CS3204 | Cloud Infrastructure and Services

Lab 2: Web Services in the Cloud

Content

- Goals & Objectives
- Benefits of the AWS

Lab Work Content

- Task 1: Cloud Web Instance
    - AWS & Its €1 Fee
    - Rest-API & Fake Data
    - Finding Salvation in Mining
- Task 2: Cloud Web Database
    - Form Submission in PHP
    - The choice of PHP & working with MyAdmin
- Experience gained from using, testing and dubbing new software

Goals & Objectives

My goal for this 3ish week lab is to put into practice the theoretical knowledge I have acquired from lectures in relation to working with the Cloud environment and how different companies use this service to speed up initial development of their projects.

I will accomplish this by documenting my progression through the set-up of one of Amazon's Web Services (AWS) called Elastic Beanstalk, extracting data from an API source, creating my own database and providing a web-based method to upload files to it.

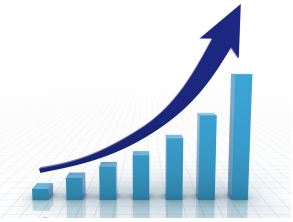
By the end of this Lab work, I hope to have enhanced my understanding of tools that can be used to speed up development of website building as well as to familiarise myself with services such as AWS & Elastic Beanstalk.

All code can be found in my GitHub Repository by following this link:

https://github.com/Passe-Sleeper/UNI-Public/tree/main/Year3/CS3204/Lab2

Benefits of AWS

Amazon Web Services is a nextensive provider of cloud based tools that allow various people to work on projects and web applications without the need for having their own server. This leads to many benefits which include: Scalability, Flexibility, Cost-effectiveness, Reliability, etc.



A bit of detail on each of these:

Scalability: AWS allows you to easily scale your infrastructure up or down based on your needs. You can quickly add or remove resources as your business demands change, without any major disruptions.

Flexibility: AWS offers a wide range of services that cater to various business requirements. Whether you need computing power, storage, databases, machine learning, or analytics, AWS has a service to fulfil your needs.

Cost-effectiveness: With AWS, you only pay for what you use. There are no long-term contracts or upfront costs. This pay-as-you-go model can help you save costs by eliminating the need for investing in expensive hardware and infrastructure.

Reliability: AWS has a global infrastructure that is designed to provide high availability and reliability. They offer multiple data centres worldwide, ensuring that your applications and data remain accessible even in the event of a failure.

There are also a couple of disadvantages that are also associated with similar benefits, such as: Complexity, Dependency, Cost, Security, etc.

A bit on each one of these:

Cost: While AWS offers a pay-as-you-go pricing model, costs can add up quickly. If you don't carefully monitor and manage your usage, you may end up with unexpectedly high bills. Additionally, certain services and features may incur additional charges, leading to hidden costs.

Complexity: AWS is a complex platform with a steep learning curve. It requires a solid understanding of various services, configurations, and management practices, which can be overwhelming for beginners or small organisations with limited technical expertise.

Security: As with any cloud service provider, there are security concerns when using AWS.

Dependency: By using AWS, you become dependent on their services and infrastructure. This can be a disadvantage if AWS experiences outages or disruptions, which could impact your business operations. It's important to have contingency plans in place to mitigate such risks.

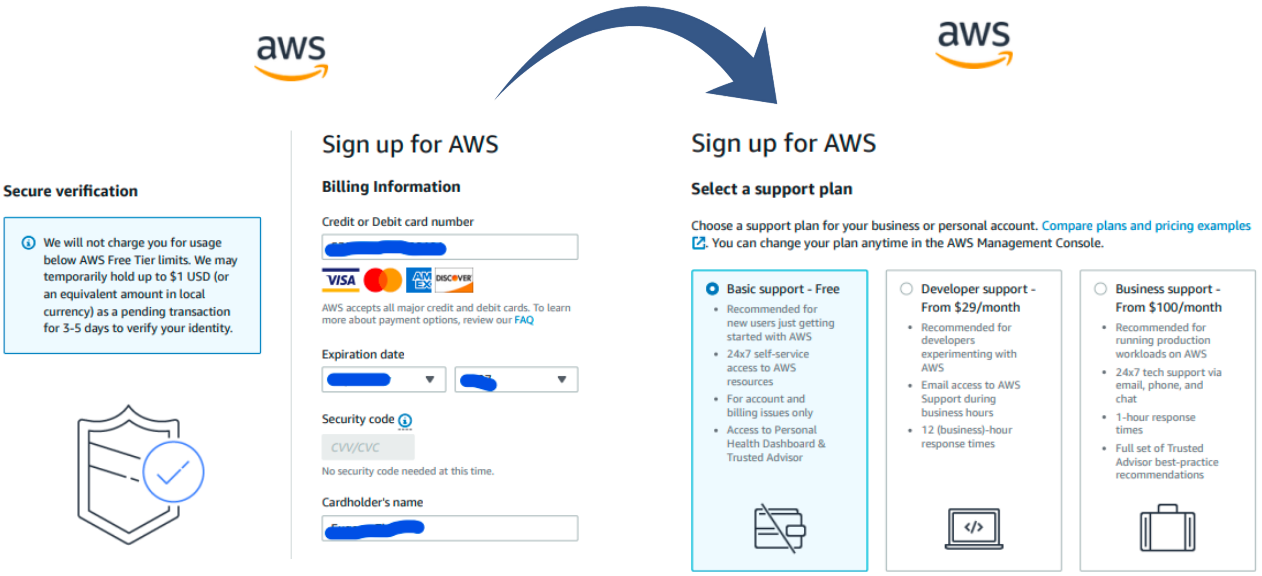
Overall, AWS offers a very wide branching service that takes time to master, is great for start-ups, but once the company grows, it is probably in their interest to start up their own website & create their own hosting services for security and cost reasons.

