**Deploy & Automation High-Availability Web Application Using**

**Amazon Web Services**

By

Md. Emran Hosen

ID: 21160200445

Jobair Hosain Bhuiyan

ID: 21160200446

Ashadujjaman Hiru

ID: 21160200463

A project submitted in partial fulfillment of the requirements for the degree of Bachelor of Science in computer science

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NORTHERN UNIVERSITY BANGLADESH

**APPROVAL**

The Project Report “**Deploy & Automation High-Availability web application Using Amazon web services**” submitted by Md. Emran Hosen ID: 21160200445, Jobair Hosain Bhuiyan ID: 21160200446, Ashadujjaman Hiru ID: 21160200463 to the Department of Computer Science and Engineering, Northern University Bangladesh, has been accepted as satisfactory for the partial fulfillment of the requirement for the degree of Bachelor of Science in Computer Science and Engineering and approved as to its style and contents.

Board of Examiners

1.

2.

3.

|  |  |
| --- | --- |
| ------------------------------------- |  |
| Assistant Professor & Head  [Md. Raihan-ul-Masood](https://nub.ac.bd/pims/faculty-member/dfoxgpw6/md.-raihan-ul-masood)  Department of Computer Science and Engineering  Northern University Bangladesh |  |

**DECLARATION**

We, hereby, declare that the work presented in the Project is the outcome of the investigation performed by us under the supervision of [Md. Raihan-ul-Masood](https://nub.ac.bd/pims/faculty-member/dfoxgpw6/md.-raihan-ul-masood), Assistant Professor & Head, Department of Computer Science and Engineering, Northern University Bangladesh. We also declare that no part of this project has been submitted elsewhere for the award of any degree.

|  |  |
| --- | --- |
| Countersigned | Signature |
| ----------------------- | ----------------------- |
| ([Md. Raihan-ul-Masood](https://nub.ac.bd/pims/faculty-member/dfoxgpw6/md.-raihan-ul-masood)) | (Md. Emran Hosen)  --------------------------- |
| **Supervisor** | (Jobair Hossain Bhuiyan)  ------------------------------ |
|  | (AshaduzzamanHiru) |
|  | **Candidates** |

**ABSTRACT**

In this project, we deploy a high-availability LAMP stack e-commerce web application using AWS Platform and Amazon Relational Database Service (RDS). The stack uses Linux, Apache, MySQL, and PHP. Application will automatically handle the security, deployment, capacity provisioning, load balancing, auto-scaling to monitor and continues integration. Servers will automatically scale up and down based on application's specific need using easily adjustable Auto Scaling settings. We use MySQL on Amazon RDS to set up, operate and scale a relational database in the cloud. For automatic code deploy and integration we use AWS CodeDeploy and AWS CodePipeline which automatically sync all the commit from Github.

|  |  |
| --- | --- |
| **TABLE OF CONTENT** | Page |
|  |  |
|  |  |
| **APPROVAL** | 2 |
| **DECLARATION** | 3 |
| **ABSTRACT** | 4 |
| **TABLE OF FIGURES** | 6 |
|  |  |
| **Chapter I : Introduction** | 7 |
| 1.1 Overview | 8 |
| 1.2  Objective of the Project | 8 |
|  |  |
| **Chapter II : Project Goal and Requirement** | 9 |
| 2.1 Goals of the Project | 10 |
| 2.2 Security | 10 |
| 2.3 Flexible and Customizable | 10 |
|  |  |
| **Chapter III : Overall system design and Implementation** | 11 |
| 3.1 Architecture of the system | 12 |
| 3.2 Web application info. | 12 |
| 3.3 Stack & Architecture overview | 12 |
| 3.4 Design Principles for this Project | 13 |
| 3.5 Networking Setup | 13 |
| 3.6 MySQL Database on Amazon Relational Database Service (RDS) | 15 |
| 3.7 Why using RDS | 15 |
| 3.8 Auto Scaling | 16 |
| 3.9 Application Load Balancer | 17 |
| 3.10 Continuous deployment and Continuous Integration | 18 |
| 3.11 Route DNS Traffic for Our Domain to Our Website Bucket | 19 |
| 3.12 Test Our Website | 19 |
| **Chapter IV : Conclusion and Future Work** | 20 |
| 4.1 Facilities of our system | 21 |
| 4.2 Future plan of our system | 21 |
| 4.3 Conclusion | 21 |

