COSC349 Assignment 2

Ben Scobie
5683357
https://github.com/bscobie/COSC349A2.git

Introduction:

I chose to extend the program I developed for assignment one. The program is a booking system for reserving video games. This application allows users to book a video game from the supplied list as well as giving their details. There is a second web page for an admin user that is able to see the current bookings stored in the database.

Application Deployment:

I began deploying my application to AWS using Vagrant following the example from lab 9. However, I couldn't get very far as Vagrant wouldn't accept my AWS access credentials. With the deadline coming close I decided to manually deploy my application to AWS using Ec2 instances, Relational Database Service(RDS) for the database, and make use of Amazons Simple Notification Service(SNS) service.

I began by launching the first Ec2 instance using the Ec2 dashboard on AWS. I selected Amazons Linux 2 AMI, t2.micro as the instance type, and configured the security group. The rules needed to allow for SSH, HTTP, and HTTPS connections so the instance could be accessible. Once the instance was running and available, I was able to SSH to it from my own machine and begin installing an Apache web server. The following command,

```
sudo yum install -y httpd
```

installed the web server and then,

```
sudo systemctl start httpd
```

started the server and visiting the instances address displayed the initial Apache server page. This is when I was able to upload my files for the client page to the instance so they would be displayed.

Next, I created a MySQL DB instance using the Amazon RDS console. To access the database, I had to install MySQL on to the Ec2 instances through the terminal using

```
sudo amazon-linux-extras install php8.0 mariadb10.5
```

Then, I used the command,

```
mysql -h booking-db.crcmg19fp53h.us-east-1.rds.amazonaws.com -P 3306 -u admin -p
```

to connect to the database and execute the create table and insert statements from assignment 1. Once I debugged any errors that arose during this process, I was left with a website that would send booking information and be stored in the RDS. At this point, all that was left was to launch an identical Ec2 instance as the first one and follow the same steps to deploy the admin web server from assignment 1.

Once the application was running and deployed to AWS, I set up the SNS to interact with the RDS. This just listens to the RDS and when a certain event happens, i.e. the database is rebooted, then any subscribed users will receive a notification of the event.

Accessing the Application:

To access the application in the cloud, use the following in a browser to access the client facing part:

http://ec2-107-23-216-245.compute-1.amazonaws.com

To access the admin facing part of the application, use the following in a browser:

http://ec2-34-230-99-236.compute-1.amazonaws.com

To access the client server instance:

ssh -i ~/.aws/cosc349-2022.pem ec2-user@ec2-107-23-216-245.compute-1.amazonaws.com

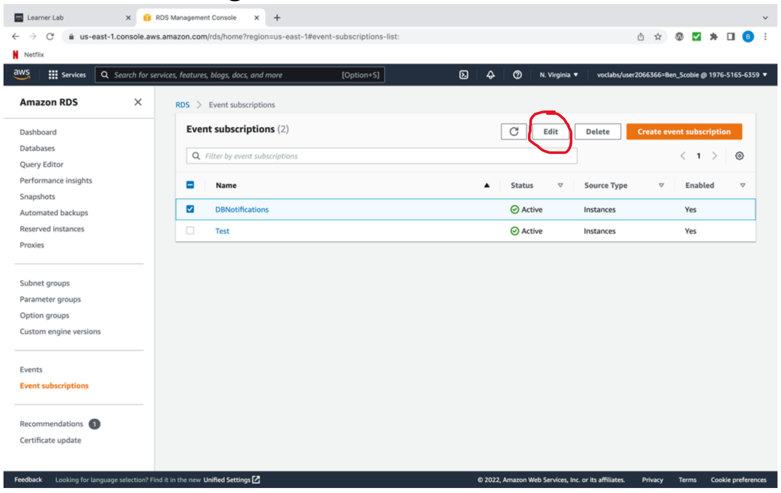
To access the admin server instance:

ssh -i ~/.aws/cosc349-2022.pem ec2-user@ec2-34-230-99-236.compute-1.amazonaws.com

Service Interaction:

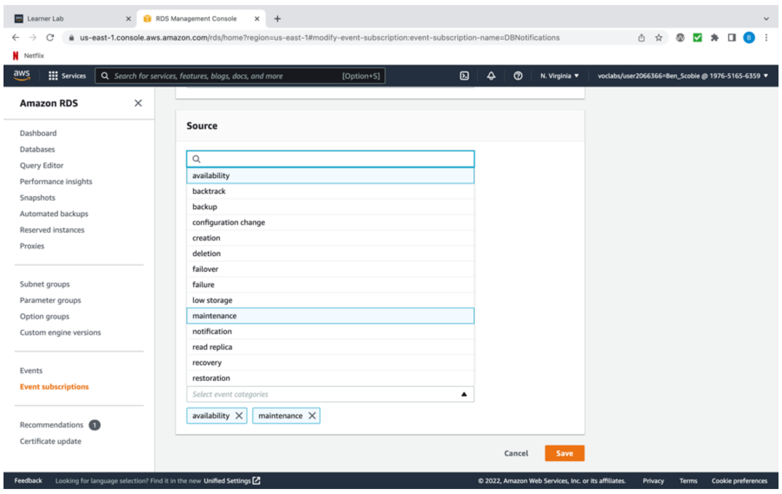
My application is made up of two virtual machines as Amazon Ec2 instances that are running an Apache web server each. Each of the Ec2 instances interact with the MySQL RDS instance in different ways. The web-server instance retrieves the stored title values from the Games table to populate the dropdown box that is in the booking form. When the user submits the booking form, the input data is then posted to the database and stored in the bookings table. The second Ec2 instance running the admin server only pulls data from the bookings table and displays it in a table.

The other non-Ec2 service I chose to use was the Simple Notification System. I set this up to interact with the RDS and notify me every time the database is shutdown, rebooted, or going under maintenance. There are many more events that can be selected such as backups and storage levels. The following screenshots show how the event subscription can be configured.



Event subscriptions (2)

Name	Status	Source Type	Enabled
☑ DBNotifications	Active	Instances	Yes
☐ Test	Active	Instances	Yes


Source

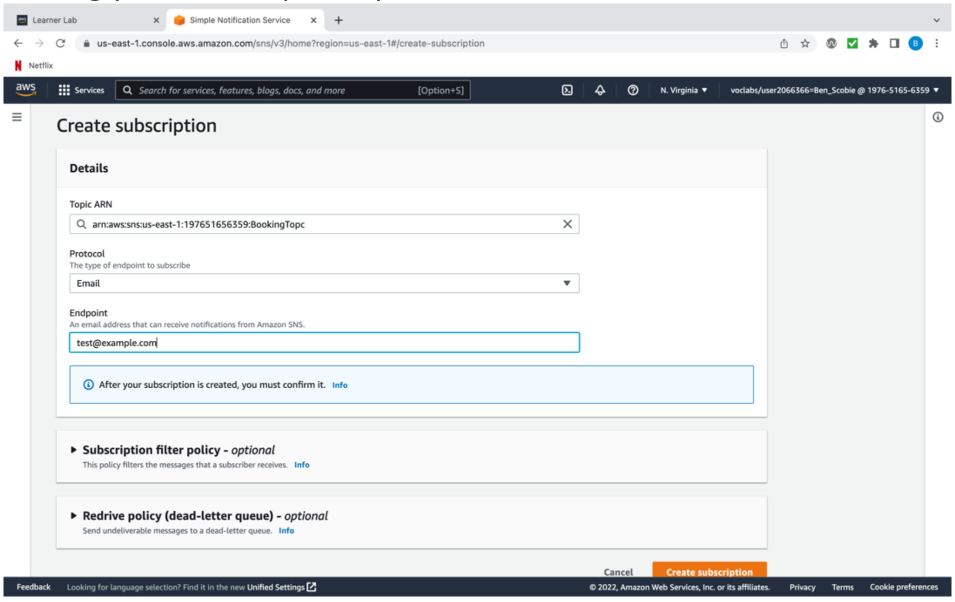
availability
backtrack
backup
configuration change
creation
deletion
failover
failure
low storage
maintenance
notification
read replica
recovery
restoration

Select event categories

availability ✕ maintenance ✕

Cancel Save

To setup your own email to receive these notifications, head to the SNS dashboard and select the subscriptions tab. Then, click the create new subscription button. Select the ARN and protocol as in the following screenshot. Endpoint is where you enter the email address you want to receive notifications on. Click create and then after confirming your subscription, you will now receive notifications about the RDS.


Create subscription

Details

Topic ARN
arn:aws:sns:us-east-1:197651656359:BookingTopc

Protocol
The type of endpoint to subscribe
Email

Endpoint
An email address that can receive notifications from Amazon SNS.
test@example.com

After your subscription is created, you must confirm it. Info

Subscription filter policy - *optional*
This policy filters the messages that a subscriber receives. Info

Redrive policy (dead-letter queue) - *optional*
Send undeliverable messages to a dead-letter queue. Info

Cancel Create subscription

Running Costs:

Using the AWS pricing calculator, I was able to estimate that the Ec2 instances together would incur between $13 to $20 monthly, depending on peak usages. The RDS is estimated to cost just under $24 a month assuming it will be up and running 24 hours a day. The SNS service won’t incur any charges until it reaches around 10,000 notifications per month which will be very unlikely in this application. So, in total, the estimated monthly costs for this application to be hosted on AWS would be around $45.