Lab Work Content

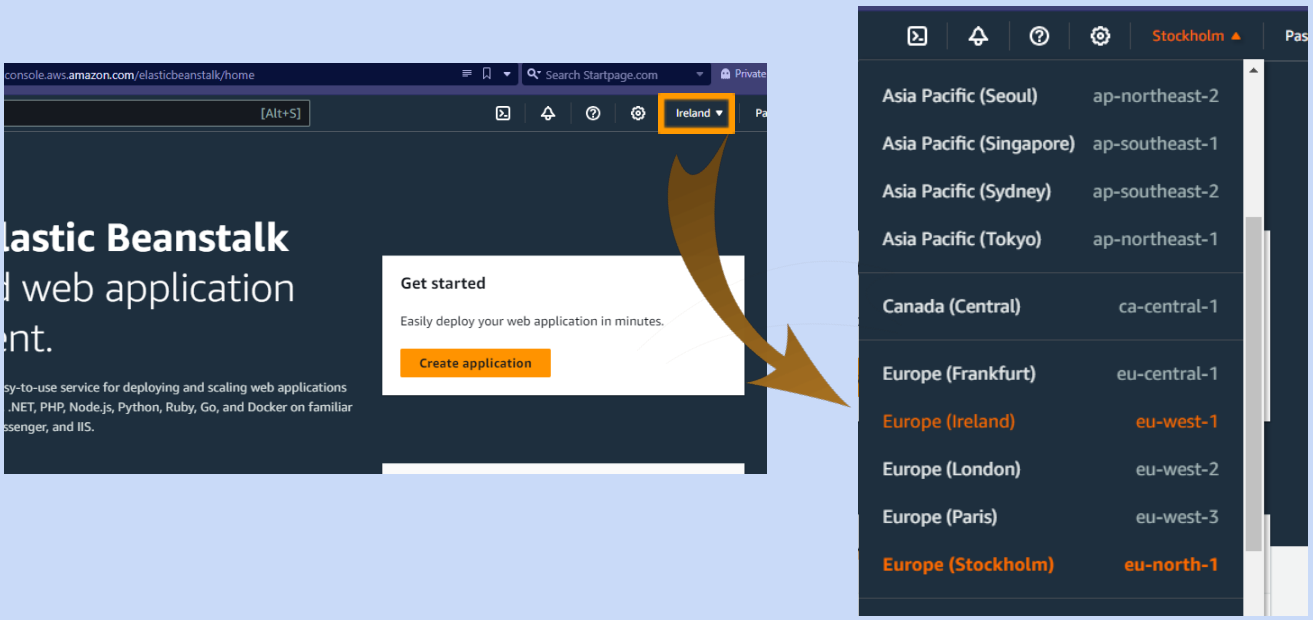
Task 1: Cloud Web Instance

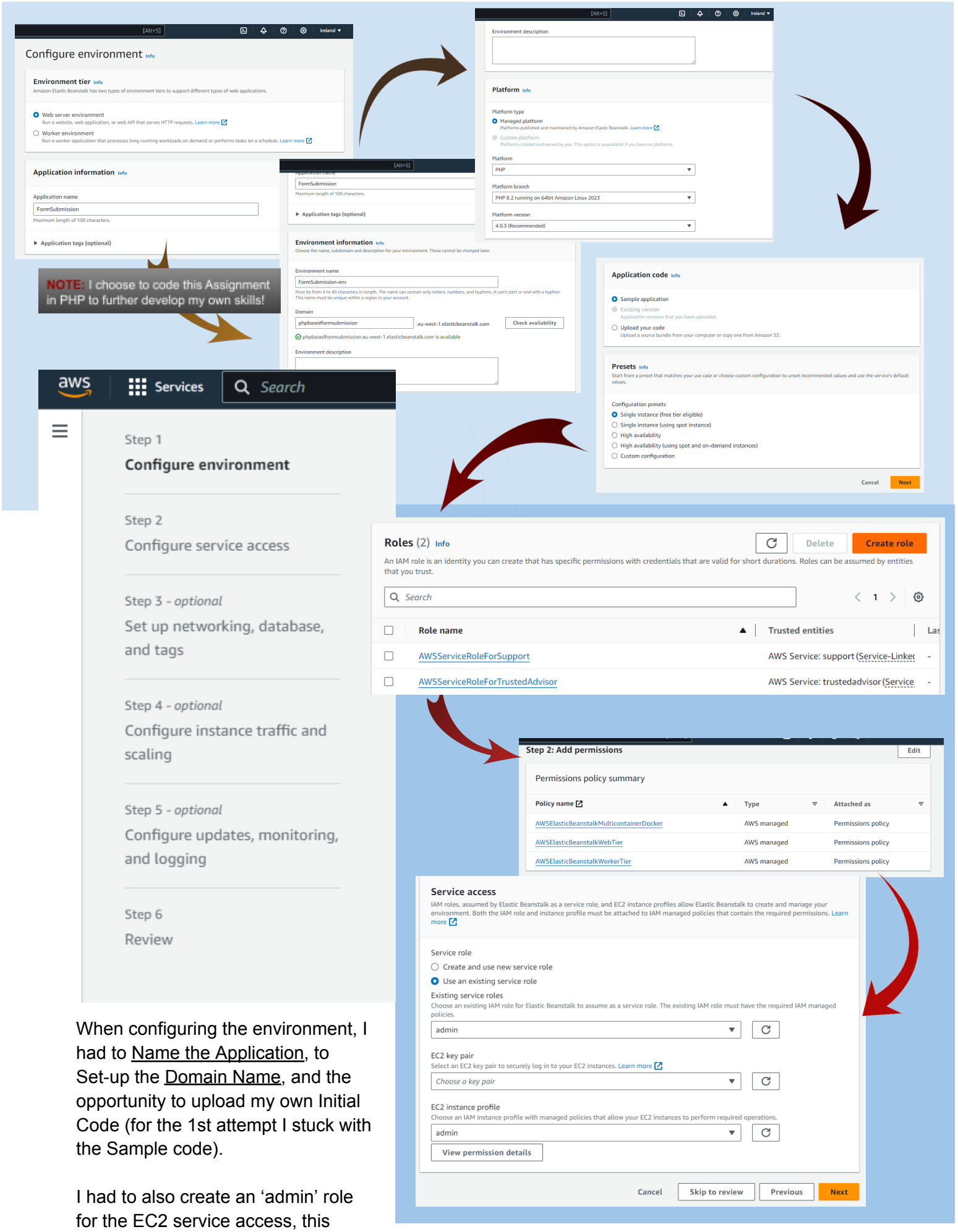
AWS & Its €1 Fee

When initially following the lab instructions of signing up to AWS services, I was forced to create an account and even pay a deposit of €1! I was quite surprised at this, but, since it was just €1, it should be fine…



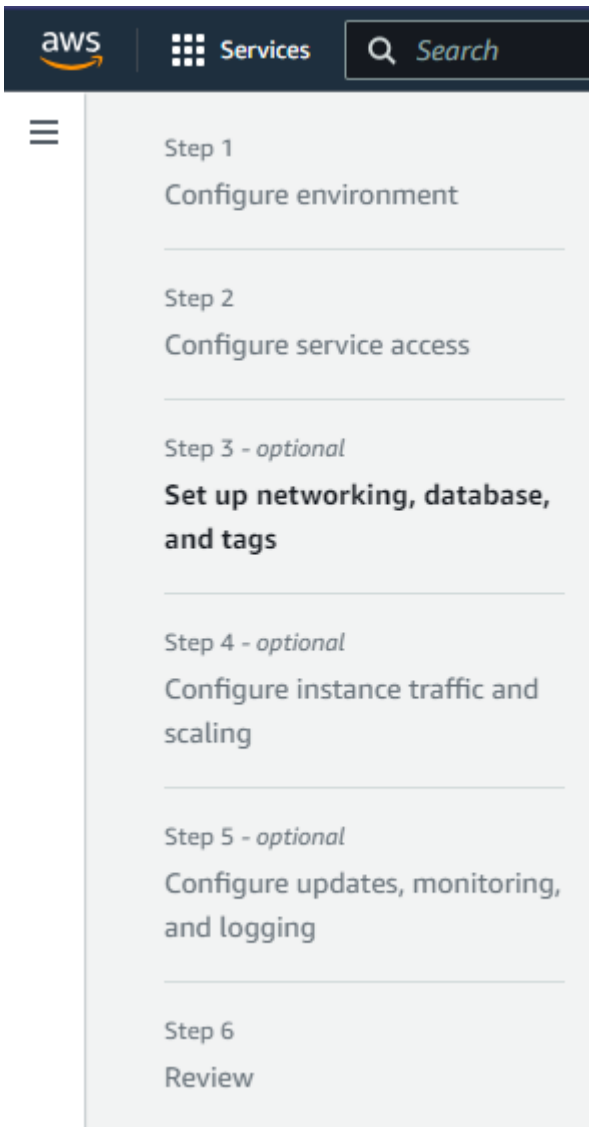
After Registration of my Account, I continued to follow the same procedures as stated in the notes, that being to create my own Web-based application and set the Region!



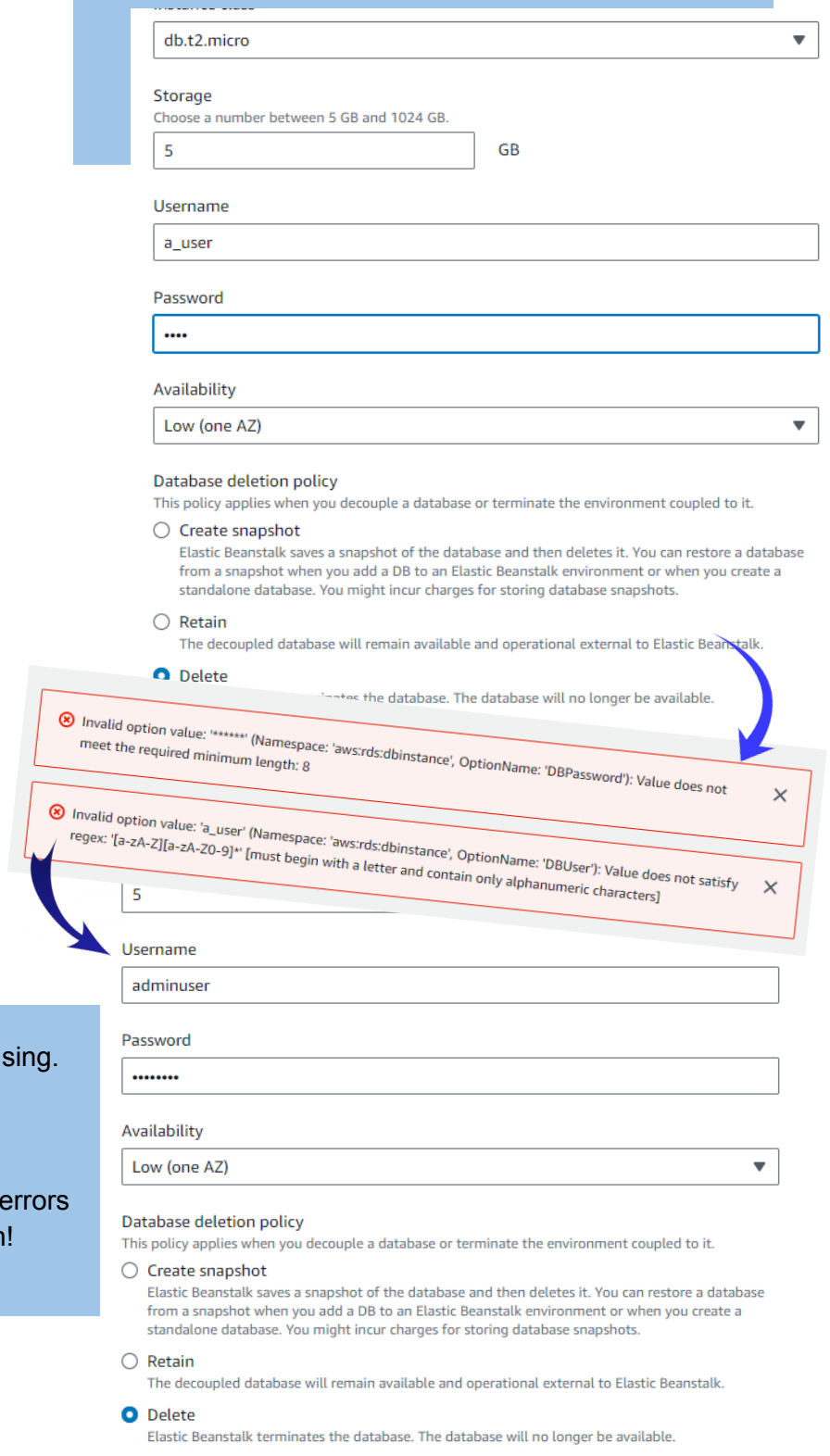


When configuring the environment, I had to Name the Application, to Set-up the Domain Name, and the opportunity to upload my own Initial Code (for the 1st attempt I stuck with the Sample code).

I had to also create an ‘admin’ role for the EC2 service access, this allowed me to use the web service application at a later stage!



Next I had to Set-up the Database connection as this would be used Task 2!