|  |  |  |
| --- | --- | --- |
|  | **TABLE OF FIGURES** |  |
|  |  | Page |

|  |  |  |
| --- | --- | --- |
| Fig 1 | Architecture of System | 12 |
| Fig 2 | Virtual Private Cloud | 14 |
| Fig 3 | RDS Configuration | 15 |
| Fig 4 | Auto Scaling Group | 16 |
| Fig 5 | Scaling Policy | 16 |
| Fig 6 | Application Load Balancer Configuration | 17 |
| Fig 7 | Pipeline progress | 18 |
| Fig 8 | Ecommerce application on browser | 19 |

**Chapter I**

**Introduction**

* 1. **Overview**

Ecommerce sites has huge of user’s traffic and transitions that’s why it needs high availability. In this project we hosted an ecommerce application into Amazon Web Services. It offers a broad set of global cloud-based products including compute, storage, databases, analytics, networking, mobile, developer tools, management tools, IoT, security, and enterprise applications: on-demand, available in seconds, with pay-as-you-go pricing. From data warehousing to deployment tools, directories to content delivery, over 140 AWS services are available. New services can be provisioned quickly, without the upfront capital expense. This allows enterprises, start-ups, small and medium-sized businesses, and customers in the public sector to access the building blocks they need to respond quickly to changing business requirements. This whitepaper provides us with an overview of the beneﬁts of the AWS Cloud and introduces to the services that make up the platform.

* 1. **Objective of the Project**

The main objective of our project is to deploy a highly available and secured web server infrastructure including automation based on CDCI (Continues deployment and continues integration). Web applications are needs maximum uptime and data security in lower latency our approach is to make maximum uptime of web application including automation with no limits of network traffic.

**Chapter II**

**Project Goal and Requirement**

**2.1 Goals of the Project**

On premise server and datacenter has some limitations of power loss, internet connectivity issues, server hardware management or hardware failure, long time require for provisioning new servers and many more. That’s why we wanted a solution where we can provision servers at any time without any dependencies. A highly available, scalable, secured, isolated network, lower costly service for deploy an ecommerce application. And automatic code deploy and integration from Github using AWS CodeDeploy and AWS CodePipeline.

**2.2 Security**

* 24/7 access to data experts in case issues should arise
* Built-in firewall which allows for very specific access, from highly restrictive to public
* IAM services that track users access
* Multi-factor authentication and encrypted data storage capabilities
* Private cloud services are highly isolated from internet world

