University Of Technology Sydney – 4 June 2023

Amazon Web Service for LAMP stack

41001 | Cloud Computing and Software as a Service

Andrew Soen

andrew.soen-1@student.uts.edu.au

http://thestartupapp-env.eba-djqyrk2j.us-east-1.elasticbeanstalk.com/

1. Introduction

This document outlines the proposed system for a small startup company leveraging Amazon Web Services (AWS). It includes the requirements for the analysis and specifications provided by the startup, as also assumptions could be made during the process of the system designing. AWS for LAMP Stack offers a comprehensive suite of cloud services specifically designed to streamline the web development and hosting process for small startups. The startup's concerns revolve around scalable infrastructure to accommodate uncertain demand and disaster recovery for high performance. The objective is to design and implement a scalable, elastic, highly available, and fault-tolerant architecture that facilitates organic growth. Leveraging AWS's capabilities, the proposed solution will address the startup's requirements, optimize costs, and ensure continuity. Together, we will develop a tailored infrastructure that aligns with their vision, setting them on a path to success in the dynamic business landscape. By leveraging the scalability and reliability of AWS, businesses can focus on their core competencies while leaving the infrastructure management to the experts. The AWS services selected to fulfill all the requirements needed for requirement to be meet, and CloudFormation's auto template will generate an architecture diagram.

2. Assumptions and Constraints

As the startup's requirement is provided and gives lots of description, some assumptions and constraints can be seen and need to think about to give better service as great as possible as what is show below are the assumptions made during design.

Assu 1. Startup's application was compatible with AWS environment and can be easily be modified from the lamp stack that exist before.

Assu 2. Their environment is running the recent version of PHP language.

Assu 3. Startup does not need to have PHP language with storage of 512 MB or larger.

Assu 4. The notification of email is supposed to be sent to the environment developer.

Assu 5. It requires auto scaling as it was based on the output of the connectivity to ensures that one instances is created or removed during the time necessary

Assu 6. The implementation of the classic Load Balancer can be used and implemented in the environment.

Assu 7. Startup's MySQL database does not need to have 5 GB or more worth of storage to start up the database.

Assu 8. Database and web server needed Security group of HTTP and SSH for it to run.

There are certain constraints that should be taken into account throughout the building of the environment, as specified in the assignment deliverables and description. The following constraints have been taken into account.

Cons 1. When it starts up, RDS could use up to 5 GB worth of storage to start up their database

Cons 2. All instances must use the same custom key pairs.

Cons 3. All of the Subnets need to be public and have at least two of them with the availability zone.

Cons 4. The request of HTTP and SSH connectivity can be triggered by any ip since it's 0.0.0.0/0

Cons 5. Auto scaling must have a minimum of 2 instances and maximum of 8 instances.

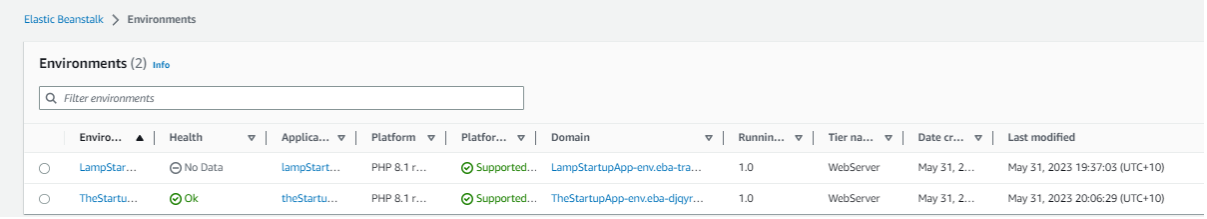
Cons 6. Since it uses elastic beanstalk, email notification is crucial for important events in the environment.

