**ABSTRACT**

This project main aim is to handle thousands of requests (traffic load) on a web portal, when millions of users want to access the same webpage. When the user hits on a certain URL and if the requests are more on that URL the traffic load will be more. There will be lagging of the site and can’t be accessed by all the users at a time, to avoid this problem we are going to change the existing policies in AWS Cloud, and create virtual instance servers by using AWS.

The main objective of this project is to maintain auto scaling and load balancing on a certain web portal. In Load Balancers, Elastic Load Balancing automatically distributes your incoming traffic across multiple targets, such as EC2 instances. Auto scaling monitors your applications and automatically adjusts capacity to maintain steady and better performance at the lowest cost. Network security consists of the policies and practices to prevent and monitor unauthorized access, misuse, modification, or denial of a computer network and network-accessible resources.

**ACKNOWLEDGEMENT**

It is with immense pleasure that we would like to express our indebted gratitude to our guide Mr. Pullela SVVSR Kumar, who has guided and encouraged to our project.

We feel thankful to our HOD Dr. P.S.V.V.S.R kumar of Aditya College Of Engineering, for his cooperation and help in completion of our project and throughout our course.

We avail this opportunity to express our heartful thanks to our principal Mr. Ramesh of Aditya College Of Engineering, and to the management for providing a great support for our project completion.

S. Sri Gowri 15MH1A0541

T. Sai Rajasekhar Reddy 15MH1A0546

DECLARATION

We here by declare that this project entitled “AWS CLOUD & NETWORK SECURITY” has been undertaken by us and this work has been submitted to ADITYA COLLEGE OF ENGINEERING affiliated to J.N.T.U Kakinada, in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Computer Science & Engineering.

We declare that our project entitled “AWS CLOUD & NETWORK SECURITY” is not done in any university.

**Project Associates**

S. Sri Gowri 15MH1A0541

T. Sai Rajasekhar Reddy 15MH1A0546

**INDEX**

1. INTRODUCTION

2. REQUIREMENT ANALYSIS

2.1 Hardware & Software Requirements

2.2 Services & Platforms

3. SYSTEM DESIGN

3.1 UML Diagrams

3.1.1 Use Case Diagram

3.1.2 Activity Diagram

3.1.3 Sequence Diagram

4. AWS CLOUD

4.1 What is AWS and why AWS?

4.2

5. Website and DB details

5.1 Main web site code

5.2 IOT page code

5.3 Database Creation

5.3.1 Setting Database Path

5.3.2 Downloading and Installing Node js

5.3.3 Connecting Database to Website

6.0 IoT Set up code

6.1 MQTT Connection

6.2 Storing Atmosphere details into Database tables

6. ELASTIC LOAD BALANCER

**1. INTRODUCTION**

**2. REQUIREMENT ANALYSIS**

**2.1 Hardware & Software Requirements:**

Cisco Networking Devices (Switch and Router)

Html, CSS, Bootstrap

JavaScript

**2.2 Services and Platforms:**

AWS (VPC, S3)

Load Balancing

Auto Scaling

**3.** **SYSTEM DESIGN**

The architecture or the structure of our project is explained to have a brief idea of how it gets processed to meet the requirements of the user.

**3.1** **UML Diagrams**:

**3.1.1 Use Case Diagram:**

Use case Diagrams represent the functionality of the system from a user’s point of view. Use case focuses on the behaviour of the system from external point of view.

**Use Case:** Use case describes the behaviour of a system. It is used to structure things in a model. It contains multiple scenarios, each of which describes a sequence of actions that is clear enough for outsiders to understand.

**Actor:** An actor represents a coherent set of roles that users of a system play when interacting with the use cases of the system. An actor participates in use cases to accomplish an overall purpose. An actor can represent the role of a human, a device, or any other system.

**Description:** When a commercial client comes to have a deal with an idea for providing the advance maintenance of his data and security policies. He can choose AWS as the better platform. The developer creates a webservers to run the commercial website, provides secure login credentials to the user. If the number of users increases beyond the range then load balancing and auto-scaling comes up, the average percentage of c.p.u is calculated and allocates the web instances and remove the instances when the percentage goes down.