**2.3 Flexible and Customizable**

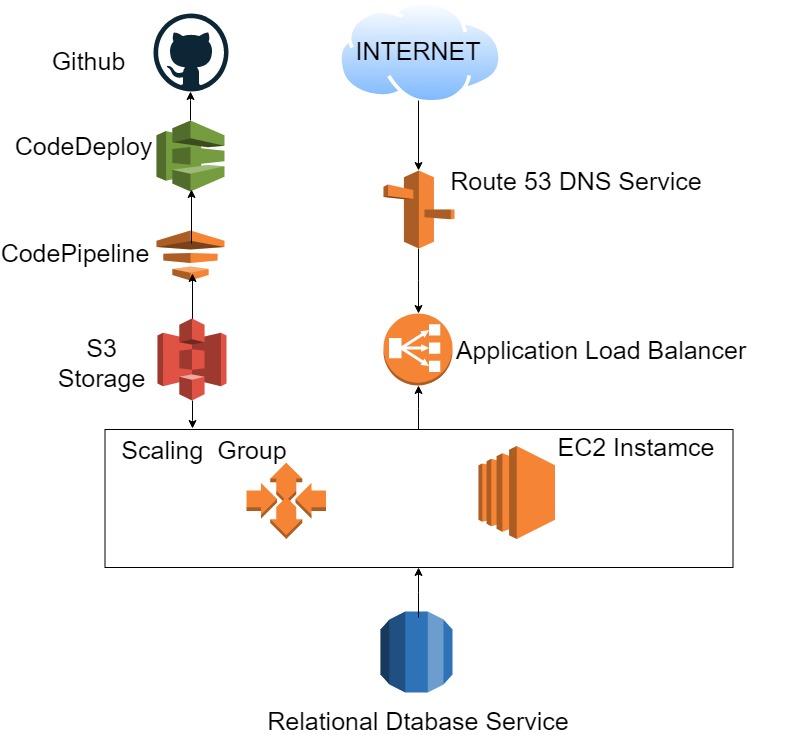
AWS lets us select the programming language, operating system, database, and other assets, so that we can create the solution that works best for our team. We’re not locked into an unfamiliar program that ends up costing our team time and money instead of freeing up resources to help continue to grow and support our business.

**Chapter III**

**Overall system design and Implementation**

**3.1 Architecture of the system**

Architecture designed based on five pillars operational excellence, security, reliability, performance efficiency, and cost optimization.



**Fig 1: Architecture of System**

# 3.2 Web application info

# Application that we are hosting on aws is an Ecommerce application, used to sell online products, mainly gym products. Application is constructed by php and react language.

# 3.3 Stack & Architecture overview

# We used MYSQL database on amazon relational database service and integrated with Ec2 Linux server to connection between application and database. For load balancing we used application load balancer and DNS service route 53. And we configured auto code integration to the server from Github repository by AWS CodePipeline, which allows us to sync every commit we do in the Github repository. None of the servers has internet accessibility so that servers are secured and isolated from internet.

# 3.4 Design Principles for this project

There are certain principles of architecture that one needs to follow to make the most of the tremendous capabilities of the Cloud.

## We must think Adaptive and Elastic

## We treated servers as disposable resources

## Implement loose coupling.

## Focused on services, not servers

## Database is the base of it all

## We must remove single points of failure

## Optimize for cost

## Caching

## AWS Cloud Architecture Security

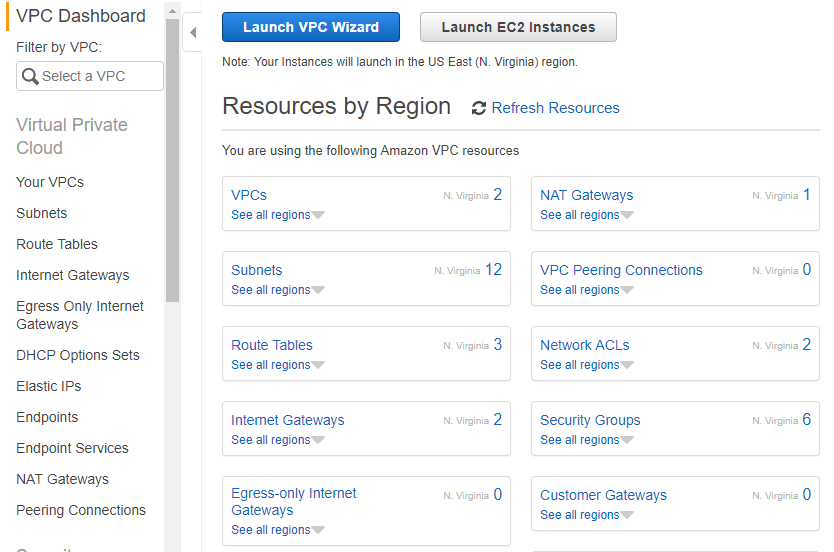
**3.5 Networking Setup**

Amazon Virtual Private Cloud (Amazon VPC) is the network backbone, it lets us provision a logically isolated section of the AWS Cloud where we can launch AWS resources in a virtual network that we define. We have complete control over our virtual networking environment, including selection of our own IP address range, creation of subnets, and configuration of route tables and network gateways.

