Deploying a Webservice to the Cloud

Alec Fraser – 4696790 Luke Tucker – 9809482

Note: Most of our edits were done manually through the AWS dashboard. Within our git repository we have tried to reflect the stages of our development as accurately as possible.

Our Non-EC2 Instances

The non-EC2 instances used within this cloud application are an Amazon RDS running MySQL which stores user and coin information, and an S3 bucket which takes incremental snapshots of the database to provide a backup of the database, should malicious SQL injection occur.

Justification of Our Non-EC2 Instances

To store user data, we have used an RDS instance, which allows for easy storage and access of user data between the EC2 instance and our database. However, if an error occurs within the database or instantized input triggers a harmful SQL request (such as a drop table), the S3 bucket allows for a roll-back to a stable version of the database.

Without the S3 bucket providing a stable and relatively up-to-date backup of the RDS any harmful SQL injections would lead to a complete restart of the database, deleting all user data which would be catastrophic, especially if the service had been running for a prolonged period of months or years.

Deployment of Our Service

Initially we attempted to follow the steps below using an Amazon Linux instance of EC2, however due to issues using httpd instead of apache2 we struggled to get working queries between our PHP files and the MySQL database.

The files for our project in assignment one were stored in a .zip file which was uploaded to an S3 bucket. An EC2 instance running Ubuntu was initialized, with security groups allowing for an SSH connection on port 22 and HTML connections on port 80 for both IPv4 and IPv6.

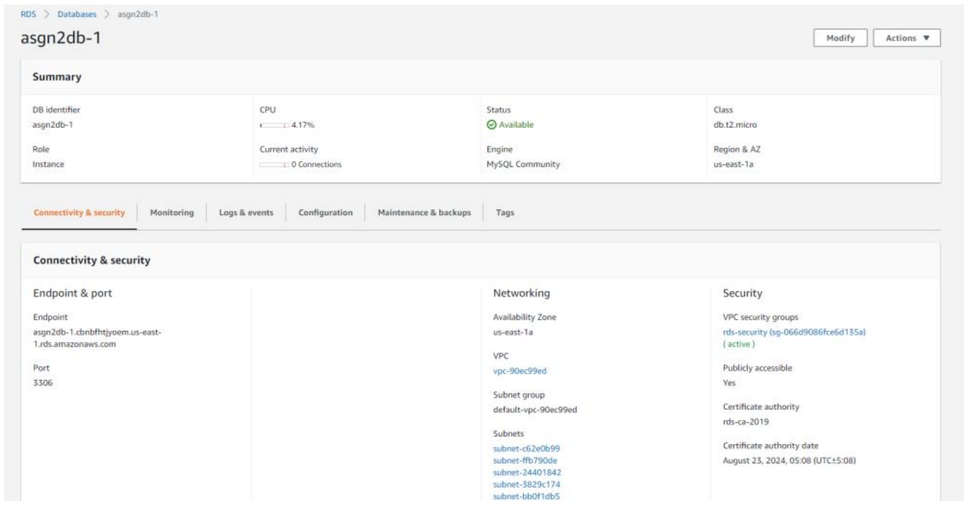
The extra libraries installed were MySQL-server, PHP, apache2, php-mysql, and libapache2-mod-php. After this, the apache2 service was started. The files from the S3 bucket were retrieved and unzipped within the file path /var/www/html/.

An Amazon RDS running MySQL was initialized with a security group allowing for all traffic from MySQL/Aurora queries. The username for the database is “admin”, with password “password”.

The EC2 server attempted the connection to MySQL via MySQL –h (link) -u admin –p. When a connection was established and confirmed, the connection PHP file within the EC2 was edited to make the change from a locally hosted website to a cloud-based MySQL server.

The apache2 service was restarted to ensure all information was up-to-date, and the website was publicly accessible via the address http://3.81.63.54/site.php. This test was a success.