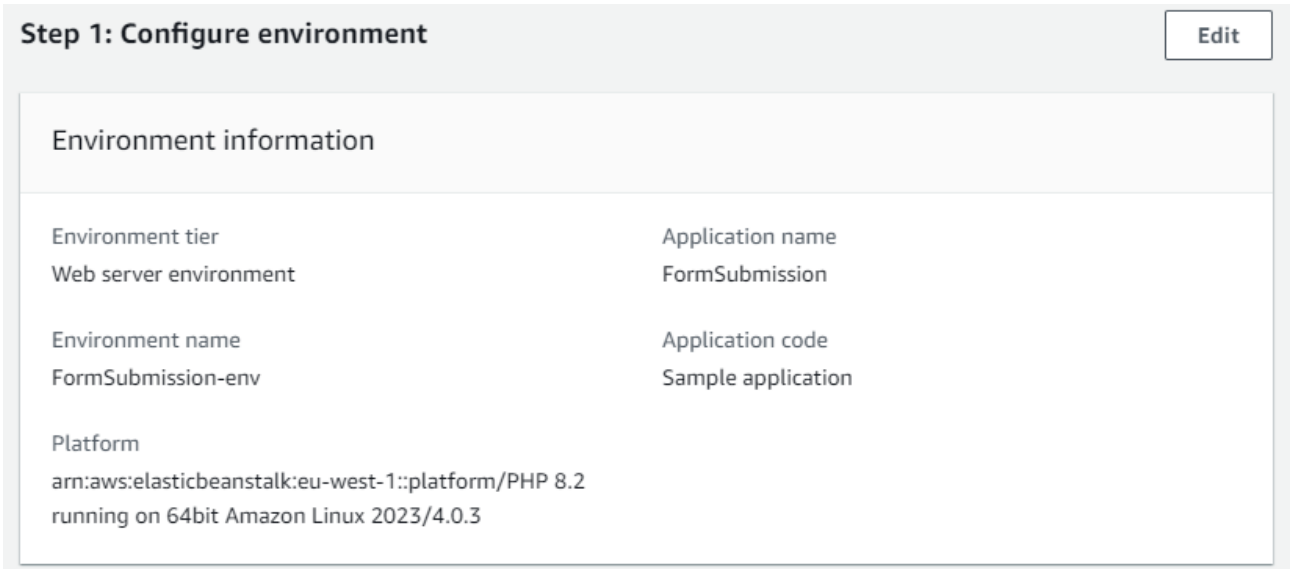


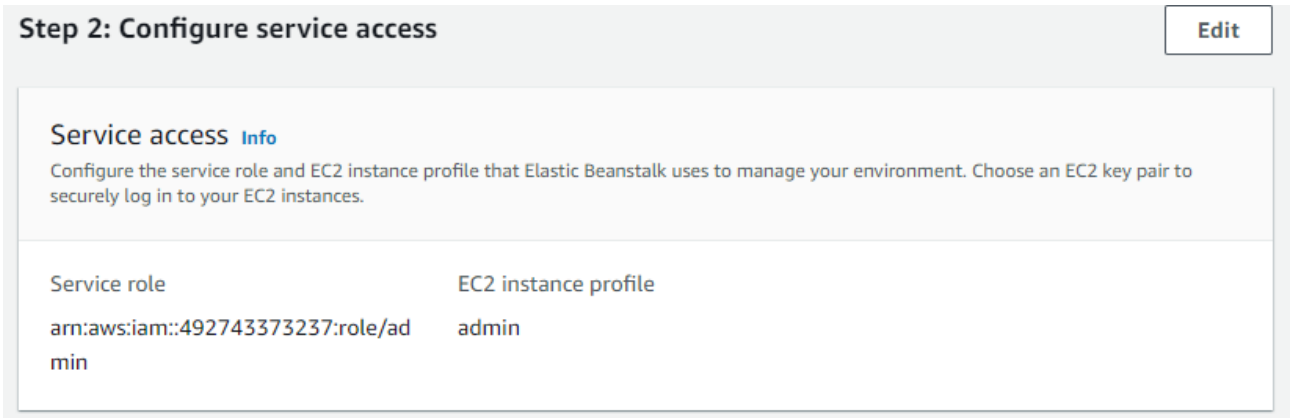
I added the Database that I would be using. Created a Username & Password as required of me.

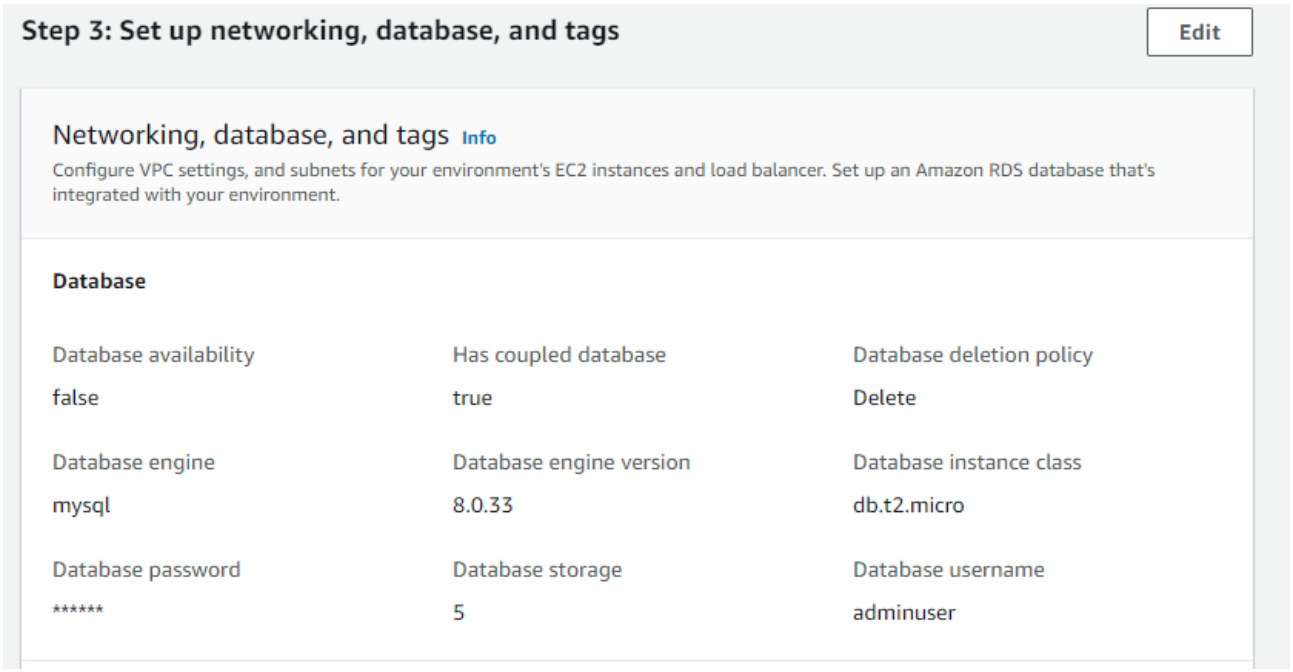
Of course, coming across a few minor errors in Username & Password length - ouch!

Steps 4 & 5, were just skipped as they were not necessary for my project, but I watched a tutorial to help me understand what they were for.

Before we continue, let us perform a quick **Review** of what we have created/set-up!

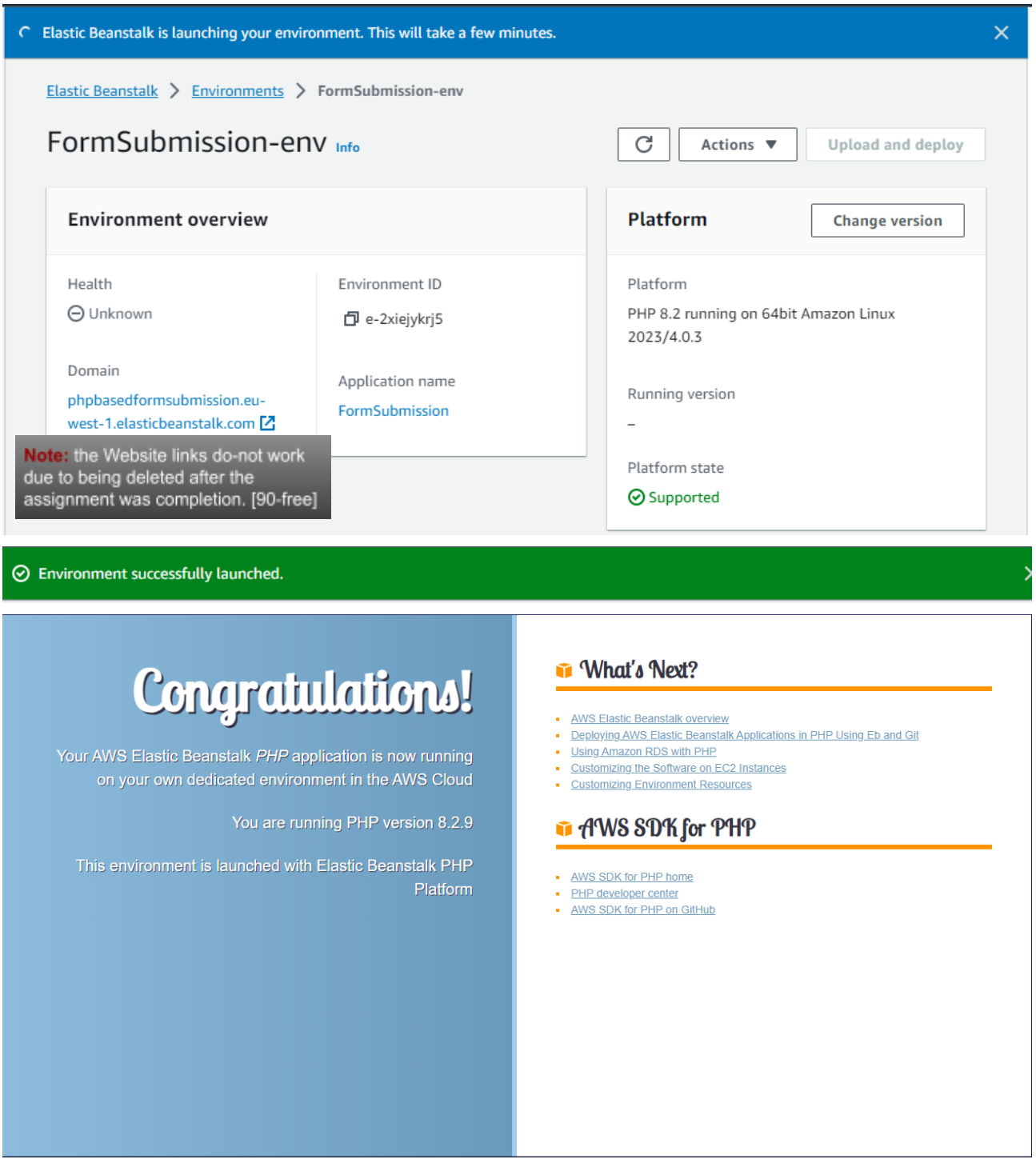






Sample Application Code

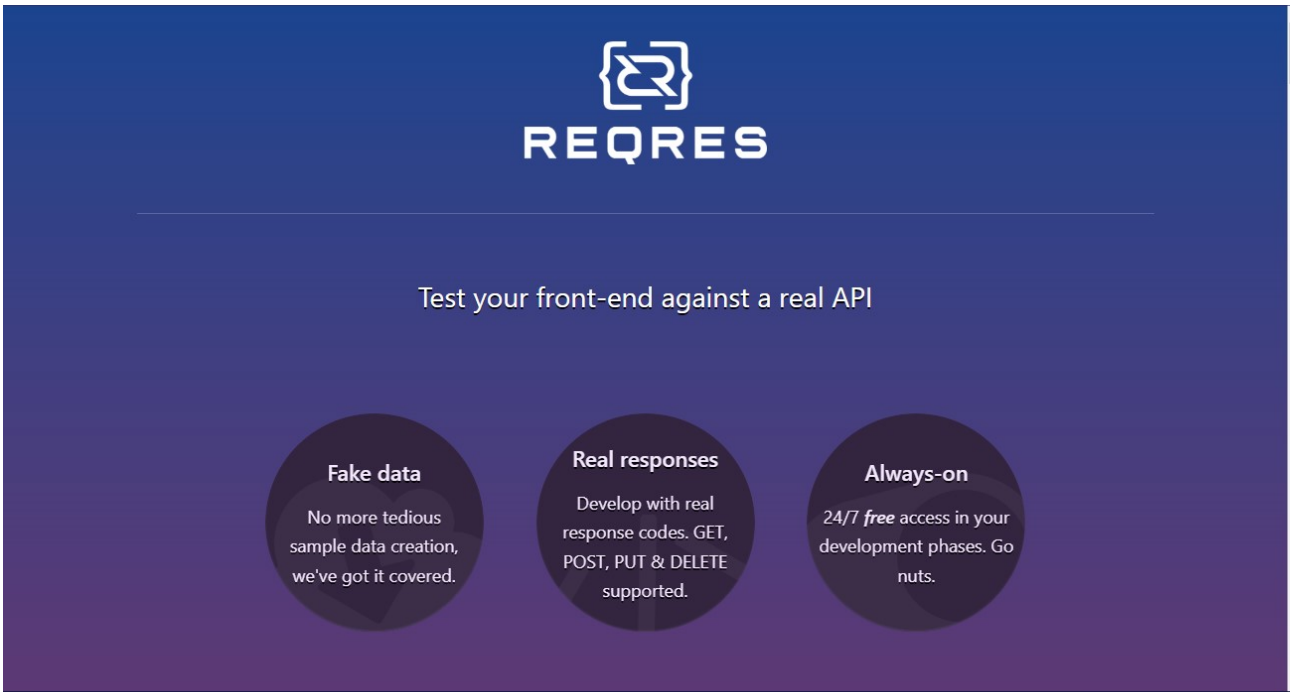
After the Initial set-up, the Web Environment was ready to launch the application.