**Use Case View**



**3.1.2 Activity Diagram:**



**3.1.3 Sequence Diagram**

**4. AWS**

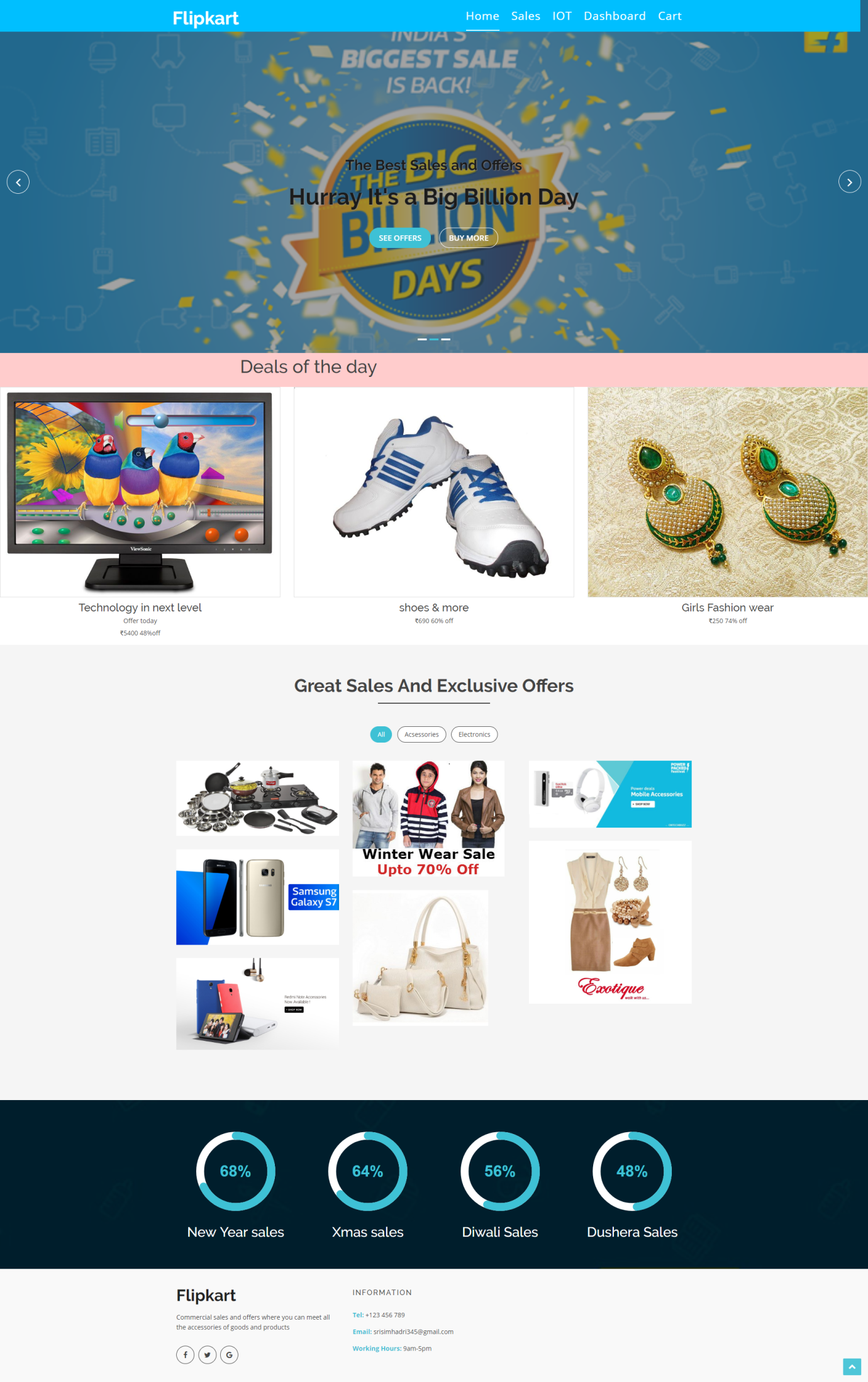
**Amazon Web Services (AWS)** is owned by Amazon that provides On-demand cloud computing platforms to individuals, companies and governments, on a paid subscription basis. The technology allows subscribers to have at their disposal a virtual cluster of computers, available at all time through the Internet.