The following table describes the inbound and outbound rules for the security group.

|  |  |  |  |
| --- | --- | --- | --- |
| **Inbound** | | | |
| **Source IP** | **Protocol** | **Port Range** | **Comments** |
| 0.0.0.0/0 | TCP | 80 | Allows inbound HTTP access from any IPv4 address. |
| 0.0.0.0/0 | TCP | 443 | Allows inbound HTTPS access from any IPv4 address. |
| Public IPv4 address range of we home network | TCP | 22 | Allows inbound SSH access from we home network to a Linux/UNIX instance. |
| Public IPv4 address range of we home network | TCP | 3389 | Allows inbound RDP access from we home network to a Windows instance. |
| **Outbound** | | | |
| **Destination IP** | **Protocol** | **Port Range** | **Comments** |
| 0.0.0.0/0 | All | All | The default outbound rule that allows all outbound IPv4 communication. |

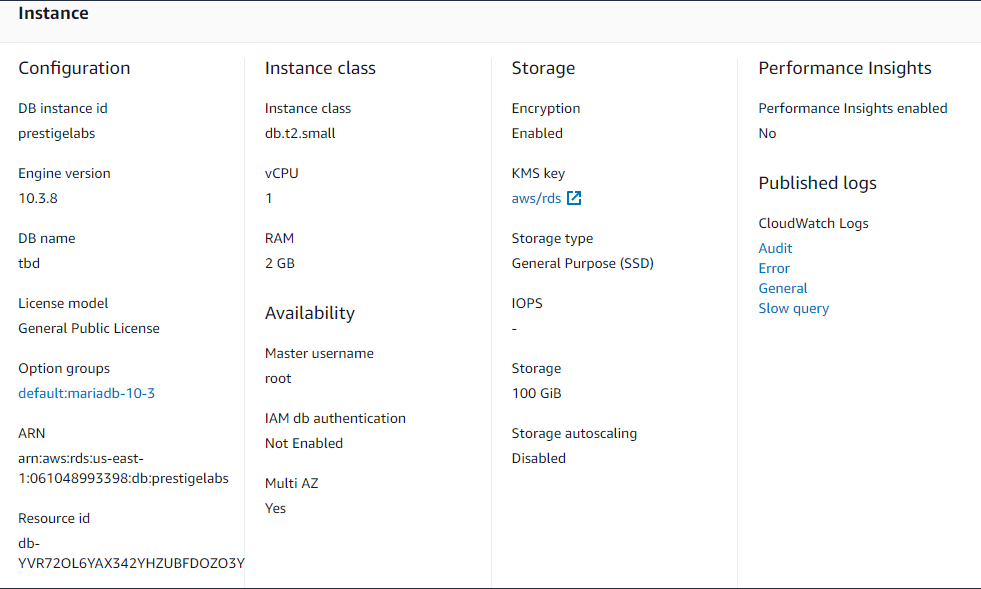
VPC network configuration is given below:



**Fig 2: Virtual** **Private Cloud**

# 3.6 MYSQL Database on Amazon Relational Database Service (RDS)

We are using MySQL Database on Amazon Relational Database Service (Amazon RDS). Provides cost-efficient and resizable capacity while automating time-consuming administration tasks such as hardware provisioning, database setup, patching and backups.