3. AWS Services

Although these can be expensive to operate over time, Amazon Web Services offers a wide selection of services that enable developers to build anything from a basic web server to an incredibly complete web server that can handle a lot. A variety of servicers was employed in order to finish this starting web server and give the new business a practical and efficient solution, which is include services such as:

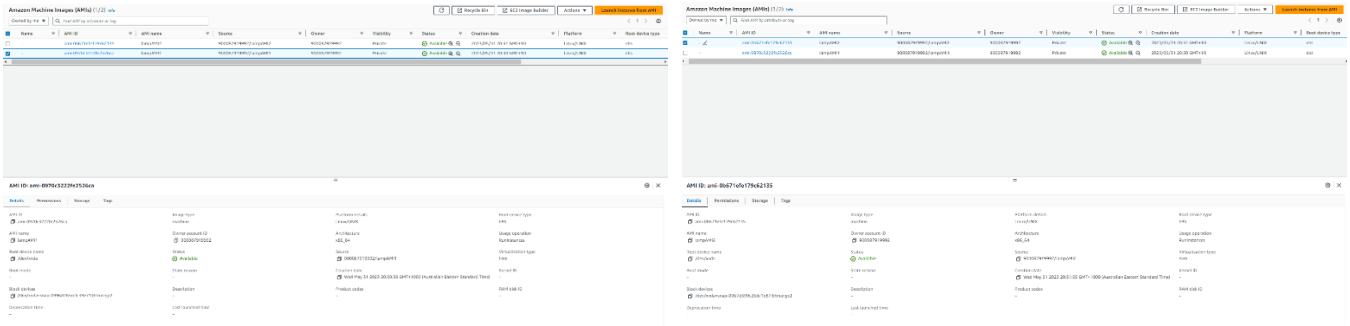
a. **AWS Beanstalk** It uses AWS Beanstalk to create applications uses all the service shown on the picture as it was developed by an application. This way makes it easy and simple to be use in manage

all the services in one application *(image of the application environment shown below and the web server on the appendix a)*

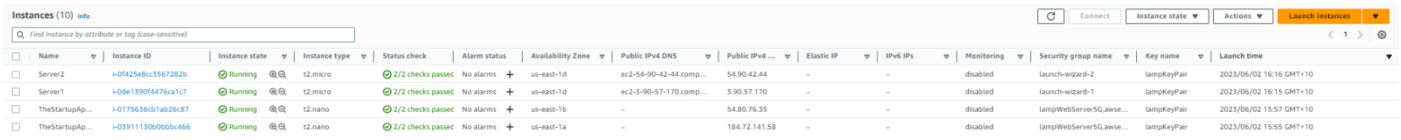


b. **Amazon EC2** This service allows for the development of a web server which when used with Beanstalk integration with RDS and other services is easy. This was a vital service required in completing the assignment allowing me to work with instances and create a Custom AMI

c. **Custom AMI (Amazon Machine Image).** A custom AMI was generated specifically for the Beanstalk environment, utilizing Amazon's documentation. This involved substituting the automatically generated AMI ID with the custom AMI ID, resulting in an updated environment.

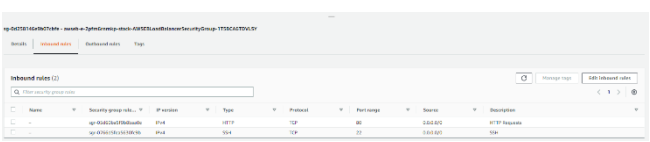


The Instance was used with Amazon Machine Image

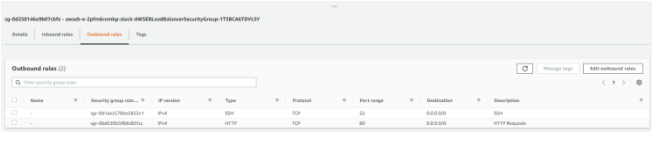


d. **Custom Security groups allowing HTTP and SSH requests.** To facilitate HTTP and SSH requests, custom security groups were implemented. Inbound rules were established for both protocols, functioning as a firewall for instances. This concept is similar to configuring advanced settings in the Windows Defender Firewall on a Windows 11 system. (Shown in the appendix).