**Why AWS?**

To explain these auto scaling and load balancing functionalities on a cloud, we have designed a web portal using HTML, CSS, JavaScript and Bootstrap as the programming languages.

* Now create a public folder where we need to place our main web page that we want to display when we start the web server.

(Index.html 🡪the main web page that was designed to launch on to aEC2 instance.)



Html code for our website

<!doctype html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>Commercial</title>

<meta content="width=device-width, initial-scale=1.0" name="viewport">

<meta content="" name="keywords">

<meta content="" name="description">

<!-- Favicons -->

<link href="img/img1.jpg" rel="icon">

<!-- Google Fonts -->

<link href="https://fonts.googleapis.com/css?family=Open+Sans:300,400,400i,600,700|Raleway:300,400,400i,500,500i,700,800,900" rel="stylesheet">

<!-- Bootstrap CSS File -->

<link href="lib/bootstrap/css/bootstrap.min.css" rel="stylesheet">

<!-- Libraries CSS Files -->

<link href="lib/nivo-slider/css/nivo-slider.css" rel="stylesheet">

<link href="lib/owlcarousel/owl.carousel.css" rel="stylesheet">

<link href="lib/owlcarousel/owl.transitions.css" rel="stylesheet">

<link href="lib/font-awesome/css/font-awesome.min.css" rel="stylesheet">

<link href="lib/animate/animate.min.css" rel="stylesheet">

<link href="lib/venobox/venobox.css" rel="stylesheet">

<!-- Nivo Slider Theme -->

<link href="css/nivo-slider-theme.css" rel="stylesheet">

<!-- Main Stylesheet File -->

<link href="css/style.css" rel="stylesheet">

<!-- Responsive Stylesheet File -->

<link href="css/responsive.css" rel="stylesheet">

<style>

.grid-container {

display: grid;

grid-template-columns: auto auto auto auto;

grid-gap: 10px;

background-color:hsl(0, 100%, 90%);;

padding: 10px;

}

}

</style>

</head>

<body data-spy="scroll" data-target="#navbar-example">

<div id="preloader"></div>

<header>

<!-- header-area start -->

<div id="sticker" class="header-area">

<div class="container">

<div class="row">

<div class="col-md-12 col-sm-12">

<!-- Navigation -->

<nav class="navbar navbar-default">

<!-- Brand and toggle get grouped for better mobile display -->

<div class="navbar-header">

<button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target=".bs-example-navbar-collapse-1" aria-expanded="false">

<span class="sr-only">Toggle navigation</span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

</button>

<!-- Brand -->

<a class="navbar-brand page-scroll sticky-logo" href="index.html">

<h1>Flipkart</h1>

<!-- Uncomment below if you prefer to use an image logo -->

<!-- <img src="img/logo.png" alt="" title=""> -->

</a>

</div>

<!-- Collect the nav links, forms, and other content for toggling -->

<div class="collapse navbar-collapse main-menu bs-example-navbar-collapse-1" id="navbar-example">

<ul class="nav navbar-nav navbar-right">

<li class="active">

<a class="page-scroll" href="#home">Home</a>

</li>

<li>

<a class="page-scroll" href="#about">Sales</a>

</li>

<li>

<a class="page-scroll" href="iot.html">IOT</a>

</li>

<li>

<a class="page-scroll">Dashboard</a>

</li>

<li>

<a class="page-scroll">Cart</a>

</li>

</ul>

</div>

<!-- navbar-collapse -->

</nav>

<!-- END: Navigation -->

</div>

</div>

</div>

</div>

<!-- header-area end -->

</header>

<!-- header end -->

<!-- Start Slider Area -->

<div id="home" class="slider-area">

<div class="bend niceties preview-2">

<div id="ensign-nivoslider" class="slides">

<img src="img/slider/img3.jpg" alt="" title="#slider-direction-1" />

<img src="img/slider/img4.png" alt="" title="#slider-direction-2" />

<img src="img/slider/img5.png" alt="" title="#slider-direction-3" />

</div>

<!-- End Slider Area -->

</div>

</div>

<!-- direction 1 -->

<div id="slider-direction-1" class="slider-direction slider-one">

<div class="container">

<div class="row">

<div class="col-md-12 col-sm-12 col-xs-12">

<div class="slider-content">

<!-- layer 1 -->

<div class="layer-1-1 hidden-xs wow slideInDown" data-wow-duration="2s" data-wow-delay=".2s">

