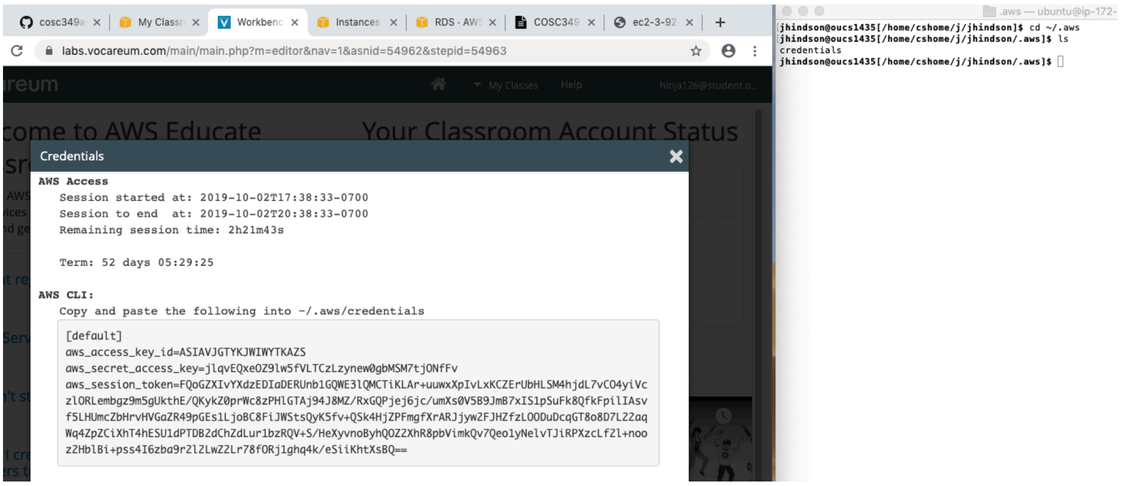
COSC349 Assignment 2
Jasmine Hindson
8148181

For the second assignment, I continued on from my first assignment and used the two original web server VMs that I had already created. Instead of using the third VM (database), I removed it from my Vagrantfile and instead started using an AWS database, RDS. I chose RDS because it used MySQL which was familiar to me and it provided the functionalities that I needed for my application. My commits can be seen on Github for assignment 2. It is still in the repository called 'cosc349' since I just continued working from assignment 1.

Deploying the application

To run my application, I first needed to initialise the credentials file in the file path '~/.aws'. To do this I typed 'cd ~/.aws' into terminal. In here, I created the file 'credentials' by typing 'nano credentials' whilst in the directory '~/.aws'. In the credentials file, I had to paste the credentials information for my personal AWS account from the 'Account Details' button that was on the AWS Workbench webpage. I copied the credentials information and pasted it into the file, 'credentials'. In the screenshot below, I am showing an example of the credentials (during that session time) from AWS that needed to be copied into the credentials file that I have set up in terminal next to it.