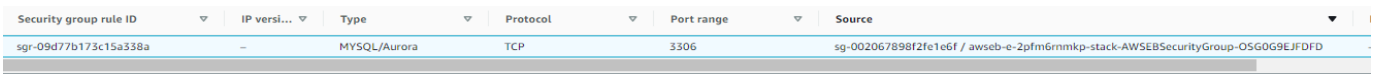
Here are the Inbound rules:



Here are the Outbound rules:

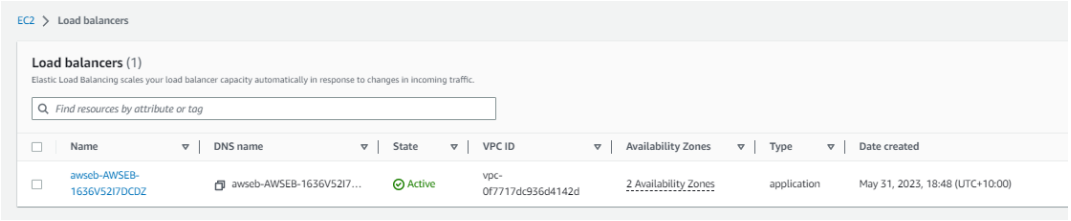


A security group for the MySQL database was also developed to allow requests made through the web server.



e. **Load Balancer** Application Load Balancer was integrated into the system during the transition from a single-instance setup to a load-balanced environment. This load balancer plays a critical role in distributing the workload across instances, ensuring efficient resource

accessibility.



f. **Auto Scaling**. Auto Scaling was implemented with a minimum of two instances and a maximum of eight instances. Scaling triggers were set based on network output traffic, with upper and lower thresholds of 60% and 30% respectively. The use of On-Demand instances (t2.micro) and a scaling cooldown of 360 seconds were employed to dynamically adjust resource availability based on traffic demand

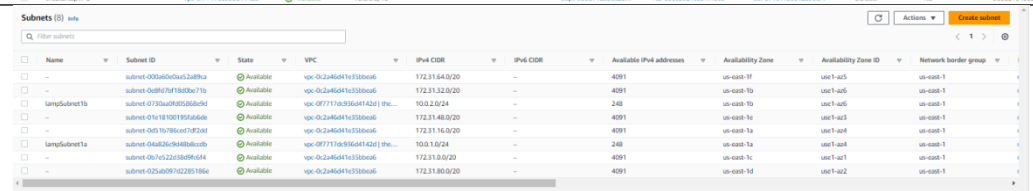
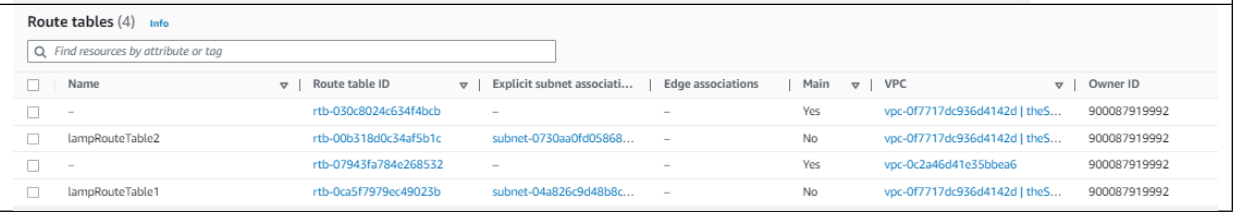
Environment type Load balanced	Min instances 2	Max instances 8
Fleet composition On-Demand instances	On-demand base 0	On-demand above base 70
Capacity rebalancing Deactivated	Processor type x86_64	Instance types t2.nano,t2.micro
AMI ID ami-0ca42562f43556e13	Availability Zones Any	Metric NetworkOut
Statistic Average	Unit Percent	Period 5
Breach duration 5	Upper threshold 60	Scale up increment 1
Lower threshold 30	Scale down increment -1	

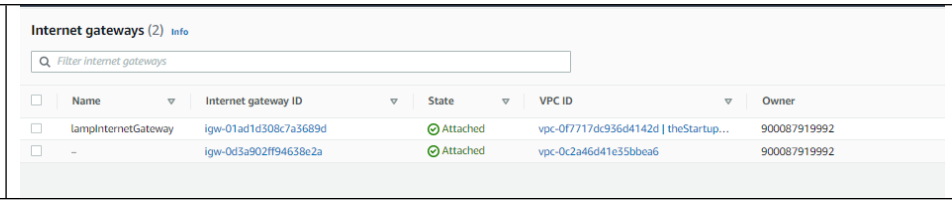
g. **RDS.** A high availability MySQL RDS was deployed to complement the existing LAMP stack. Placeholder credentials and a small storage space of 5 GB were provisioned initially. **The** Placeholder credentials with the Username of `***lampUsername***` and Password of `***lampPassword***` have been added to database.