Excellent, that procedure was relatively straightforward. It is now time to upload my own project onto the Cloud to see how it goes!

Rest-API & Fake Data

Initially I wished to import data from a website called [Reqres.in](), and in order to do so, I followed some online tutorials on how the Rest-API worked. I even coded up a small application to extract the code from the database and to write out everything it collected. The result was consistent with what other user's of this dataset had done in the past.



I wished to get access to the following JSON layout data-set:

This task was simple in nature, but required a lot of background work to accomplish.The reason for that was my choice in programming language - PHP.

My choice | PHP

Why pick PHP and not stick to Python? This module, Cloud Infrastructure and Services, teaches you about various web services; How they work, Why they work and most importantly: Why we need them. This module teaches you in what direction the current modern (cloud) world is moving in, as well as the history of what came before!

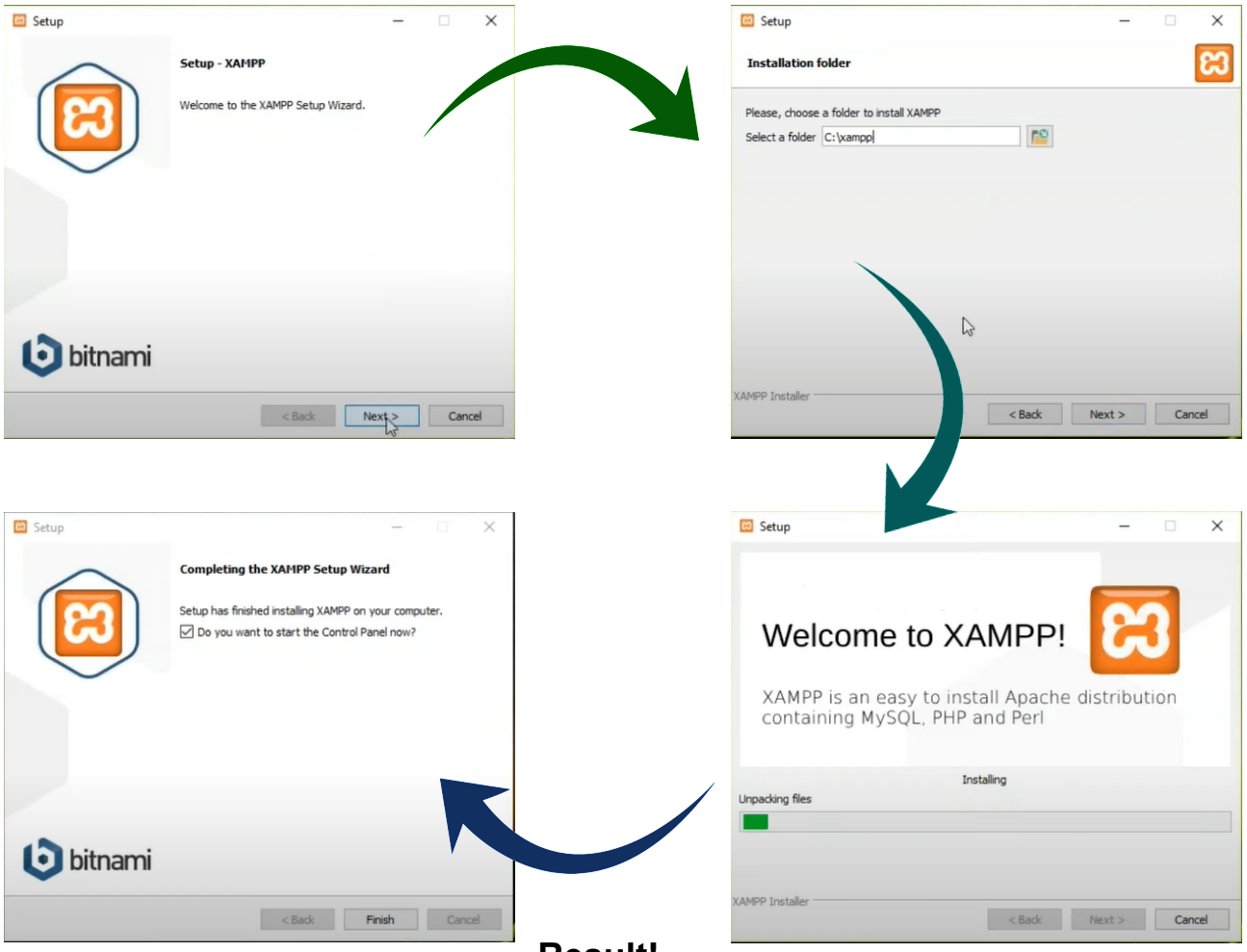
PHP as a programming language is old and has its disadvantages: Poor Security Features, Limited Debugging Tools, Inability to Modify Core Behaviour, Not as Versatile and that there are easier languages to learn (due to complex writing).

However, I believe that this assignment is the perfect opportunity for me to step into learning this new language! A tutorial here & a tutorial there, I should be able to write a simple website with it!

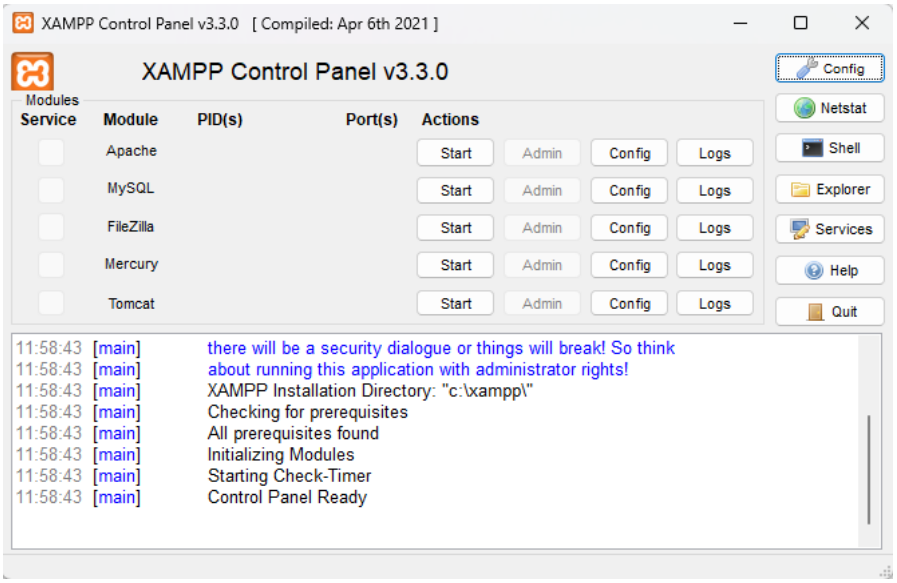
Local Testing with XAMPP

While doing my research online, I found suggestions/recommendations, for if you are using PHP for Deployment purposes, it is suggested to run an Apache server and then run your application using it rather than run your code via terminal (as with Python).

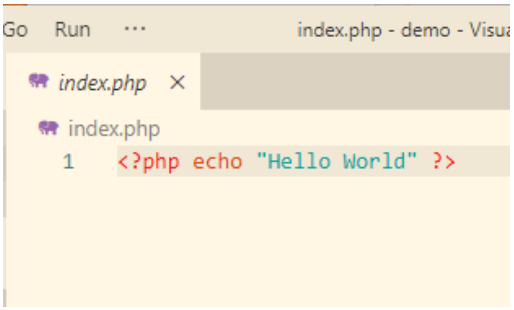
To test and consequently learn all of this for myself, I went through a XAMPP Installation Guide which can be found here and watched a Youtube video to confirm.

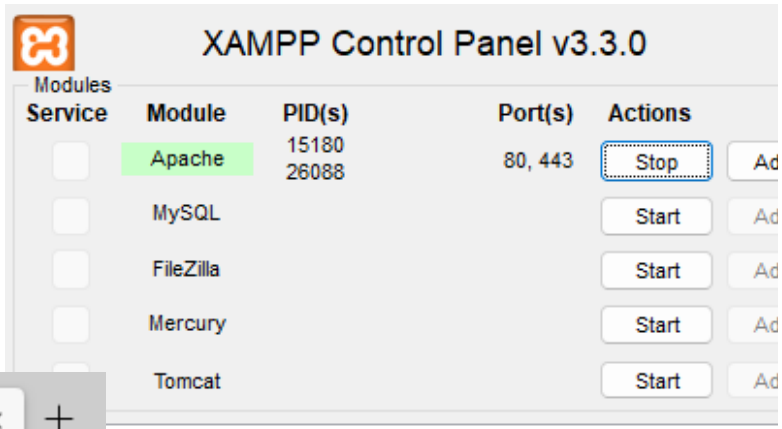


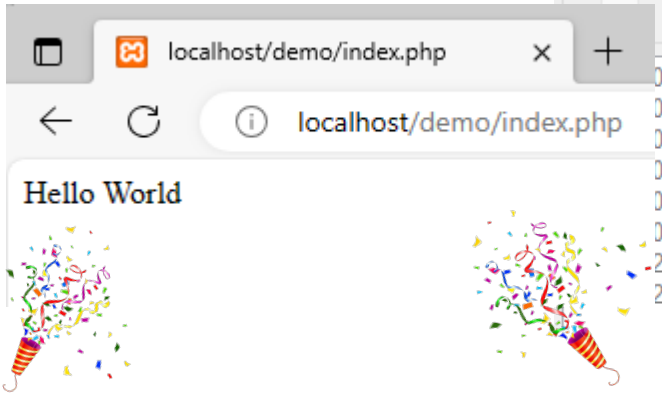
Result!