<h2 class="title1" style="color:#FF3855; text-shadow:1px 1px 1px grey;">The Best Sales </h2>

</div>

<!-- layer 2 -->

<div class="layer-1-3 hidden-xs wow slideInUp" data-wow-duration="2s" data-wow-delay=".2s">

<a class="ready-btn right-btn page-scroll" href="#services">See Offers</a>

<a class="ready-btn page-scroll" href="#about">Buy More</a>

</div>

</div>

</div>

</div>

</div>

</div>

<!-- direction 2 -->

<div id="slider-direction-2" class="slider-direction slider-two">

<div class="container">

<div class="row">

<div class="col-md-12 col-sm-12 col-xs-12">

<div class="slider-content">

<!-- layer 1 -->

<div class="layer-1-1 hidden-xs wow slideInDown" data-wow-duration="2s" data-wow-delay=".2s">

<h2 class="title1" style="color:#1B1B1B; text-shadow:1px 1px 1px grey;">The Best Sales and Offers</h2>

</div>

<!-- layer 2 -->

<div class="layer-1-2 wow slideInUp" data-wow-duration="2s" data-wow-delay=".1s">

<h1 class="title2" style="color:#1B1B1B; text-shadow:1px 1px 1px grey;">Hurray It's a Big Billion Day</h1>

</div>

<!-- layer 3 -->

<div class="layer-1-3 hidden-xs wow slideInUp" data-wow-duration="2s" data-wow-delay=".2s">

<a class="ready-btn right-btn page-scroll" href="#services">See Offers</a>

<a class="ready-btn page-scroll" href="#about">Buy More</a>

</div>

</div>

</div>

</div>

</div>

</div>

<!-- direction 1 -->

<div id="slider-direction-3" class="slider-direction slider-three">

<div class="container">

<div class="row">

<div class="col-md-12 col-sm-12 col-xs-12">

<div class="slider-content">

<!-- layer 1 -->

<div class="layer-1-1 hidden-xs wow slideInDown" data-wow-duration="2s" data-wow-delay=".2s">