Database

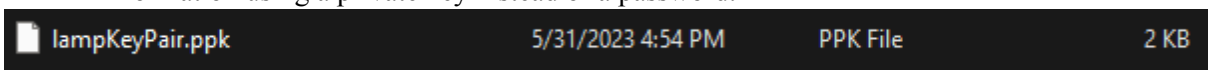
Database availability true	Has coupled database true	Database deletion policy Delete
Database engine mysql	Database engine version 8.0.32	Database instance class db.t2.micro
Database password ******	Database storage 5	Database subnets subnet-0730aa0fd05868e9d,subnet-04a826c9d48b8ccdb
Database username lampUsername		

h. **Custom Virtual Private Cloud (VPC)** The creation of a custom VPC with two subnets in different availability zones (us-east-1a and us-east-1b) facilitated a shared resource environment between instances as it was vital. A route table and an internet gateway were established to enable communication within this VPC.

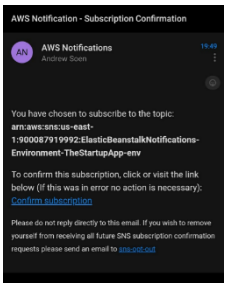
VPC	
Subnets	
Route Table	

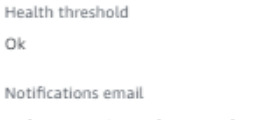
Internet Gateway	

i. **Key Pairs.** The instances utilized a custom key pair, named 'lampKeyPair', which encrypts login information using a private key instead of a password.