After our setup, our DB instance looked as below on the AWS console:


RDS > Databases > asgn2db-1

asgn2db-1

Summary

DB identifier asgn2db-1	CPU 4.17%	Status Available	Class db.t2.micro
Role Instance	Current activity 0 Connections	Engine MySQL Community	Region & AZ us-east-1a

Connectivity & security

Endpoint & port

Endpoint
asgn2db-1.cbnbfhtjyoem.us-east-1.rds.amazonaws.com

Port
3306

Networking

Availability Zone
us-east-1a

VPC
vpc-90ec99ed

Subnet group
default-vpc-90ec99ed

Subnets
subnet-c62e0b99
subnet-ffb790de
subnet-24401842
subnet-3829c174
subnet-bb0f1db5

Security

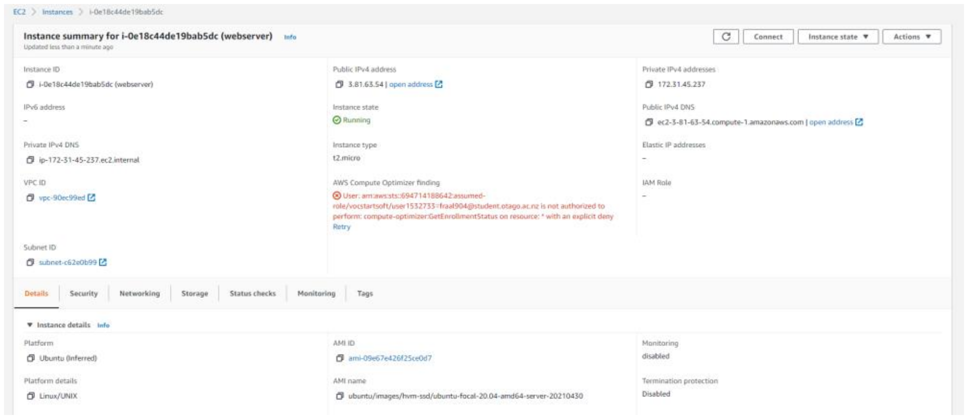
VPC security groups
rds-security (sg-066d9086fce6d135a)
(active)

Publicly accessible
Yes

Certificate authority
rds-ca-2019

Certificate authority date
August 23, 2024, 05:08 (UTC±5:08)

Our EC2 instance page displayed as such on the AWS console:


EC2 > Instances > i-0e18c44de19bab5dc

Instance summary for i-0e18c44de19bab5dc (webserver) Info
Updated less than a minute ago

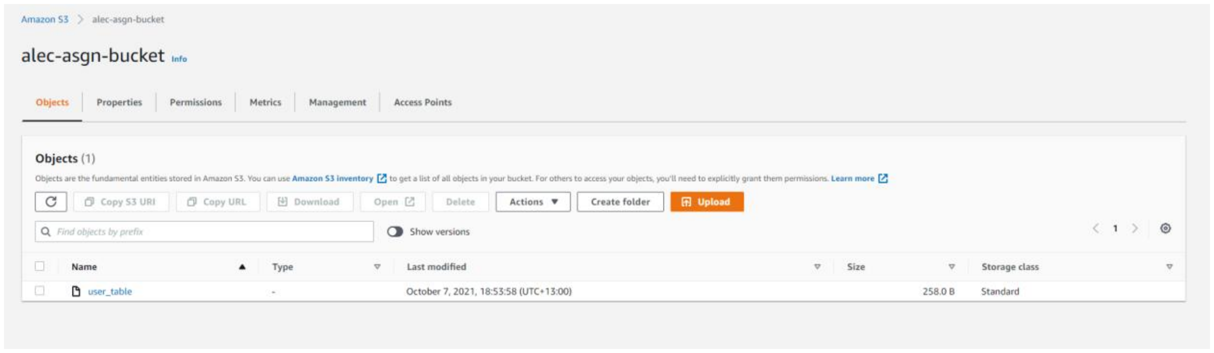
Instance ID i-0e18c44de19bab5dc (webserver)	Public IPv4 address 3.81.63.54 | open address	Private IPv4 addresses 172.31.45.237
IPv6 address –	Instance state Running	Public IPv4 DNS ec2-3-81-63-54.compute-1.amazonaws.com | open address
Private IPv4 DNS ip-172-31-45-237.ec2.internal	Instance type t2.micro	Elastic IP addresses –
VPC ID vpc-90ec99ed	AWS Compute Optimizer finding User: arn:aws:sts::694714188642:assumed-role/vocstartsoft/user1532733=fraal904@student.otago.ac.nz is not authorized to perform: compute-optimizer:GetEnrollmentStatus on resource: * with an explicit deny Retry	IAM Role –
Subnet ID subnet-c62e0b99		

▼ Instance details Info

Platform Ubuntu (Inferred)	AMI ID ami-09e67e426f25ce0d7	Monitoring disabled
Platform details Linux/UNIX	AMI name ubuntu/images/hvm-ssd/ubuntu-focal-20.04-amd64-server-20210430	Termination protection Disabled

A Proof of Execution of Our S3 Backups

Below is our S3 bucket after setup, with data from our first test backup.



