn.us-west-2.rds.amazonaws.com)
4. If not already installed, install mySQL Workbench (<https://dev.mysql.com/downloads/workbench>)
5. Open mySQL Workbench
6. Click the plus sign in a circle to create a connection to the RDS instance
7. For **Connection Name** pick a name for your connection. The name doesn't really matter, it's to help you find it
8. For **Hostname** paste the endpoint string
9. For **Username**, enter the username you selected earlier for RDS creation
10. Click **Test Connection** and enter the mySQL admin/root password
11. If you successfully made the mySQL connection, click Ok to exit the **Setup New Connection** dialog box
12. Click your new connection under **mySQL Connections**
13. Click the tab **Administration**
14. Click **Users and Privileges**
15. To add a new user account click **Add Account**
16. **WARNING: You may be tempted to use the *Limit to Hosts Matching* box to restrict IP access. If you do this on your admin account AND your IP address changes, you will lose admin access to your database. Consider yourself warned.**
17. Under the tabs *Administrative Roles* and *Schema Privileges* you can set what access the user has and to what data respectively
18. Once finished, click the button **Apply**

You are also free to design your database structure and ETL data from within mySQL Workbench. Database design and ETL work is beyond the scope of this tutorial. **Note:** If you are planning on accessing your RDS instance from the UT Tyler campus, there is an additional step! You will need to let Information Technology know about the RDS instance and how long you plan on using it (if applicable) and request access on-campus.

1. Create an email to Chris Green (cgreen@uttyler.edu), Director of Information Security, cc Robbie Woodley (rwoodley@uttyler.edu), Networking Manager and cc Matt Izard (mizard@uttyler.edu), Senior Security Analyst.
2. Request permission for on-campus access to your RDS instance
3. Specify the endpoint you copied your clipboard earlier. Include the words "mySQL, tcp port 3306"
4. Specify the purpose of your database and how long you intend to use it (if known)

3.4 Installing R and RStudio

1. If not already installed, install R (<https://cran.r-project.org/>)
2. If not already installed, install RStudio Desktop (<https://www.rstudio.com/products/rstudio/download>)
3. Open RStudio
4. Install package RMariaDB, type **install.packages("RMariaDB")**
5. **install.packages("dplyr")**

3.5 Example R Code

The following code is a quick example to use on my OceanPlatform instance. This RDS is used by my data analytics students. Please run this code in RStudio to test the data connective environment.

- `rm(list=ls())`
- `library(RMariaDB)`
- `library(dplyr)`
- `mydb <- dbConnect(MariaDB(), user='student', password='cosc4347', dbname='COSC4347', host='oceanplatform0.cdb7tnix15tn.us-west-2.rds.amazonaws.com')`
- `rs = dbSendQuery(mydb, "SELECT Univ, CourseName, Course, CourseYear, Semester, AcadYear,N, Rating, A, B, C, D, F, W FROM Evaluations")`
- `data = dbFetch(rs, n=-1)`

- `UTTFall <- filter(data, Univ=='UTT' & Semester=='Fall')`
- `mean(UTTFall$Rating, na.rm=TRUE)`
- `UTTFall[is.na(UTTFall)] <- 0`
- `predict.eval <- lm(Rating ~ A+B+C+D+F, data=UTTFall)`
- `summary(predict.eval)`

Once you have ETL'ed to your own RDS instance, change the host endpoint, dbname, user and password to reflect your specific parameters.
Enjoy!