<h2 class="title1" style="color:#1B1B1B;">Great Indian Sales</h2>

</div>

</div>

</div>

</div>

</div>

</div>

<!-- Start sales Area -->

<div class="grid-container">

<div class="container text-center" >

<div class="row">

<h2>Deals of the day</h2>

<!--<button id="btn\_click">view more</button!-->

</div>

</div>

</div>

<div class="row">

<div class="col-md-4 col-sm-3 col-xs-12">

<div class="single-team-member">

<a href="#">

<img src="img/sales/14.jpg" alt="">

</a>

<div class="team-social-icon text-center">

<ul>

<li>

<a href="#">

<i class="fa fa-gittip"></i>

</a>

</li>

</ul>

</div>

</div>

<div class="team-content text-center">

<h4>Technology in next level</h4>

<p>Offer today</p>

<p>₹5400 48%off</p>

</div>

</div>

<!-- End column -->

<!-- <div class="team-top"> -->

<div class="col-md-4 col-sm-3 col-xs-12">

<div class="single-team-member">

<a href="#">

<img src="img/sales/12.jpg" alt="">

</a>

<div class="team-social-icon text-center">

<ul>

<li>

<a href="#">

<i class="fa fa-gittip"></i>

</a>

</li>

</ul>

</div>

</div>

<div class="team-content text-center">

<h4>shoes & more</h4>

<p>₹690 60% off</p>

</div>

</div>

<!-- End column -->

<!-- <div class="team-top"> -->

<div class="col-md-4 col-sm-3 col-xs-12">

<div class="single-team-member">

<a href="#">

<img src="img/sales/13.jpg" alt="">

</a>

<div class="team-social-icon text-center">

<ul>

<li>

<a href="#">

<i class="fa fa-gittip"></i>

</a>

</li>

</ul>

</div>

</div>

<div class="team-content text-center">

<h4>Girls Fashion wear</h4>

<p>₹250 74% off</p>

</div>

</div>

</div>

<!-- End column -->

</div>

</div>

</div>

</div>

<!-- End Team Area -->

<!-- Start About area -->

<div id="about" class="about-area area-padding">

<div class="container">

<div class="row">

<div class="col-md-12 col-sm-12 col-xs-12">

<div class="section-headline text-center">

<h2>Great Sales and Exclusive Offers</h2>

</div>

</div>

</div>

<div class="row">

<!-- Start Portfolio -page -->

<div class="awesome-project-1 fix">

<div class="col-lg-12 col-md-12 col-sm-12 col-xs-12">

<div class="awesome-menu ">

<ul class="project-menu">

<li>

<a href="#" class="active" data-filter="\*">All</a>

</li>

<li>

<a href="#" data-filter=".design">Acsessories</a>

</li>

<li>

<a href="#" data-filter=".photo">electronics</a>

</li>

</ul>

</div>

</div>

</div>

<div class="awesome-project-content">

<!-- single-awesome-project start -->

<div class="col-md-4 col-sm-4 col-xs-12 design development">

<div class="single-awesome-project">

<div class="awesome-img">

<a href="#"><img src="img/sales/kitchen.jpg" alt="" /></a>

<div class="add-actions text-center">

<div class="project-dec">

<a class="venobox" data-gall="myGallery" href="img/sales/kitchen.jpg">

<h4>Kitchen Items 50% off</h4>

</a>

</div>

</div>

</div>

</div>

</div>

<!-- single-awesome-project end -->

<!-- single-awesome-project start -->

<div class="col-md-4 col-sm-4 col-xs-12 design">

<div class="single-awesome-project">

<div class="awesome-img">

<a href="#"><img src="img/sales/winter.jpg" alt="" /></a>

<div class="add-actions text-center">

<div class="project-dec">

<a class="venobox" data-gall="myGallery" href="img/sales/winter.jpg">

<h4>Winter shopping</h4>

</a>

</div>

</div>

</div>

</div>

</div>

<!-- single-awesome-project end -->

<!-- single-awesome-project start -->

<div class="col-md-4 col-sm-4 col-xs-12 photo development">

<div class="single-awesome-project">

<div class="awesome-img">

<a href="#"><img src="img/sales/accessories.jpg" alt="" /></a>

<div class="add-actions text-center">

<div class="project-dec">

<a class="venobox" data-gall="myGallery" href="img/sales/accessories.jpg">

<h4>Gadgets</h4>

<span>On sales</span>

</a>

</div>

</div>

</div>

</div>

</div>

<!-- single-awesome-project end -->

<!-- single-awesome-project start -->

<div class="col-md-4 col-sm-4 col-xs-12 design">

<div class="single-awesome-project">

<div class="awesome-img">

<a href="#"><img src="img/sales/7.jpeg" alt="" /></a>

<div class="add-actions text-center">

<div class="project-dec">

<a class="venobox" data-gall="myGallery" href="img/sales/7.jpeg">

<h4>Fashion Week</h4>

<span>40% sales</span>

</a>