Amazon S3 > alec-asgn-bucket

alec-asgn-bucket Info

Objects | Properties | Permissions | Metrics | Management | Access Points

Objects (1)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

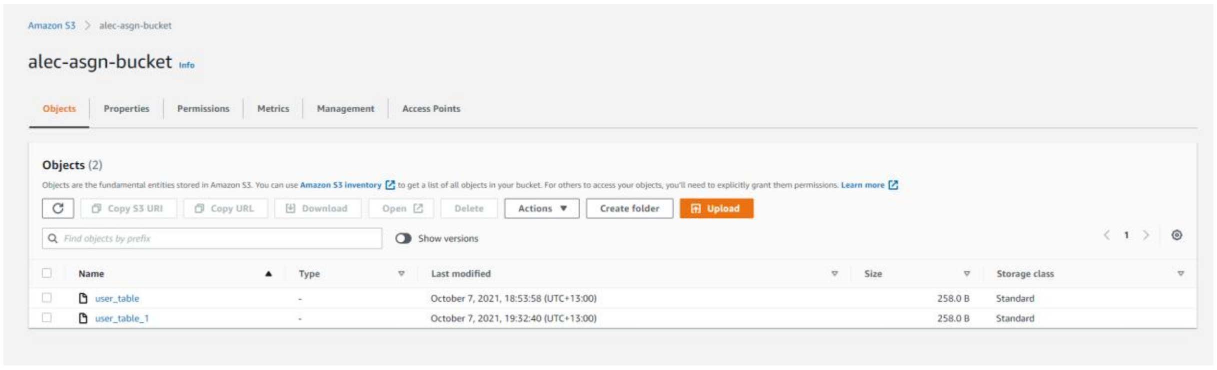
Copy S3 URI | Copy URL | Download | Open | Delete | Actions | Create folder | Upload

Find objects by prefix — Show versions

	Name	Type	Last modified	Size	Storage class
	user_table	-	October 7, 2021, 18:53:58 (UTC+13:00)	258.0 B	Standard

Our bucket with our first backup

After pressing "Export Table" the bucket looked like this



Amazon S3 > alec-asgn-bucket

alec-asgn-bucket Info

Objects | Properties | Permissions | Metrics | Management | Access Points

Objects (2)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

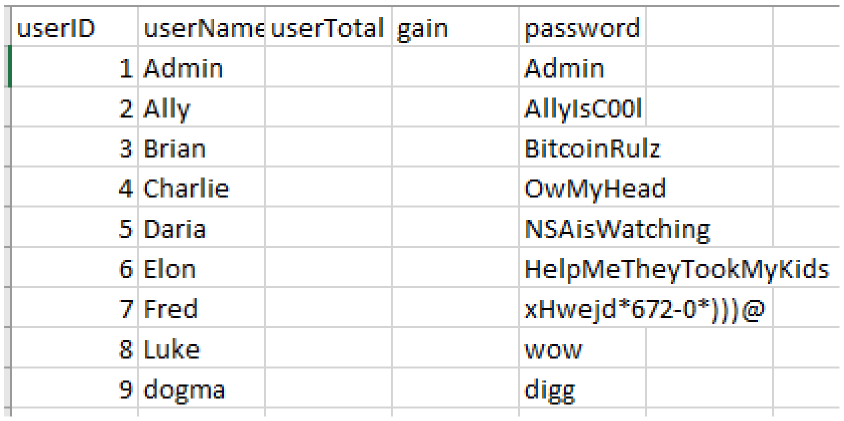
Copy S3 URI | Copy URL | Download | Open | Delete | Actions | Create folder | Upload

Find objects by prefix — Show versions

	Name	Type	Last modified	Size	Storage class
	user_table	-	October 7, 2021, 18:53:58 (UTC+13:00)	258.0 B	Standard
	user_table_1	-	October 7, 2021, 19:32:40 (UTC+13:00)	258.0 B	Standard

The bucket after multiple backups

Which contained a CSV of our user data below



userID	userName	userTotal	gain	password	
1	Admin			Admin	
2	Ally			AllyIsC00l	
3	Brian			BitcoinRulz	
4	Charlie			OwMyHead	
5	Daria			NSAisWatching	
6	Elon			HelpMeTheyTookMyKids	
7	Fred			xHwejd*672-0*)))@	
8	Luke			wow	
9	dogma			digg	

The database after .csv export

The reason we have included the above walkthrough of our S3 backup is as proof of execution, which unfortunately will not work after three hours due to the expiration of the CLI session data.

