**B8IT121**

***WordPress application hosted in AWS load balancer and auto scaling***

***Web and Cloud Application Development***

Document Version No. v1.0

Presented by:

**Daniel Titello**

Table of Contents

[Overview 2](#_Toc41999022)

[In Scope: 3](#_Toc41999023)

[Architecture 8](#_Toc41999024)

[Costs Plan 9](#_Toc41999025)

[Conclusion 10](#_Toc41999026)

[Bibliography / References: 10](#_Toc41999027)

## 

## Overview

The purpose of this project is to test the extent to which I have achieved a specific learning objective of the course. I have been commissioned to design, build and deploy a WordPress application, using a cloud service provider, taking into account all the learning objectives and minimum requirements for the project.

## In Scope:

|  |  |
| --- | --- |
| Objective | Critical Success Factor |
| * Student’s union | Deploy a fully functional WordPress application. |
| * User Management | Configure user management for the website using WorPress module. |
| * Autoscaling and load balancing | Run the full architecture behind a load balancer and make sure to use an auto scaler to. |
| * Database | Use AWS services to implement the database |

Environment Information:

**Virtual Private Cloud (VPC):**

The first step towards the goal is to create and Virtual Private Cloud that will allow me to logically isolated section of AWS cloud to launch the necessary resources in a virtual network. By default, the VPC will create a private and public subnet. The public subnet has access to the internet, and I will use it to install the load balancer. The private subnet will host the ec2 instances and database instances providing a layer of security that can’t be access via internet. Each subnet provides about 490 usable IP addresses for the instance.



One of the requirements is to build a web application with high availability able to handle a failure of an availability zone. To do that, we will use one public subnet and two privates (Web app + DB) for each available zone and two zones to guarantee that the application will be still running in case of another’s failure.

**A screenshot of a cell phone

Description automatically generated**

**RDS:**

I create an RDS instance as the main database. I need to launch this instance before building my container because I will have to use this endpoint (host) later on.

**Virtual Machines:**

As the main scope of the project was to develop a web application, the focus was on developing a working application using AWS resources with docker. Initially, I launched an EC2 instance in a public subnet with the security group that allows me to access the instance via ssh port 22. I need to connect to the ec2 to install docker, docker compose and create a docker-compose.yml file to build a WordPress container.

I don’t need to use mysql services, only wordpress because the database is configured using AWS RDS.

A screenshot of a social media post

Description automatically generated

To install docker and docker compose I used the official documentation via website <https://docs.docker.com/compose/install/>. The installation can bring a bit of trouble if you don’t understand what each command does, for example, it’s important to have docker compose running and you try to build the WordPress container but depending of the tutorial you follow, you don’t check that. When the ec2 instance restarts, you need to run a command to start docker or you simply run a command to configure docker to start every time an ec2 instance restarts.

Command: *sudo chkconfig docker on*

It’s very important to use elastic IP when you are configuring your public VM because it allows you to use the same IP address which is very important when configuring your Wordpress container.

A screenshot of a cell phone

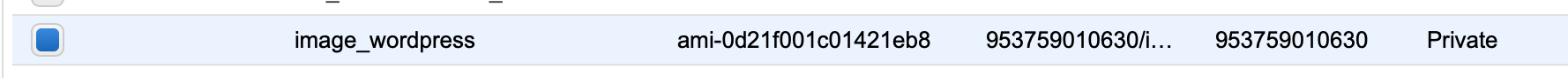
Description automatically generated

A screenshot of a cell phone

Description automatically generated

My WordPress container will be access at port 8080. We need to configure this port in the security group inbound ports.

Now, I have an ec2 instance with wordpress and mysql database (in an RDS instance) running. The next step is to create an AMI image that will be used later by the auto scaler. The new ec2 instances will be launched in a private subnet later without access which means we need to configure everything before setting load balancer and auto scaler.



**Auto Scaling:**

Auto scaling allows me to automatically launch ec2 instances when the traffic in my web application increases or decreases. Also, I can set the number of instances I will be using. For this project I created an auto scaler that initially is using 2 ec2 instances and I can increase by one if the traffic goes over 60% and I can decrease by one instance if traffic goes down 20%. Auto scaling is using my AMI image\_wordpress to create the new instances.

A screenshot of a cell phone

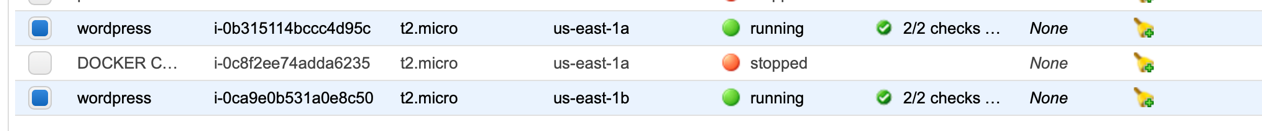
Description automatically generated

As you can see below, I am using two availability zones with minimum of 2 instances (one per zone) and maximum of 6 instances.