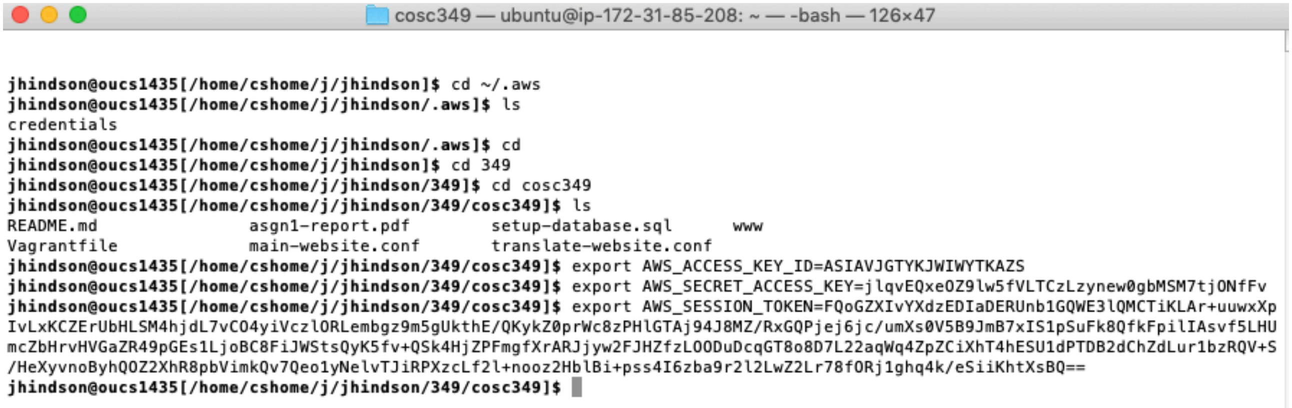


```
AWS Access
   Session started at: 2019-10-02T17:38:33-0700
   Session to end  at: 2019-10-02T20:38:33-0700
   Remaining session time: 2h21m43s

   Term: 52 days 05:29:25

AWS CLI:
   Copy and paste the following into ~/.aws/credentials
    [default]
    aws_access_key_id=ASIAVJGTYKJWIWYTKAZS
    aws_secret_access_key=jlqvEQxeOZ9lw5fVLTCzLzynew0gbMSM7tjONfFv
    aws_session_token=FQoGZXIvYXdzEDIaDERUnb1GQWE3lQMCTiKLAr+uuwxXpIvLxKCZErUbHLSM4hjdL7vCO4yiVc
    zlORLembgz9m5gUkthE/QKykZ0prWc8zPHlGTAj94J8MZ/RxGQPjej6jc/umXs0V5B9JmB7xIS1pSuFk8QfkFpilIAsv
    f5LHUmcZbHrvHVGaZR49pGEs1LjoBC8FiJWStsQyK5fv+QSk4HjZPFmgfXrARJjyw2FJHZfzLOODuDcqGT8o8D7L22aq
    Wq4ZpZCiXhT4hESU1dPTDB2dChZdLur1bzRQV+S/HeXyvnoByhQOZ2XhR8pbVimkQv7Qeo1yNelvTJiRPXzcLf2l+noo
    zZHblBi+pss4I6zba9r2l2LwZ2Lr78fORj1ghq4k/eSiiKhtXsBQ==
```

```
jhindson@oucs1435[/home/cshome/j/jhindson]$ cd ~/.aws
jhindson@oucs1435[/home/cshome/j/jhindson/.aws]$ ls
credentials
jhindson@oucs1435[/home/cshome/j/jhindson/.aws]$ 
```

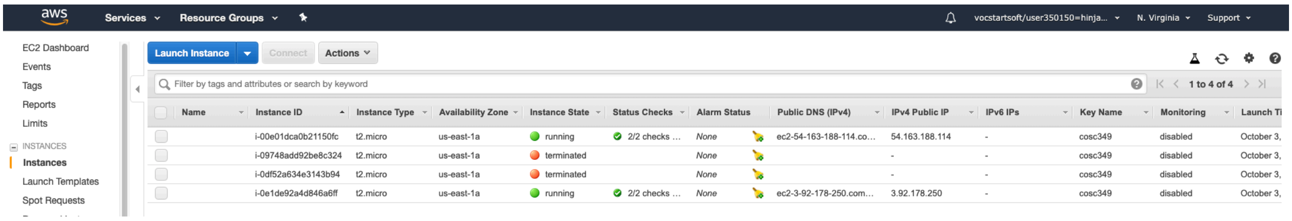
Once this was done, I navigated back to the folder that had the Vagrantfile and typed these 3 commands with the corresponding values from the credentials file (using the values from the previous screenshot):

- export AWS_ACCESS_KEY_ID=
- export AWS_SECRET_ACCESS_KEY=
- export AWS_SESSION_TOKEN=