Reaching and Interacting with the Service

The webserver we have created is using our code from the first assignment which can be found here: https://github.com/Nexus112/COSC349-asgn1 the modifications made to the user site was only a change in the conn.php file where the mysql connection details have been changed to reflect our RDS. This change from the old to new is not specifically shown with the git commits as it was changed on setup of the website to get the webserver initially running. No other modifications had to be made to the user site for it to be operational on the EC2 instance.

The RDS uses also uses sql code from the first assignment that initializes the database and table required for the user site, no modifications were needed when migrating this from the first assignment to RDS.

The application can be accessed via http://3.81.63.54/site.php. To use the service a user will first need to log in or register. To see test data, please log in to the user "Admin" with password "Admin" (without quotations). Once logged in, a user can add the coins they wish to see displayed, along with their current value. If a user wishes to remove a coin from their wallet, they simply click "remove coin" next to the target coin. The user may also edit their current balance by clicking "Edit Amount" next to the desired coin.

Examples of these steps can be found below.

Example Usage of the Service

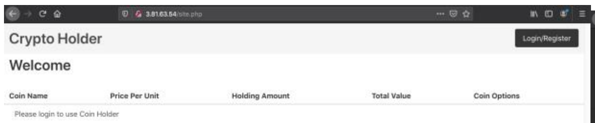


Figure 1: The Landing Page

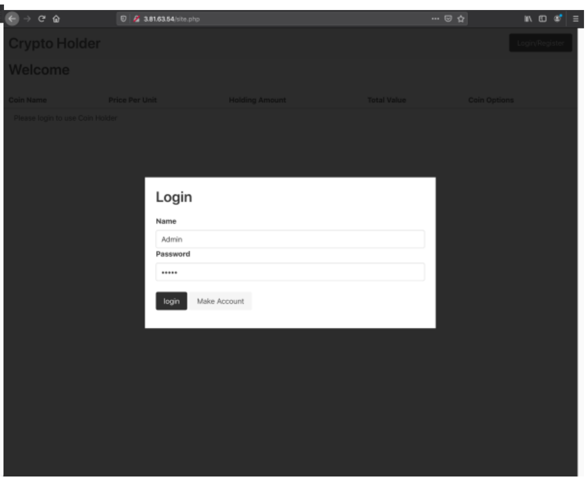
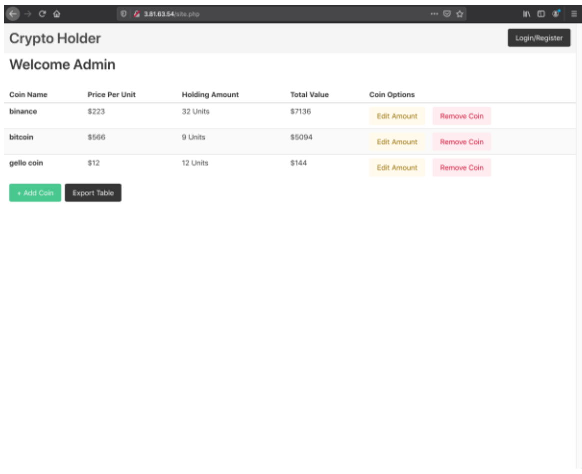