A screenshot of a social media post

Description automatically generated

Auto scaler launched my two ec2 instances right after I configured auto scaling group.



**Load Balancer:**

A load balancer distributes incoming application traffic across multiple EC2 instances in multiple Availability Zones. This increases the fault tolerance of your applications. Elastic Load Balancing detects unhealthy instances and routes traffic only to healthy instances.

Your load balancer serves as a single point of contact for clients. This increases the availability of your application.

I set the load balancer using the class tutorial. The DNS (elb-1221065641.us-east-1.elb.amazonaws.com: port) is used to access the web application so I had to update my database site url and home for my wordpress. The DNS name was given by AWS.

A screenshot of a social media post

Description automatically generated

By default, AWS uses port 80 for HTTP application, but I am using port 8080 as mentioned before. Listeners are responsible for calling the right target group that will select the proper ec2 instances. In my case, I created a listener for the port 8080 forwarding to target group elb8080.

A screenshot of a cell phone

Description automatically generated

Current URL:

<http://elb-1221065641.us-east-1.elb.amazonaws.com:8080/>

WEB Application

My web application is created in Wordpress with the theme Astra and a few plugins such as editor text Gutenberg and forms creator WP-forms. The HOME page contains a bit of information about the DBS student union. The news page is a blog page with a few posts. The About us page provides more information about DBS student union and Contact us page contains a form that anyone can send queries to the admin.

A group of people looking at a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

A computer sitting on top of a desk

Description automatically generated

A screenshot of a cell phone

Description automatically generated

## Architecture

**A screenshot of a social media post

Description automatically generated**

Lesson’s Learned

**Database:**

I had a few issues with database connection before using elastic IP because the ec2 instance ip was constantly changing every time I initiated the instance.

**HTTP code problems:**

**HTTP 502: Bad Gateway**

I was facing this issue when the ec2 instances registered in my target group weren’t healthy which means the load balancer was forwarding the work to the target group that didn’t have any healthy ec2 instance registered. Those instances weren’t able to connect to the database initially causing the problem.

**HTTP 503: Service Unavailable**

This issue happened when I was trying to connect to the load balancer DNS but my ec2 instances were not available. Once the instances were up, the error went away.

**HTTP 504: Gateway Timeout**

Target groups have healthy checks. Only healthy ec2 instances will have work forwarded to them. To sort this issue, I increased the timeout time to 20 seconds instead of 5.

**WordPress performance**

My wordpress application works fine but I am facing a lot of performance issues related to slowness and the installation of plugins.

## Costs Plan

**A screenshot of a social media post

Description automatically generated**

## Conclusion

From review of the scope and critical success factor, this project has been marked as successful. All servers and services were accessible and configurable.

|  |  |
| --- | --- |
| Objective | Critical Success Factor |
| * Student’s union | Deploy a fully functional WordPress application. **(WORKING)** |
| * User Management | Configure user management for the website using WorPress module. **(WORKING)** |
| * Autoscaling and load balancing | Run the full architecture behind a load balancer and make sure to use an auto scaler to. **(WORKING)** |
| * Database | Use AWS services to implement the database  **(WORKING)** |

## Bibliography / References:

* Docs.aws.amazon.com. 2020. Troubleshoot A Classic Load Balancer: HTTP Errors - Elastic Load Balancing. [online] Available at: <https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/ts-elb-error-message.html#ts-elb-errorcodes-http504> [Accessed 1 June 2020].
* Docs.aws.amazon.com. 2020. Configure The Idle Connection Timeout For Your Classic Load Balancer - Elastic Load Balancing. [online] Available at: <https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/config-idle-timeout.html> [Accessed 1 June 2020].
* Docker Documentation. 2020. Install Docker Compose. [online] Available at: <https://docs.docker.com/compose/install/> [Accessed 1 June 2020].
* UpCloud. 2020. How To Deploy Wordpress With Docker Compose. [online] Available at: <https://upcloud.com/community/tutorials/deploy-wordpress-with-docker-compose/> [Accessed 1 June 2020].
* Nickvanhoof.com. 2020. Docker Compose & Wordpress – Codingexpedition. [online] Available at: <https://nickvanhoof.com/docker-compose-wordpress/> [Accessed 1 June 2020].
* Docs.aws.amazon.com. 2020. What Is Amazon EC2? - Amazon Elastic Compute Cloud. [online] Available at: <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html> [Accessed 1 June 2020].