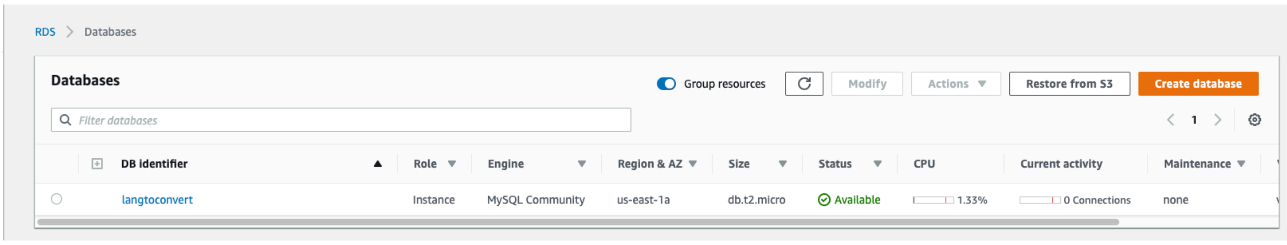
```
jhindson@oucs1435[/home/cshome/j/jhindson]$ cd ~/.aws
jhindson@oucs1435[/home/cshome/j/jhindson/.aws]$ ls
credentials
jhindson@oucs1435[/home/cshome/j/jhindson/.aws]$ cd
jhindson@oucs1435[/home/cshome/j/jhindson]$ cd 349
jhindson@oucs1435[/home/cshome/j/jhindson/349]$ cd cosc349
jhindson@oucs1435[/home/cshome/j/jhindson/349/cosc349]$ ls
README.md               asgn1-report.pdf        setup-database.sql      www
Vagrantfile             main-website.conf       translate-website.conf
jhindson@oucs1435[/home/cshome/j/jhindson/349/cosc349]$ export AWS_ACCESS_KEY_ID=ASIAVJGTYKJWIWYTKAZS
jhindson@oucs1435[/home/cshome/j/jhindson/349/cosc349]$ export AWS_SECRET_ACCESS_KEY=jlqvEQxeOZ9lw5fVLTCzLzynew0gbMSM7tjONfFv
jhindson@oucs1435[/home/cshome/j/jhindson/349/cosc349]$ export AWS_SESSION_TOKEN=FQoGZXIvYXdzEDIaDERUnb1GQWE3lQMCTiKLAr+uuwxXp
IvLxKCZErUbHLSM4hjdL7vCO4yiVczlORLembgz9m5gUkthE/QKykZ0prWc8zPHlGTAj94J8MZ/RxGQPjej6jc/umXs0V5B9JmB7xIS1pSuFk8QfkFpilIAsvf5LHU
mcZbHrvHVGaZR49pGEs1LjoBC8FiJWStsQyK5fv+QSk4HjZPFmgfXrARJjyw2FJHZfzLOODuDcqGT8o8D7L22aqWq4ZpZCiXhT4hESU1dPTDB2dChZdLur1bzRQV+S
/HeXyvnoByhQOZ2XhR8pbVimkQv7Qeo1yNelvTJiRPXzcLf2l+nooz2HblBi+pss4I6zba9r2l2LwZ2Lr78fORj1ghq4k/eSiiKhtXsBQ==
jhindson@oucs1435[/home/cshome/j/jhindson/349/cosc349]$ 
```

The next step was to run ‘vagrant up —provider=aws’ in the same terminal as seen in the screenshot above. This took a while as there were three virtual machines to start up. Once this was done I navigated to EC2 in AWS to see if the instances had been created. In the screenshot below you can see that my two web server instances have started up and they are running.