**Fig 2: RDS Configuration**

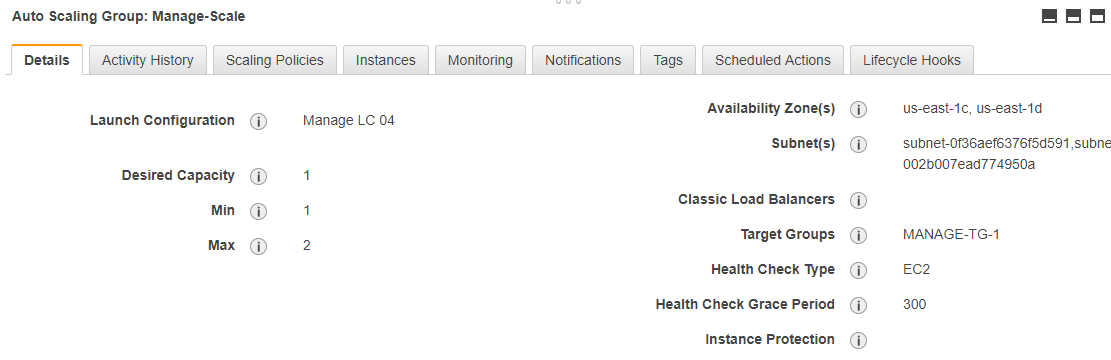
**3.7 Why using RDS**

* AWS RDS is a highly available relational database that offers a feature called Multi-AZ, If we use any other database service it has some limitations like data loss, backup, DNS record, single availability zone, no auto scaling.
* AWS also offers a domain name server (DNS) to access RDS, so even if the master database instance goes down, an RDS automatic failover mechanism will change the master DNS to a replica in order to achieve high availability.
* Database scalability can prove to be a real challenge if we try to scale our own, self-hosted database.

# 3.8 Auto Scaling

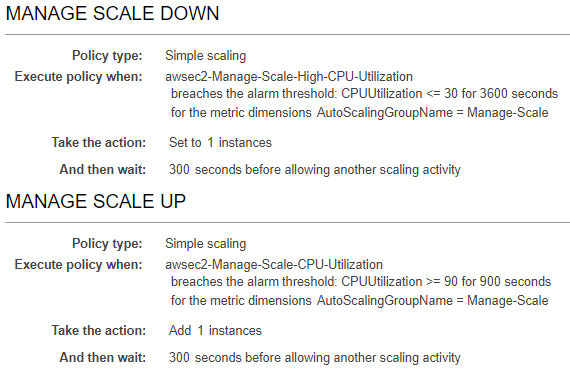
Adding Amazon EC2 Auto Scaling to application architecture is one way to maximize the benefits of the AWS Cloud.

* Better fault tolerance.
* Amazon EC2 Auto Scaling can detect when an instance is unhealthy, terminate it, and launch an instance to replace it.
* We also configure Amazon EC2 Auto Scaling to use multiple Availability Zones. If one Availability Zone becomes unavailable, Amazon EC2 Auto Scaling can launch instances in another one to compensate.

****

**Fig 4: Auto Scaling Group**

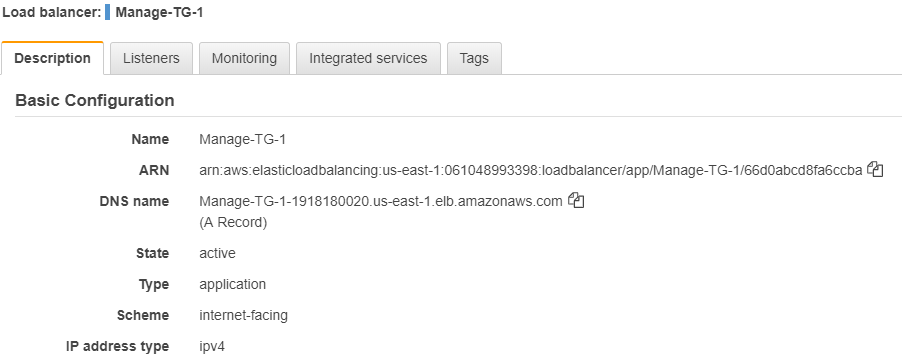
Below is the scaling policy that we used for Scaling Group:



**Fig 5: Scaling Policy**

**3.9 Application Load Balancer**

A load balancer serves as the single point of contact for clients. The load balancer distributes incoming application traffic across multiple targets, such as EC2 instances, in multiple Availability Zones. For Load balancing we need to configure a target group, and the target group contains the Ec2 server what we configured for web server and launch an application load balancer under the target group.