After writing the generic lines “Hello World!” and starting the base Apache server, I was able to launch my localised Web-page!

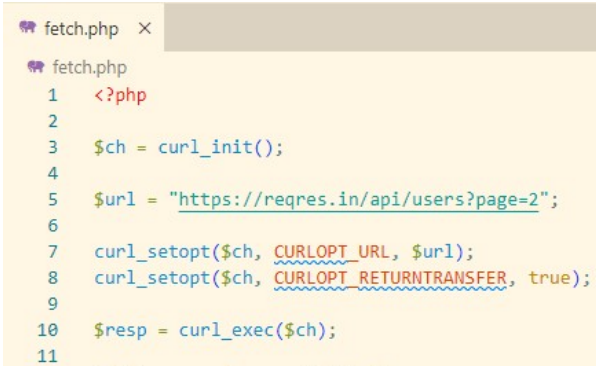


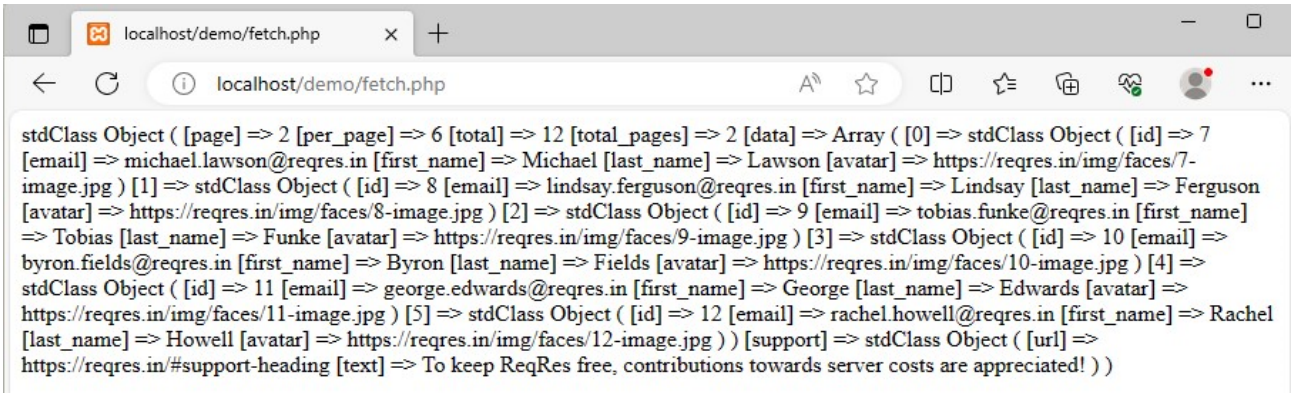




After the initial success, I began writing software to work with the Resque Database.

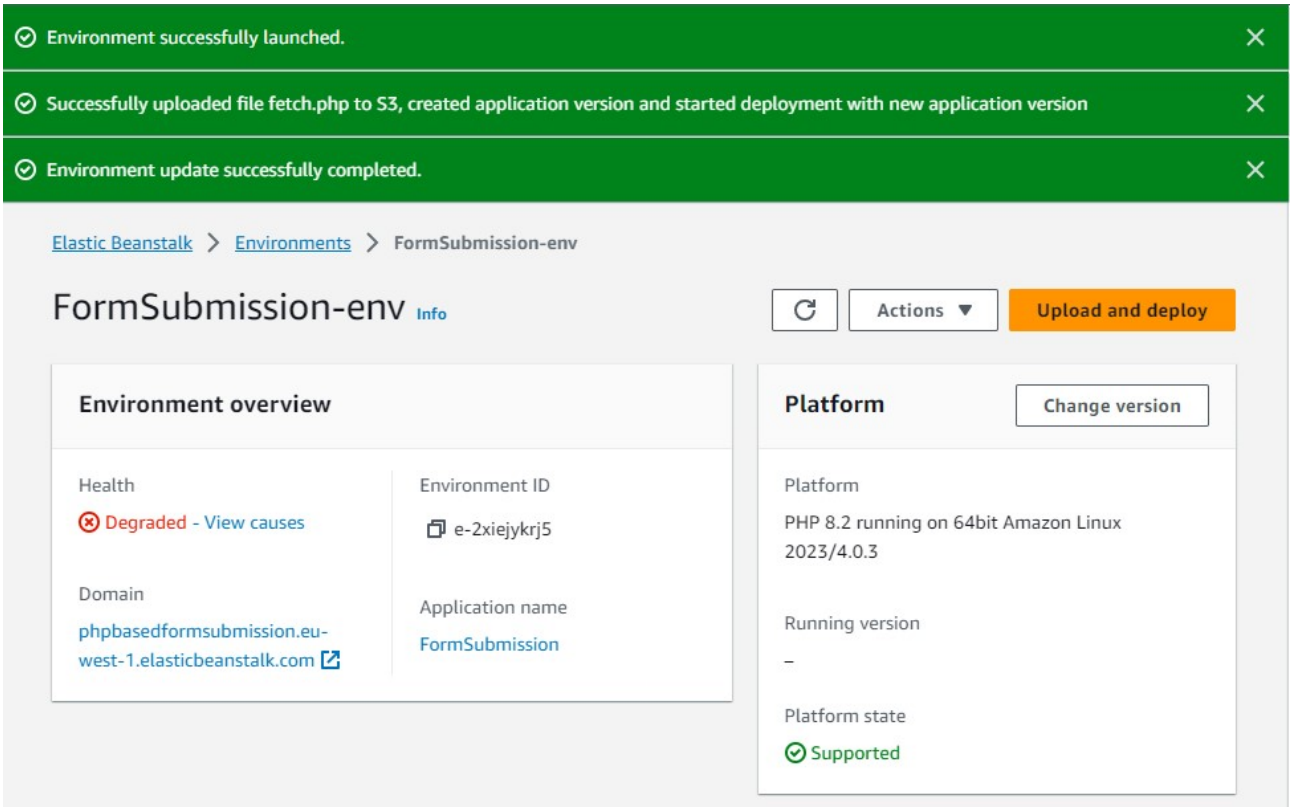
The result was the following code & output:



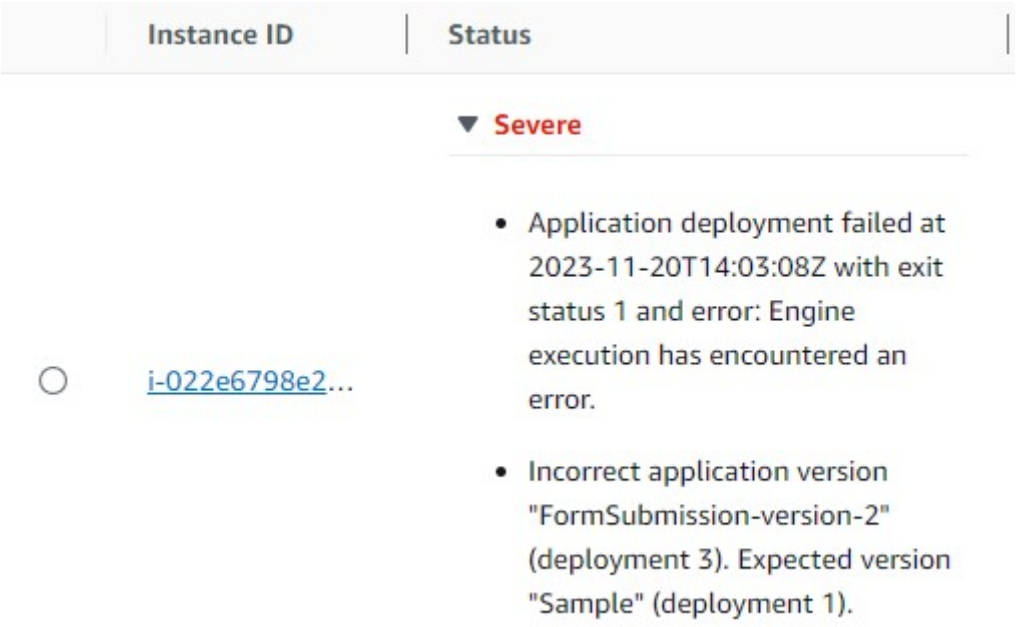


A Break from AWS

I attempted to bring this same file to the AWS server. However, when attempting to run it on the servers, I encountered a lot of errors in relation to the file health being Degraded.

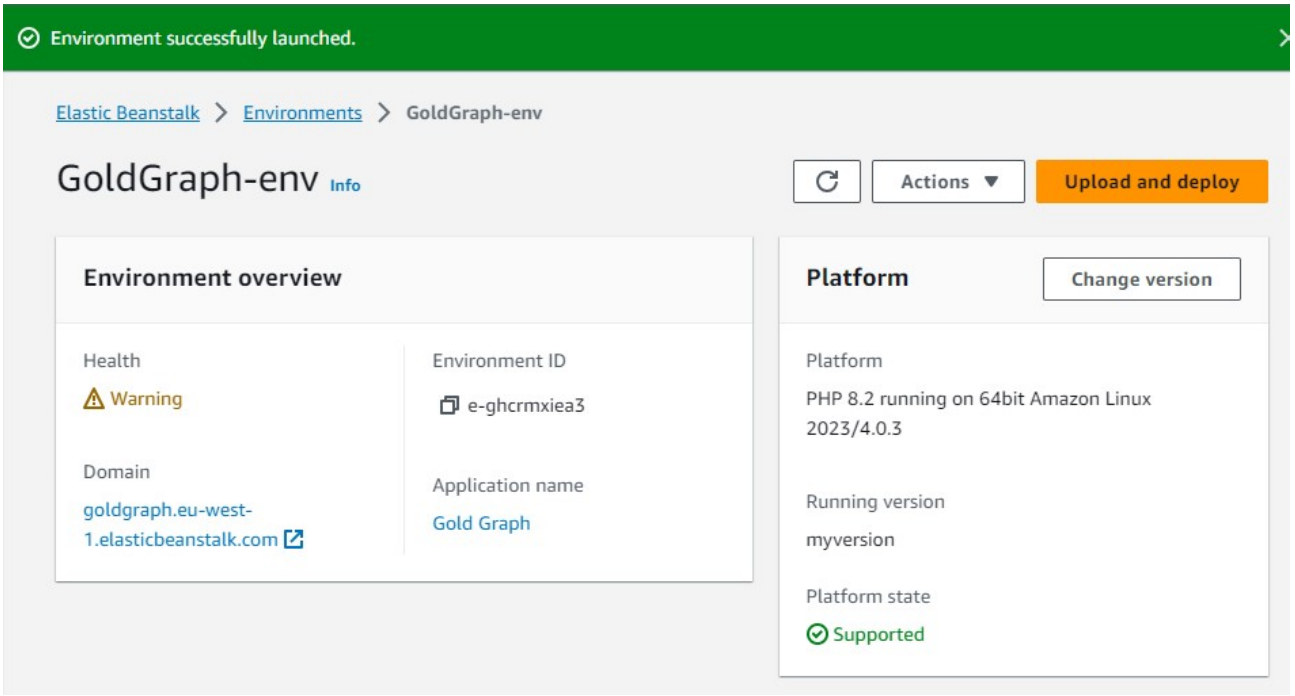


I tried to resolve the issue both via my own efforts and by googling excessively. Unfortunately, nothing worked.