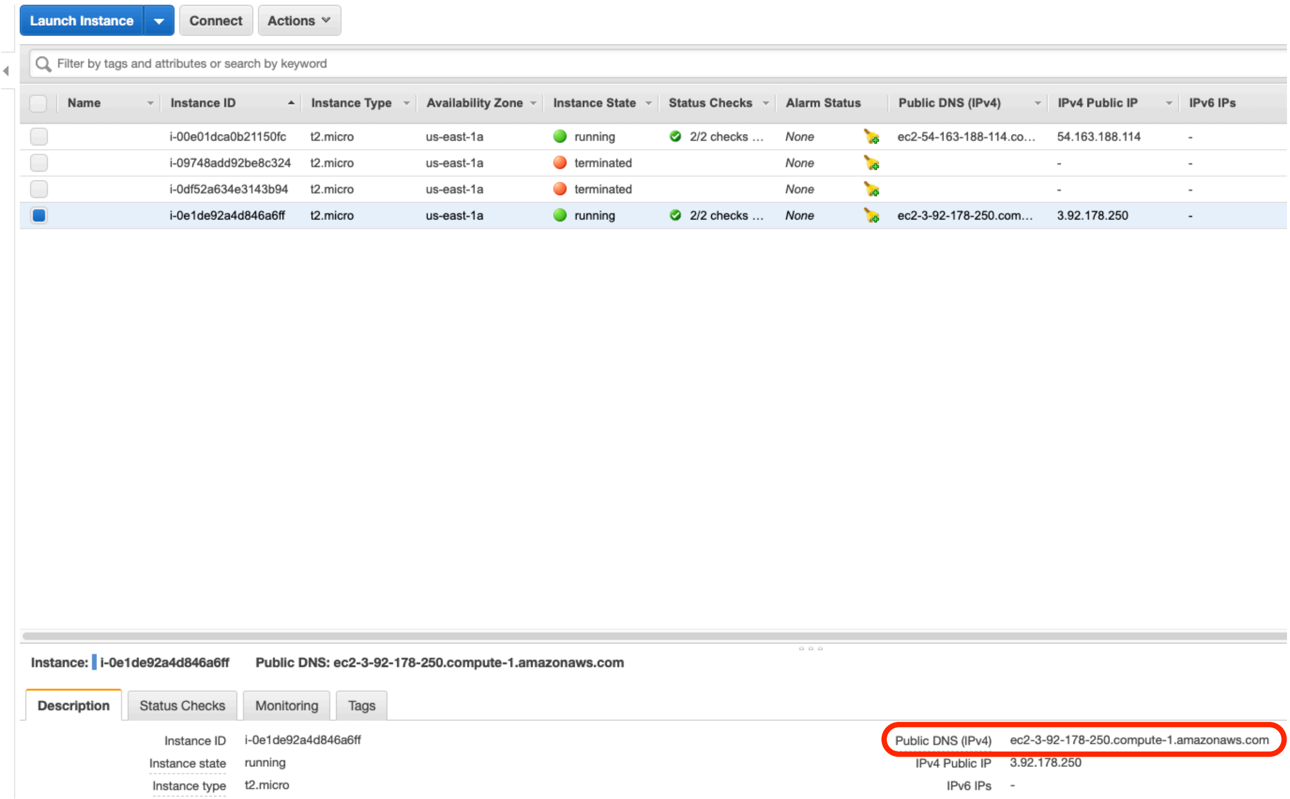
Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)	IPv4 Public IP	IPv6 IPs	Key Name	Monitoring	Launch Ti
	i-00e01dca0b21150fc	t2.micro	us-east-1a	● running	✓ 2/2 checks ...	None	ec2-54-163-188-114.co...	54.163.188.114	-	cosc349	disabled	October 3,
	i-09748add92be8c324	t2.micro	us-east-1a	● terminated		None		-	-	cosc349	disabled	October 3,
	i-0df52a634e3143b94	t2.micro	us-east-1a	● terminated		None		-	-	cosc349	disabled	October 3,
	i-0e1de92a4d846a6ff	t2.micro	us-east-1a	● running	✓ 2/2 checks ...	None	ec2-3-92-178-250.com...	3.92.178.250	-	cosc349	disabled	October 3,

Along with these two instances running, I had to make sure the RDS database instance was running as well because my web servers were now using RDS as a database instead of the old database server VM. In the screenshot below, you can see that there is a database instance called ‘langtoconvert’ and that it is available. I will go into more detail about RDS and how I used it for my application further down in the report.



DB identifier	Role	Engine	Region & AZ	Size	Status	CPU	Current activity	Maintenance
langtoconvert	Instance	MySQL Community	us-east-1a	db.t2.micro	✓ Available	1.33%	0 Connections	none

To see the web server that provided user interaction, I copied the Public DNS (highlighted with a red circle in the screenshot below) and pasted it into a new browser and saw that my web server was now being hosted by the cloud.