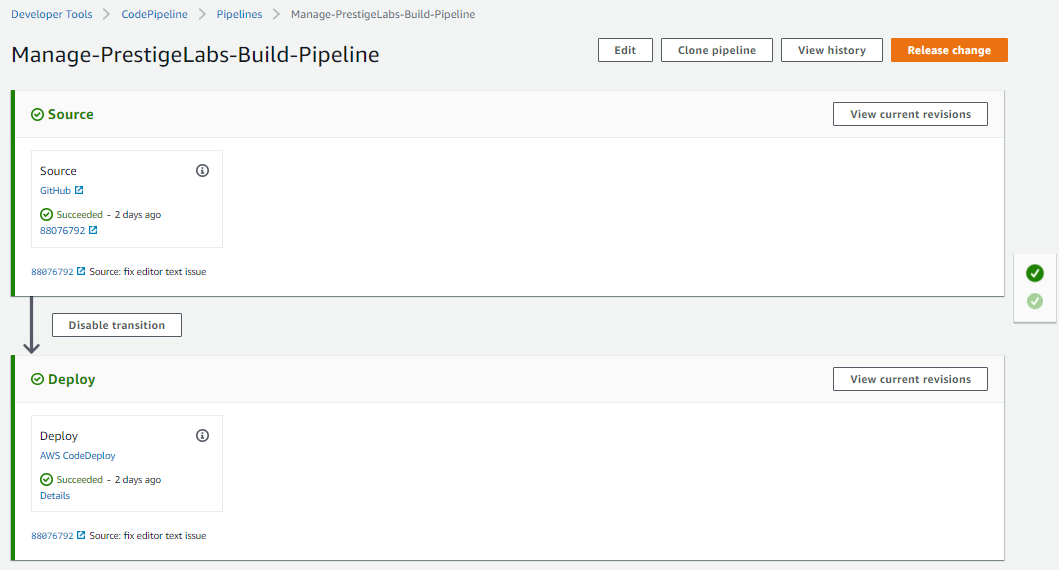


**Fig 6: Application Load Balancer Configuration**

**3.10 Continuous deployment and Continuous Integration**

CodePipeline and CodeDeploy are the continuous delivery service that enables we model, visualize and automate the steps required to release our application. Rather than using other tools like Jenkins AWS CodePipeline is better is so many way-

* Automates code deployments, allowing to deploy reliably and rapidly
* Maximize application availability by performing rolling updates across Amazon EC2 instances and tracking application health according to configurable rules
* Easily launch and track the status of your deployments through the AWS Management Console or the AWS CLI