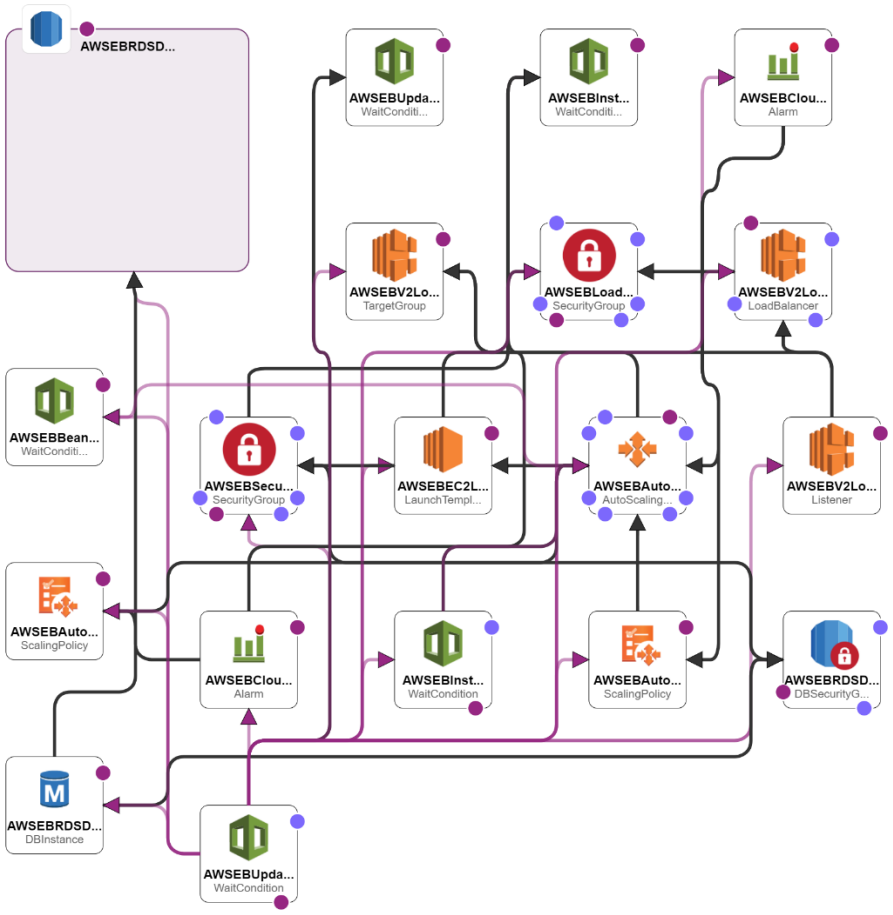


j. **Set email notifications for important events in your environment** Important events within the environment, particularly if using Elastic Beanstalk, trigger email notifications. This feature keeps developers informed of server status changes without requiring constant monitoring of the system health screen.





4. Architecture Diagram



5. Conclusion

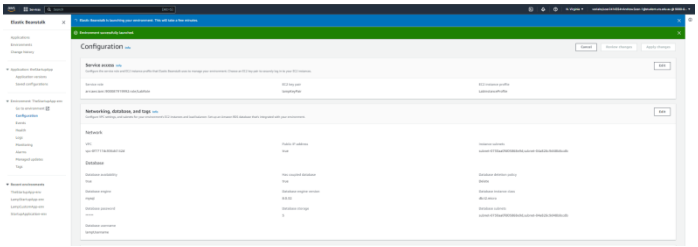
Overall, moving the startup's LAMP stack to AWS gives it a scalable and adaptable infrastructure. AWS compatibility and the most recent version of PHP make it simple for startups to adapt their applications for cloud environments. To achieve scalability, high availability, and fault tolerance, the architectural design combines services like EC2 instances, RDS for MySQL, Elastic Beanstalk, Auto Scaling, and the Classic Load Balancer. In order to provide optimal resource utilization, restrictions on storage, security groups, subnets, and instance limitations are taken into account. Overall, the firm is better able to manage its application effectively, support future expansion, and make use of cloud computing features thanks to the shift to AWS.

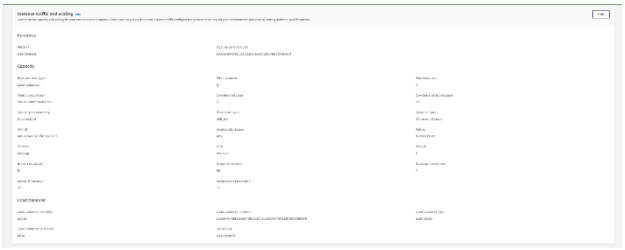
6. Appendix and Resources

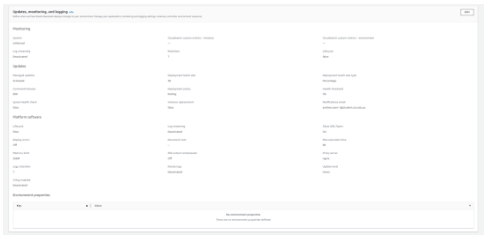
Appendix

a. Appendix 1: Web Server

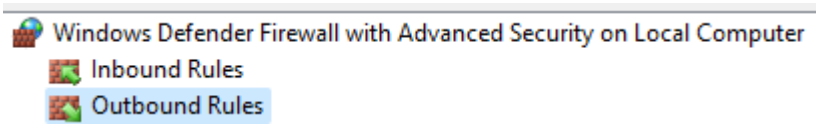
b. Appendix 2: Configuration Screen







c. Appendix 3: Defender Firewall Advanced Security settings Windows 11



Resource:

- Lab and contents regarding AWS
- Amazon Web Services. (n.d.). Elastic Beanstalk: Getting Started. Retrieved from https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/GettingStarted.html
- Amazon Web Services. (n.d.). Tutorial: Set up a scaled and load-balanced application. Retrieve from https://docs.aws.amazon.com/autoscaling/ec2/userguide/as-register-lbs-with-asg.html#as-register-lbs-console
- Amazon Web Services. (n.d.). What is Amazon VPC?. Retrieve from https://docs.aws.amazon.com/vpc/latest/userguide/what-is-amazon-vpc.html