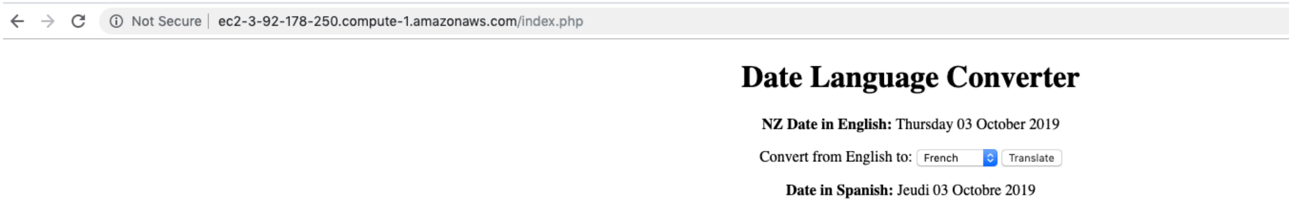
Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)	IPv4 Public IP	IPv6 IPs
	i-00e01dca0b21150fc	t2.micro	us-east-1a	● running	✓ 2/2 checks ...	None	ec2-54-163-188-114.co...	54.163.188.114	-
	i-09748add92be8c324	t2.micro	us-east-1a	● terminated		None		-	-
	i-0df52a634e3143b94	t2.micro	us-east-1a	● terminated		None		-	-
☑	i-0e1de92a4d846a6ff	t2.micro	us-east-1a	● running	✓ 2/2 checks ...	None	ec2-3-92-178-250.com...	3.92.178.250	-

Instance: i-0e1de92a4d846a6ff   Public DNS: ec2-3-92-178-250.compute-1.amazonaws.com

Instance ID i-0e1de92a4d846a6ff
Instance state running
Instance type t2.micro

Public DNS (IPv4) ec2-3-92-178-250.compute-1.amazonaws.com
IPv4 Public IP 3.92.178.250
IPv6 IPs -

Below is the screenshot of my website with the DNS in the browser's address bar.



Not Secure | ec2-3-92-178-250.compute-1.amazonaws.com/index.php

Date Language Converter

NZ Date in English: Thursday 03 October 2019

Convert from English to: French Translate

Date in Spanish: Jeudi 03 Octobre 2019

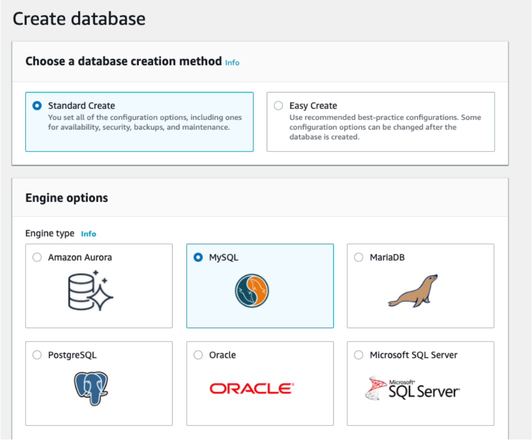
From here the user can use my original idea of a language converter from assignment 1 but it is now hosted by the AWS cloud.

Interacting with my application

Users can view my application by simply typing: http://ec2-3-92-178-250.compute-1.amazonaws.com into the address bar.
For the second assignment, I didn't change the functionality of the web server itself from assignment one. Therefore, the application still works in the way that when the user goes to the website they will see the current NZ date being displayed on the screen. Below the current date, there is a drop down box that allows the user to select a language that they would like to translate the current NZ date into. I have written this all in code in the file '/www/translate/index.php' because I still couldn't find a free open-source library that could do the translations automatically for me. I would have liked to have built on my application for this assignment by adding more languages but due to the restriction of having to code all the translations, I felt that this would not be have been a useful addition to the application.

Using RDS

Along with hosting the web servers on the cloud, I have also removed the dbserver (database server) that I created in assignment 1 and decided to use AWS's database, RDS. I used RDS because it used MySQL which we are already learnt about in assignment 1 from creating a database server. To set up RDS for my application, I first had to set up a RDS instance as shown below:



Create database

Choose a database creation method Info

● Standard Create
You set all of the configuration options, including ones for availability, security, backups, and maintenance.