**Fig 7: Pipeline progress**

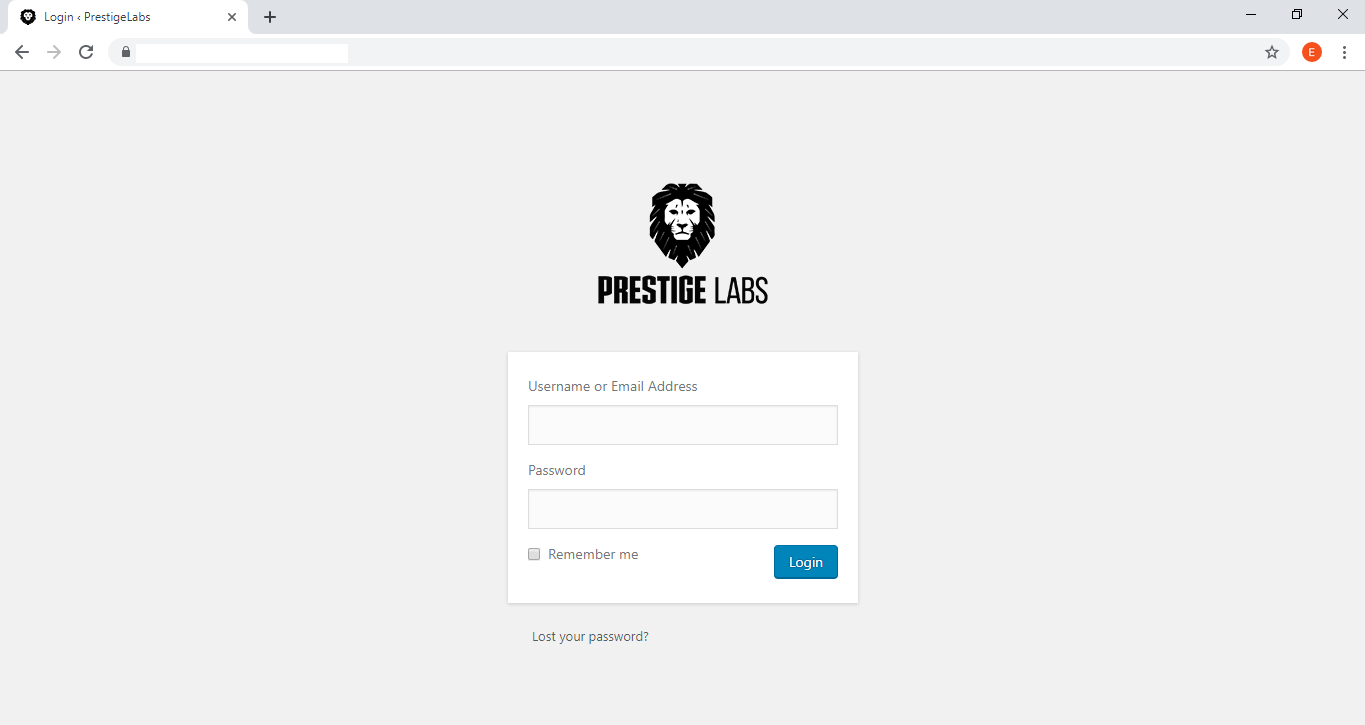
* 1. **Route DNS Traffic for Our Domain to Our Website Bucket**

Rather than using Godaddy & others, We pointed our server and load balancer to AWS route 53 DNS service. Because AWS use Anycast, and have DNS servers in 15+ locations worldwide. Their service seems well engineered for high uptime. Having our DNS resolve from 15+ locations worldwide makes your website a little bit faster for your end users. It also allows you to use a lower TTL, which means in case of a website failure, we can move our service over to a new IP faster.

**3.12 Test Our Website**

To verify that the website is working correctly, open a web browser and browse to the following URLs:

* http://domain-name – Displays the index document in the domain-name bucket
* http://www.domain-name – Redirects our request to the domain-name bucket

****

**Fig 8: Ecommerce application on browser**

We get the application on browser, we successfully deployed the application.

**Chapter IV**

**Conclusion and Future work**

**4.1 Facilities of our system**

Facilities of the system given below:

* Keep Data Safe
* Automatic Code Deploy and Delivery
* Meet Compliance Requirements
* Save Money
* Scale Quickly
* Highly available and less downtime

**4.2 Future plan of our system**

Future plan of the system given below:

* Server less Infrastructure management
* We will deploy Infrastructure as code

**4.3 Conclusion**

In this system we have used several tools to make this project efficient one. Currently our system is playing a very important role especially for DevOps professional and AWS administrator. Student of NUB who is interested in DevOps or AWS administration will be benefited by this book. We have faced many problems during implementing this project, these problems have been solved by visiting different websites and many tutorials based AWS. This project also saves AWS students valuable time.