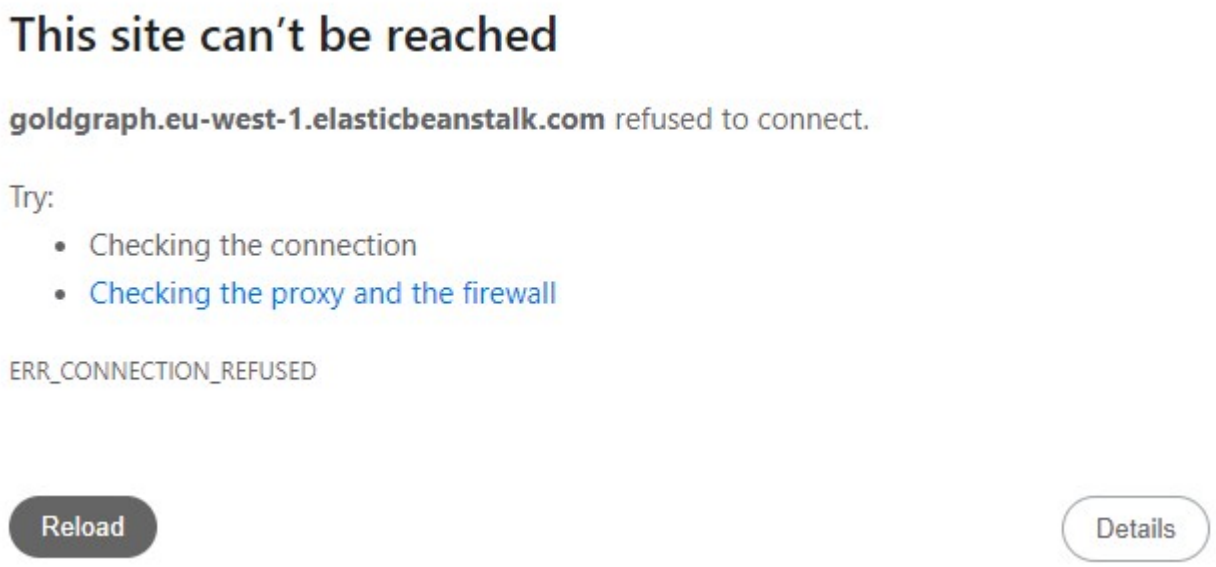


The conclusion was that the selection of Sample code instead of my own, at the very start of the Environment creation process led to these errors!

I recreated the Environment and uploaded my own file instead of using the provided Sample file, however, I only changed the error from Degraded to Warning status.



Even worse, when I tried accessing the website, nothing was working!!

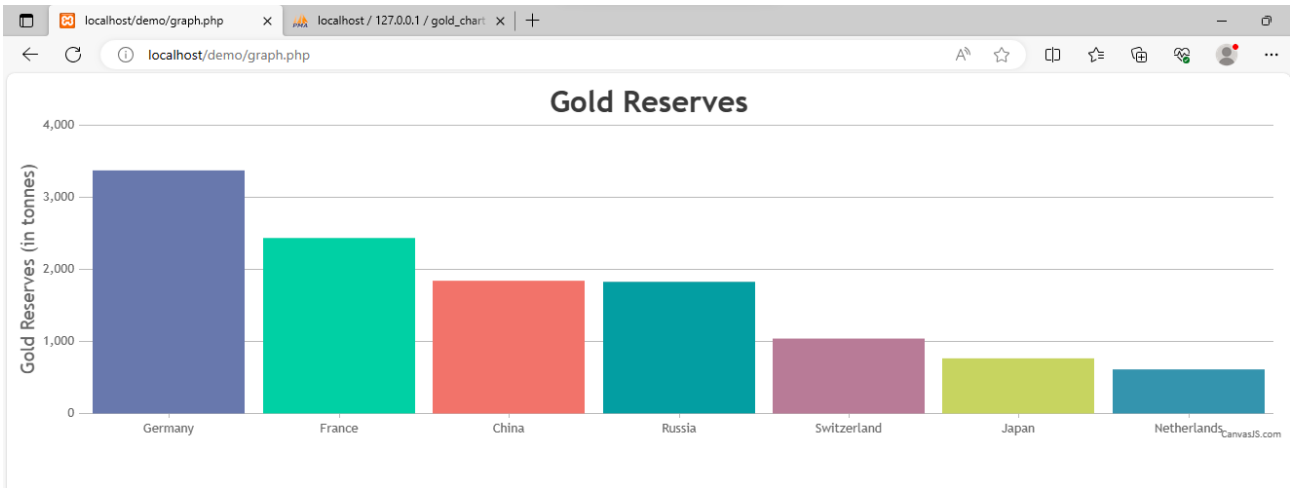


As none of this was working, I decided to not get frustrated with it and to return back to the Local Host Apache server and continue working on my Project.

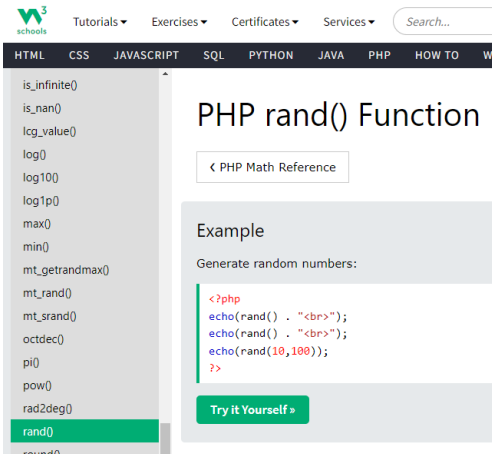
Who knows.. Maybe I will come across a solution later…?

Finding Salvation in Mining

As AWS had given up on me, I went and learned how to create graphs and draw data points on them. This was a very fun and stress relieving exercise for me! I found a cool library online called Canvas.js that contains heaps of different graphs and the style sheets needed to run them. I built a static model of the Gold Reserves of a couple of countries! This was the result:

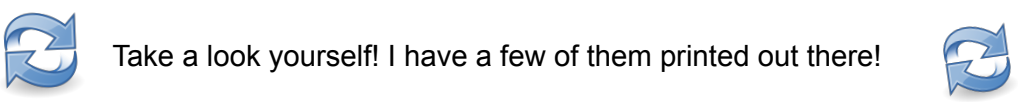


It looked very neat & I liked the colour scheme, however, it was quite boring to refresh the page and see the same result every single time! To make it more entertaining, I modified my data points to instead provide me with randomised results to my processes!




```php
$dataPointsRandom = array(
    array("y" => rand(123.47, 3691.43), "label" => "Germany" ),
    array("y" => rand(1435.21, 2739.43), "label" => "France" ),
    array("y" => rand(1942.43, 2042.23), "label" => "China" ),
    array("y" => rand(1942.43, 2042.23), "label" => "Russia" ),
    array("y" => rand(800.12, 1285.39), "label" => "Switzerland" ),
    array("y" => rand(1265.25, 2017.30), "label" => "Japan" ),
    array("y" => rand(412.453, 803.02), "label" => "Netherlands" )
);
```

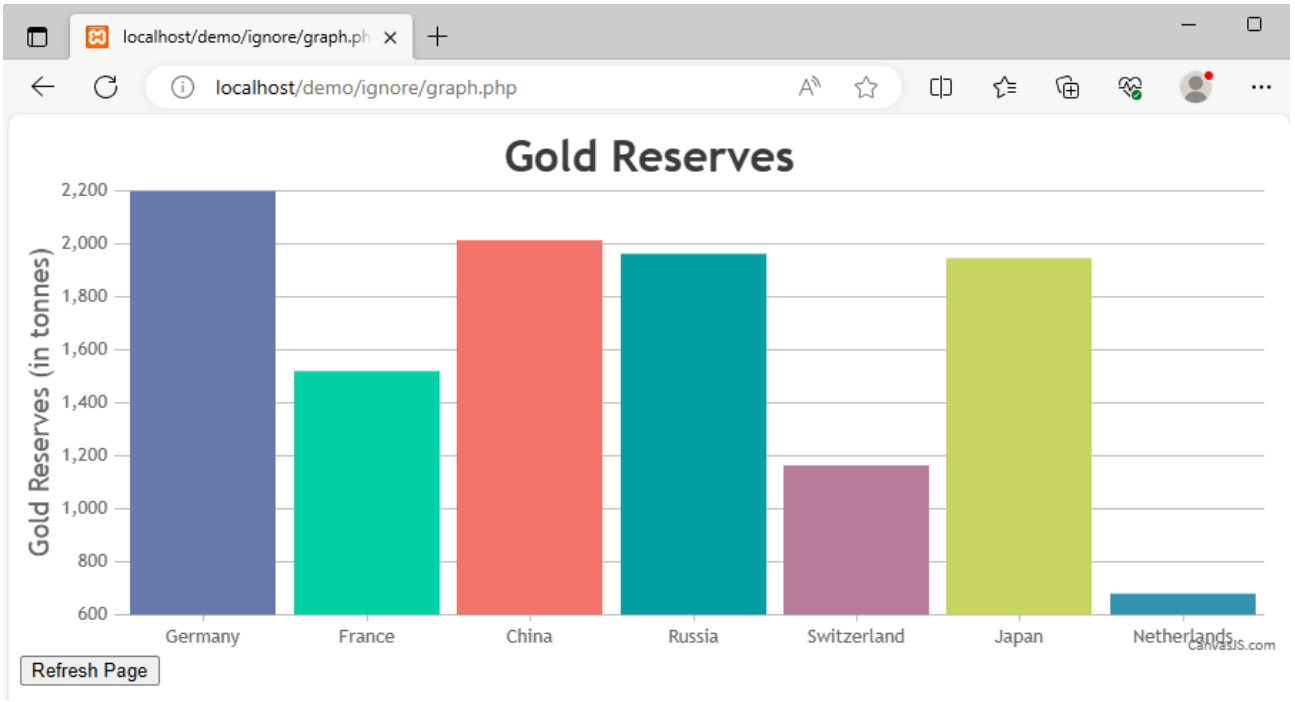
This allowed me to refresh the page and get a different graph every time!