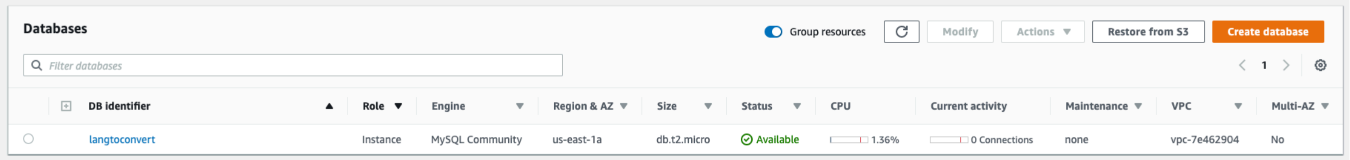
○ Easy Create
Use recommended best-practice configurations. Some configuration options can be changed after the database is created.

Engine options

Engine type Info

○ Amazon Aurora
● MySQL
○ MariaDB
○ PostgreSQL
○ Oracle
○ Microsoft SQL Server

Initialising a RDS instance involved giving the DB instance a unique identifier (name) and creating a username and password that allowed the instance to be accessed through MySQL. I also had to create a new VPC security group for the instance and I made the instance publicly accessible. Once this was all done, I selected 'Create database' which showed me the following:



Databases

DB identifier	Role	Engine	Region & AZ	Size	Status	CPU	Current activity	Maintenance	VPC	Multi-AZ
langtoconvert	Instance	MySQL Community	us-east-1a	db.t2.micro	✓ Available	1.36%	0 Connections	none	vpc-7e462904	No

As you can see from the screenshot above, there is now a DB instance running. The next step was to access MySQL using my assignment 1 dbserver. So I did this by running my assignment 1 by typing ‘vagrant up’ in the terminal. Once this was running, I accessed the dbserver by typing the command ‘vagrant ssh dbserver’. Once in here I used the command:

```
mysql -u admin -h langtoconvert.czpquyt8dyxh.us-east-1.rds.amazonaws.com -P 3306 -p
```

This command prompted me for a password (which I initialised when setting up the DB instance). From here I created the database ‘langToConvert’ by using the command ‘CREATE DATABASE langToConvert’. Then I used the command ‘USE langToConvert’ to get into this database. Once in here I used my original setup-database.sql file that I had to make a table and add data to it. After all this I was able to get my web server interacting with the DB instance which was now being used as the database server for the web server.

Development and Debugging

For the development of this assignment I followed the lab 9 tutorial to get set up with AWS. I first had to go through lab 9, and learn how different aspects of the lab could be applied to my assignment. Using the lab 9 material, I was able to set up two instances on AWS for my web servers. This meant that I still had to use the command ‘vagrant up’ to get my virtual machines running but they are now being hosted on the cloud. After this I set up the DB instance as I have explained in the section above. To get the webserver VM to communicate with the database server and the translateserver, I had to change some variables in the www/index.php file. The main variables I had to change where the ones that created the connection. So instead of using ‘http://192.168.2.32', I had to replace this value with the DNS of the instance created on AWS. This meant that the PHP file was now using the instance being hosted. For the database server, I changed the variable *$db_host = ‘192.168.2.31’;* to *$RDS_host = 'langtoconvert.czpquyt8dyxh.us-east-1.rds.amazonaws.com'*. After the variables were changed in the PHP file the connection was set up meaning that my webserver was working.

In my repository on GitHub, you can see that my first commit for assignment 2 was restructuring my Vagrantfile from assignment 1, so that there were two web server VMs along with a AWS provider. The first bug I came across was that I was getting a 403 Forbidden Access error when running my instances in the address bar. In one of the commits, you can see that I fixed this error by adding in some lines of code to the Vagrantfile that changed permissions on the files and directories using chmod. My other commits include changes that I made to variables in the PHP file and changing the instances endpoints whenever I called ‘vagrant up’.

Conclusion

In conclusion, I found that using AWS was very beneficial for an application because it made it much more accessible to a user. Rather than a user having to run vagrant up themselves, they are able to access my application by typing the instance into the address bar.