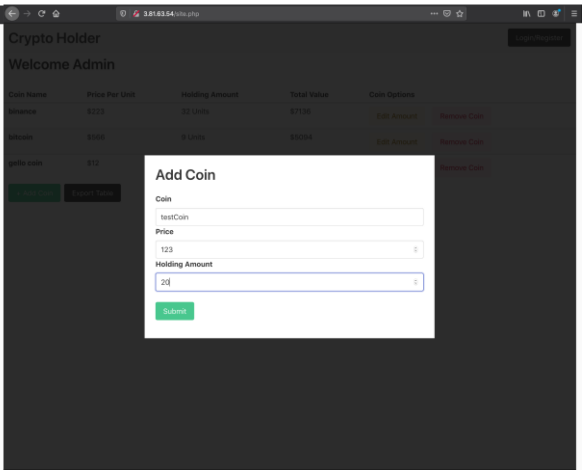
Figure 2: Signing In


3.81.63.54/site.php

Crypto Holder

Welcome Admin

Coin Name	Price Per Unit	Holding Amount	Total Value	Coin Options
binance	$223	32 Units	$7136	Edit Amount Remove Coin
bitcoin	$566	9 Units	$5094	Edit Amount Remove Coin
gello coin	$12	12 Units	$144	Edit Amount Remove Coin


3.81.63.54/site.php

Add Coin

Coin: testCoin

Price: 123

Holding Amount: 20

Figure 3: The Homepage of a User with Test Data *Figure 4: Adding a coin*

How the VMs Interact

Logging in

When a user logs in on the website, an SQL query is sent to the RDS which looks for a user with that username and sees if the associated password matches user input. If the passwords do not match, or if no user is found, the appropriate error page will be displayed.

<u>Note: passwords have not been stored securely, as implementing hashes did not relate to the web-based aspect of this assignment.</u>

Adding and Removing Coins

When adding or removing coins from a user's wallet, an SQL query is sent from the EC2 instance to the RDS. This only impacts the user's wallet in the database and does not directly edit the main coin table within the database, which is responsible for storing the value of each coin.

Creating Backups

When a backup is requested via a PHP command, a copy of the database is saved as a CSV file which is then uploaded to an S3 bucket. This requires CLI user credentials to edit the contents of the bucket remotely. Initially we had planned to make these backups automatic, however we are using an AWS Educate account, and so the required user credentials to automate bucket updates are only available for three hours at a time. To send our.csv files to our s3 bucket we used the code from lab 8 using the boto3 package for python which allowed us to connect to the bucket using CLI and send over the .csv object.

Estimation of Running Costs

The RDS

According to the AWS Pricing Calculator a db.t2.micro instance deployed as Single-AZ On-Demand pricing with 20 GB of General Purpose SSD (gp2) will have an idle running cost of $2.42 USD per month. This is assuming 1% utilization, as 0% utilization is not accepted by the calculator. This cost includes a charge of $0.115 USD per GB, with 20 GB of storage.

At 100% utilization the running cost will be $14.71 USD per month. This is charging at $0.017 USD per hour, billing for 730 hours. Also included is a $0.115 USD per GB rate, with 20GB of storage.

For light usage we have assumed 20% utilization to cost $4.78 USD per month. This is charged at $0.017 USD per hour, multiplying it by 730 hours, and then multiplying the total by 0.2.

The EC2 Instance

According to the AWS Pricing Calculator an EC2 t2.micro instance running Linux being charged with on-demand pricing, with 8GB of General Purpose SSD (gp2) storage, the idle running cost is $0.80 USD per month. This assumes by idle we mean 0% utilization of the instance.

At 100% utilization this monthly running cost is $9.27 per month. This assumes 730 hours of usage billed at $0.0116 USD per hour.

Under light usage, which I have estimated to mean 20% utilization, the monthly running cost of the EC2 instance is $2.49 USD per month which assumes 146 hours of usage in a month billed at $0.0116 USD per hour.

The S3 Bucket

S3 buckets are not charged by run time, but instead by storage used within the bucket. As this application is unlikely to use over 50 TB of data, the cost of the bucket is $0.023 per GB per month the bucket runs. However, if this application were adopted by a large company which expected to store a lot of user data, the backup would cost $0.023 per GB per month of the first 50 TB, $0.022 USD per GB per month of the next 450TB, and $0.021 per GB per month for anything over 500TB.

Total

If all instances are idle, the monthly charge comes to $3.22 USD per month plus $0.023 per GB of data within the bucket per month, which will fluctuate over time depending on how many users the site has. Over time this would grow, but if the site is completely idle, we expect the storage size to be negligible.

The cost of full utilization comes to $23.98 USD per month plus the above storage cost of the bucket, which we assume will be greater under full utilization as it implies many users requiring data to be stored.

Under light usage we estimate the running cost of all of these instances together to be $7.27 USD per month, with the additional cost of $0.022 USD per GB per month for data stored in the S3 bucket backup.