Take a look yourself! I have a few of them printed out there!


localhost/demo/graph.php
localhost / 127.0.0.1 / gold_chart
localhost/demo/graph.php
Gold Reserves
Gold Reserves (in tonnes)
3,000
2,000
1,000
0
Germany
France
China
Russia
Switzerland
Japan
Netherlands
CanvasJS.com
localhost/demo/graph.php
localhost / 127.0.0.1 / gold_chart
localhost/demo/graph.php
Gold Reserves
Gold Reserves (in tonnes)
3,000
2,000
1,000
0
Germany
France
China
Russia
Switzerland
Japan
Netherlands
CanvasJS.com
localhost/demo/graph.php
localhost / 127.0.0.1 / gold_chart
localhost/demo/graph.php
Gold Reserves
Gold Reserves (in tonnes)
3,000
2,000
1,000
0
Germany
France
China
Russia
Switzerland
Japan
Netherlands
CanvasJS.com

I also added a button that would Act the same way as a **Refresh**!



After working out this little side project, I returned back to AWS.

Finding Solutions in Mining

After a bit of online research and searching through the AWS Elastic Beanstalk Documentation , I managed to find another means by which to connect to Elastic Beanstalk and that was using the EB CLI. I followed the documentation and successfully installed the package `awsebcli`.

```
PS C:\xampp\htdocs\demo> pip install awsebcli
WARNING: Skipping C:\Users\Admin\AppData\Local\Programs\Python\Python311\Lib\site-packages\pip-22.3.1.dist-info due to invalid metada
ta entry 'name'
WARNING: Skipping C:\Users\Admin\AppData\Local\Programs\Python\Python311\Lib\site-packages\pip-22.3.1.dist-info due to invalid metada
ta entry 'name'
Requirement already satisfied: awsebcli in c:\users\admin\appdata\local\programs\python\python311\lib\site-packages (3.20.10)
Requirement already satisfied: botocore<1.32.0,>1.23.41 in c:\users\admin\appdata\local\programs\python\python311\lib\site-packages (
from awsebcli) (1.31.85)
Requirement already satisfied: cement==2.8.2 in c:\users\admin\appdata\local\programs\python\python311\lib\site-packages (from awsebc
li) (2.8.2)
Requirement already satisfied: colorama<0.4.4,>=0.2.5 in c:\users\admin\appdata\local\programs\python\python311\lib\site-packages (fr
om awsebcli) (0.4.3)
Requirement already satisfied: pathspec==0.10.1 in c:\users\admin\appdata\local\programs\python\python311\lib\site-packages (from aws
ebcli) (0.10.1)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in c:\users\admin\appdata\local\programs\python\python311\lib\site-package
```

This package allows me to connect to existing environments, as well as to create new ones (on the spot). I did need to Google a couple more commands, such as the `*eb deploy*` command to package up anything in your directory to be sent to the AWS EB server, there was also `*eb init*` which was used to first step ‘into’ the repository for that application!

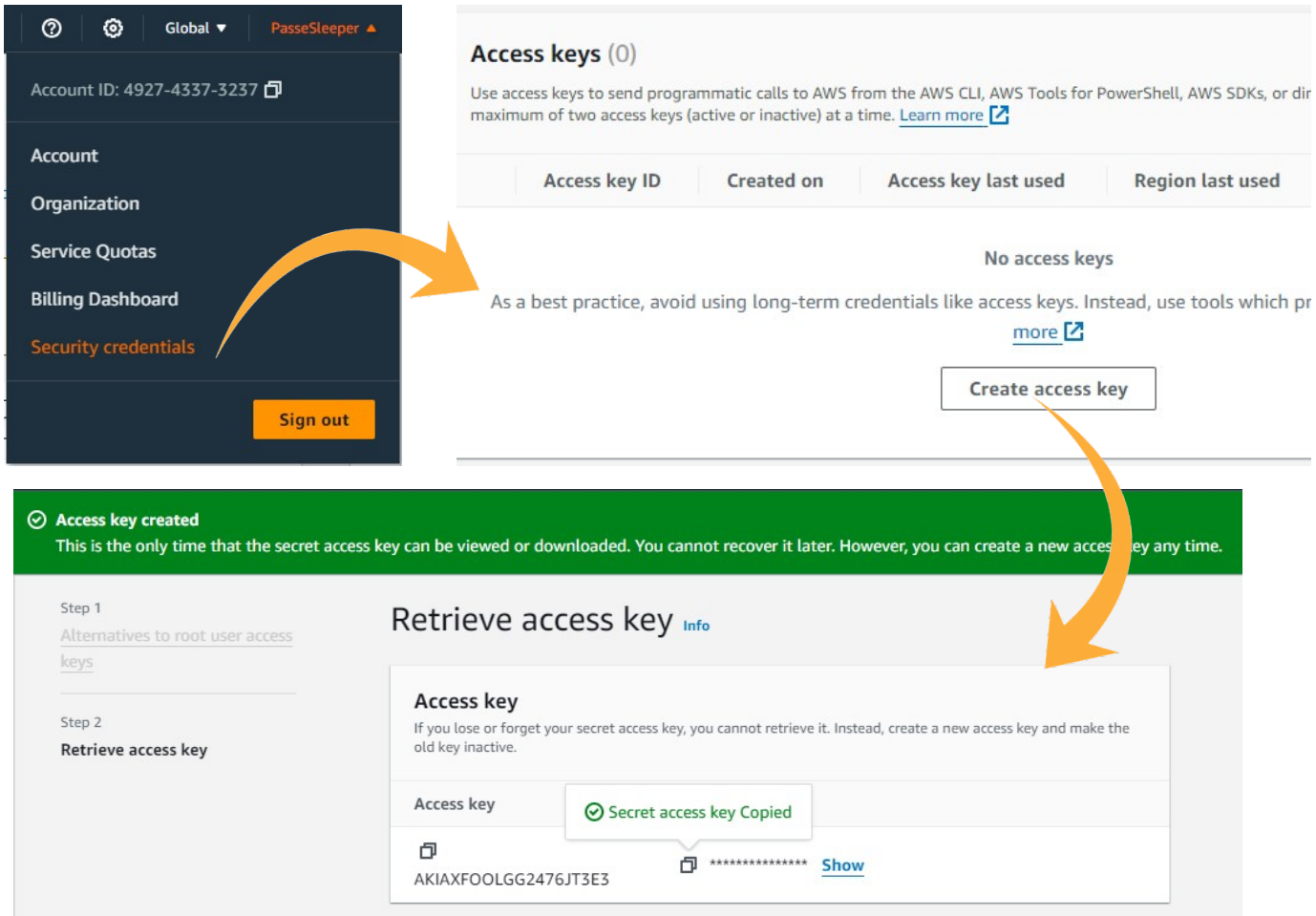
```
5) eu-central-1 : EU (Frankfurt)
6) ap-south-1 : Asia Pacific (Mumbai)
7) ap-southeast-1 : Asia Pacific (Singapore)
8) ap-southeast-2 : Asia Pacific (Sydney)
9) ap-northeast-1 : Asia Pacific (Tokyo)
10) ap-northeast-2 : Asia Pacific (Seoul)
11) sa-east-1 : South America (Sao Paulo)
12) cn-north-1 : China (Beijing)
13) cn-northwest-1 : China (Ningxia)
14) us-east-2 : US East (Ohio)
15) ca-central-1 : Canada (Central)
16) eu-west-2 : EU (London)
17) eu-west-3 : EU (Paris)
18) eu-north-1 : EU (Stockholm)
19) eu-south-1 : EU (Milano)
20) ap-east-1 : Asia Pacific (Hong Kong)
21) me-south-1 : Middle East (Bahrain)
22) il-central-1 : Middle East (Israel)
23) af-south-1 : Africa (Cape Town)
24) ap-southeast-3 : Asia Pacific (Jakarta)
25) ap-northeast-3 : Asia Pacific (Osaka)
(default is 3): 4
```

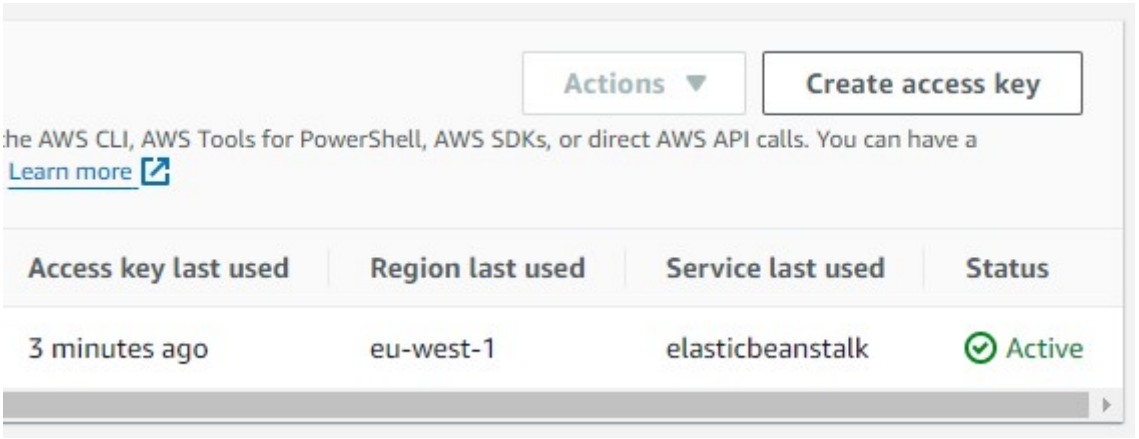
In this screenshot, you can see the list of server locations you can go to & I chose Number 4 as that was eu-west-1 (Ireland).

After connecting, you were required to input an aws-access-id & aws-secret-key.

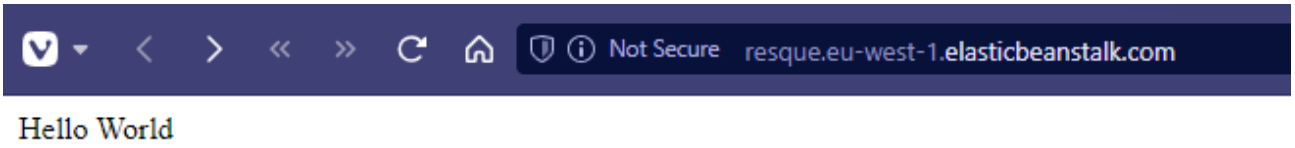
```
ERROR: The current user does not have the co ect permissions
ERROR: The current user does not have the co ect permissions
You have not yet set up your credentials  our credentials
You must provide your credentials.
(aws-access-id): AKIAXFOOLGG2476JT3E3
(aws-secret-key): n+5qgFz16OIE9VfjTc2OUCpotcRblQ/lxqWi9oTN

Select an application to use
1) Resque-collect
2) Gold Graph
3) FormSubmission
4) [ Create new Application ]
```

Those can be obtained by following the steps (can also be found here):

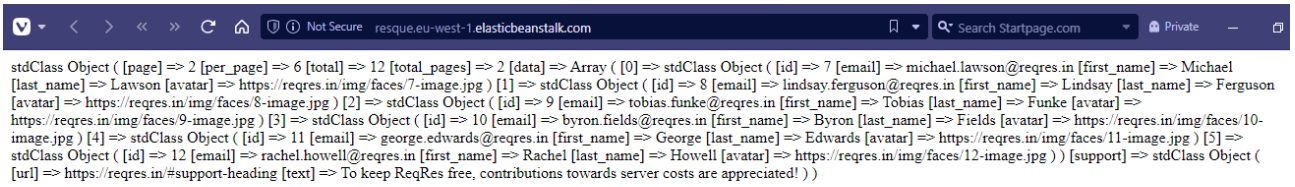


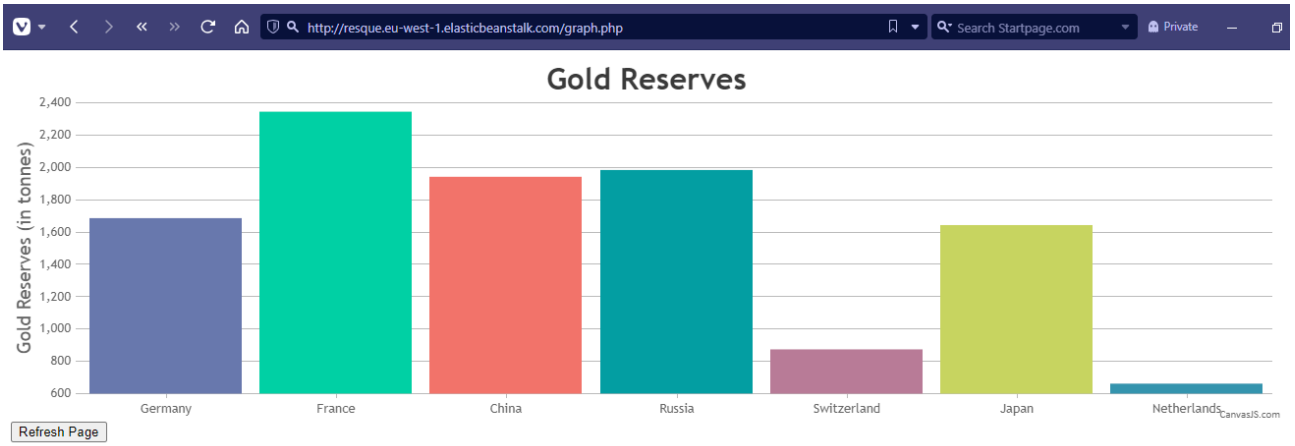


If it was created successfully, after using it for the 1st time, the last `Access key last used`should be visible!



I tried this out on the generic “Hello World” approach first, before plugging in my Gold Reserve & Resque PHP files. But just like that, everything worked successfully!





Task 2: Cloud Web Database

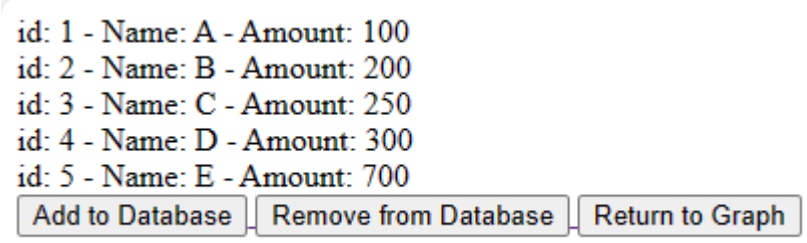
Data Reserves

To accomplish the task of connecting and making use of a database in my lab work, I decided to use my Gold Reserve website instead of the Resque form. Originally, the Resque form website was supposed to be a Form Submission page that upon completion gets automatically added to a Database, but In the limited time frame, I decided to build something simpler. I decided to repurpose my Gold Reserve website to allow a User to add & remove points from a graph, they should also be capable of naming the point.

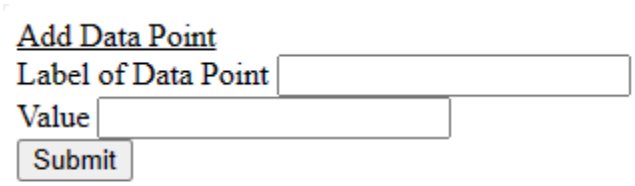
In order to accomplish this, I created more supplementary webpages for the Gold Reserve site.

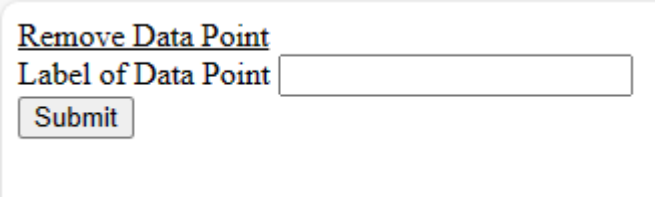
I had the Main Page.

I had the Database Page.

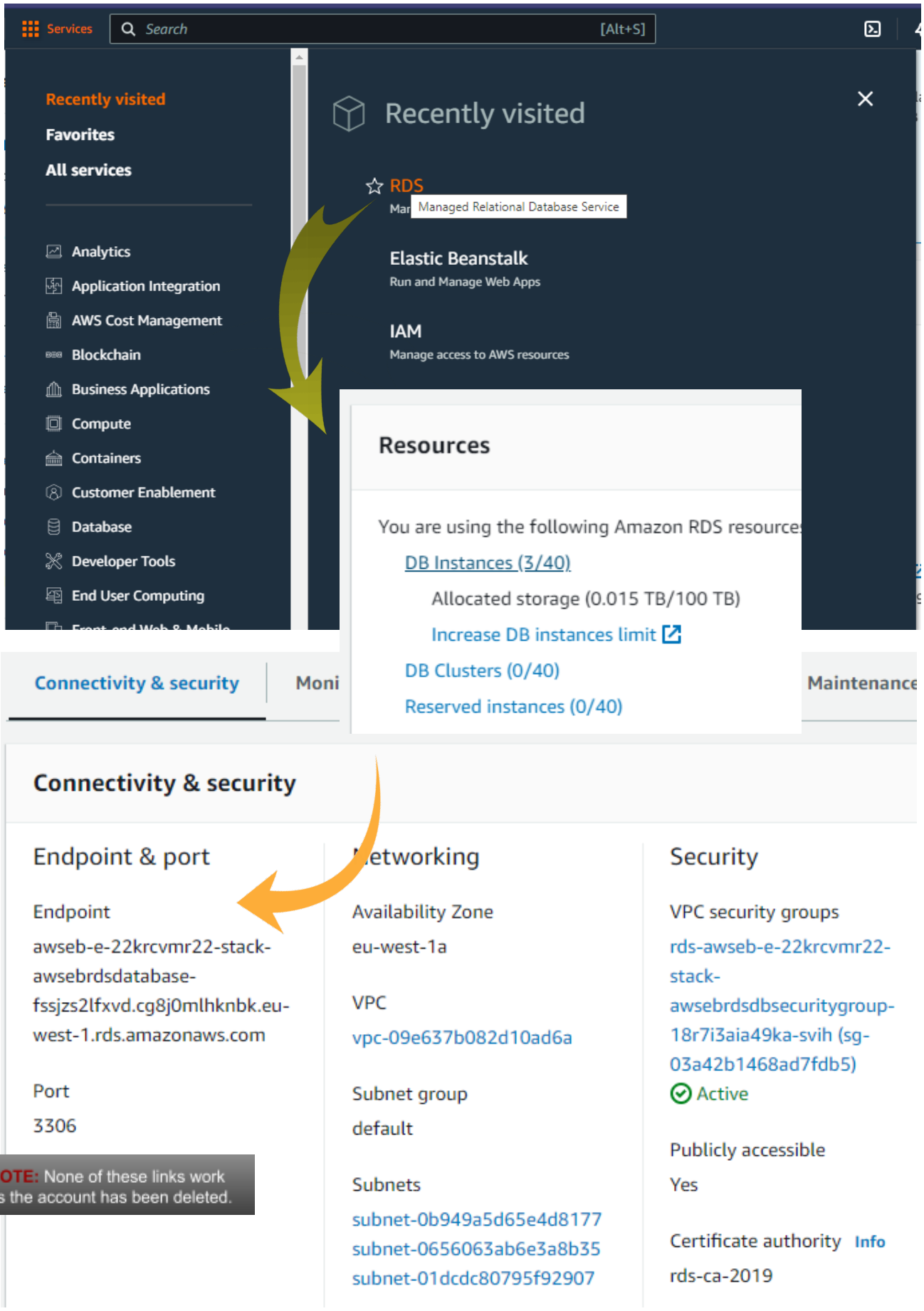


The Add Entry Page & Remove Entry Page.





I also set-up and connected to the Database using the previously created `admin` and `password`. To find the hostname, I had to go to RDS and find it there.



After collecting all the information, I was able to connect the AWS Database & display my website.

Conclusions & Experiences Gained

Throughout this Lab Assignment, I was given ample opportunity to work on new things while developing my cloud based web application.One such thing was the choice of programming language & I took the chance to learn a new one - PHP!

I am thankful for this module for allowing me this occasion. It was fun, sometimes frustrating, but a very unique experience. I loved that I could use the knowledge gained from the module to help me in understanding why I was doing what I was doing, and how AWS & Cloud Services in general help improve the lives of other people!

The most difficult parts of this assignment were definitely anything related to working on the AWS platform, as it does not provide enough material to begin right away, instead I was forced to research the proper means by which to upload my files (when using PHP, the default file is ALWAYS index.php), how to actually connect with AWS Database (very complicated for a newbie!).

I can’t wait to begin using this type of code in my future, I might even build a website of my own someday!