</div>

</div>

</div>

</div>

</div>

<!-- single-awesome-project end -->

<!-- single-awesome-project start -->

<div class="col-md-4 col-sm-4 col-xs-12 photo development">

<div class="single-awesome-project">

<div class="awesome-img">

<a href="#"><img src="img/sales/galaxy.jpg" alt="" /></a>

<div class="add-actions text-center">

<div class="project-dec">

<a class="venobox" data-gall="myGallery" href="img/sales/galaxy.jpg">

<h4>Galaxy</h4>

<span>On sales</span>

</a>

</div>

</div>

</div>

</div>

</div>

<!-- single-awesome-project end -->

<div class="col-md-4 col-sm-4 col-xs-12 design">

<div class="single-awesome-project">

<div class="awesome-img">

<a href="#"><img src="img/sales/2.jpg" alt="" /></a>

<div class="add-actions text-center">

<div class="project-dec">

<a class="venobox" data-gall="myGallery" href="img/sales/2.jpg">

<h4>Handbags and clutches</h4>

<span>30% OFF</span>

</a>

</div>

</div>

</div>

</div>

</div>

<!-- single-awesome-project end -->

<!-- single-awesome-project start -->

<div class="col-md-4 col-sm-4 col-xs-12 photo development">

<div class="single-awesome-project">

<div class="awesome-img">

<a href="#"><img src="img/sales/3.jpg" alt="" /></a>

<div class="add-actions text-center">

<div class="project-dec">

<a class="venobox" data-gall="myGallery" href="img/sales/3.jpg">

<h4>Redmi on sales</h4>

</a>

</div>

</div>

</div>

</div>

</div>

<!-- single-awesome-project end -->

</div>

</div>

</div>

</div>

<!-- awesome-portfolio end -->

<!-- our-skill-area start -->

<div class="our-skill-area fix hidden-sm">

<div class="test-overly"></div>

<div class="skill-bg area-padding-2">

<div class="container">

<!-- section-heading end -->

<div class="row">

<div class="skill-text">

<!-- single-skill start -->

<div class="col-xs-12 col-sm-3 col-md-3 text-center">

<div class="single-skill">

<div class="progress-circular">

<input type="text" class="knob" value="0" data-rel="85" data-linecap="round" data-width="175" data-bgcolor="#fff" data-fgcolor="#3EC1D5" data-thickness=".20" data-readonly="true" disabled>

<h3 class="progress-h4">New Year sales</h3>

</div>

</div>

</div>

<!-- single-skill end -->

<!-- single-skill start -->

<div class="col-xs-12 col-sm-3 col-md-3 text-center">

<div class="single-skill">

<div class="progress-circular">

<input type="text" class="knob" value="0" data-rel="80" data-linecap="round" data-width="175" data-bgcolor="#fff" data-fgcolor="#3EC1D5" data-thickness=".20" data-readonly="true" disabled>

<h3 class="progress-h4">Xmas sales</h3>

</div>

</div>

</div>

<!-- single-skill end -->

<!-- single-skill start -->

<div class="col-xs-12 col-sm-3 col-md-3 text-center">

<div class="single-skill">

<div class="progress-circular">

<input type="text" class="knob" value="0" data-rel="70" data-linecap="round" data-width="175" data-bgcolor="#fff" data-fgcolor="#3EC1D5" data-thickness=".20" data-readonly="true" disabled>

<h3 class="progress-h4">Diwali Sales</h3>

</div>

</div>

</div>

<!-- single-skill end -->

<!-- single-skill start -->

<div class="col-xs-12 col-sm-3 col-md-3 text-center">

<div class="single-skill">

<div class="progress-circular">

<input type="text" class="knob" value="0" data-rel="60" data-linecap="round" data-width="175" data-bgcolor="#fff" data-fgcolor="#3EC1D5" data-thickness=".20" data-readonly="true" disabled>

<h3 class="progress-h4">Dushera Sales</h3>

</div>

</div>

</div>

<!-- single-skill end -->

</div>

</div>

</div>

</div>

</div>

<!-- our-skill-area end -->

<!-- Start Footer bottom Area -->

<footer>

<div class="footer-area">

<div class="container">

<div class="row">

<div class="col-md-4 col-sm-4 col-xs-12">

<div class="footer-content">

<div class="footer-head">

<div class="footer-logo">

<h2>Flipkart</h2>

</div>

<p>Commercial sales and offers where you can meet all the accessories of goods and products</p>

<div class="footer-icons">

<ul>

<li>

<a href="#"><i class="fa fa-facebook"></i></a>

</li>

<li>

<a href="#"><i class="fa fa-twitter"></i></a>

</li>

<li>

<a href="#"><i class="fa fa-google"></i></a>

</li>

</ul>

</div>

</div>

</div>

</div>

<!-- end single footer -->

<div class="col-md-4 col-sm-4 col-xs-12">

<div class="footer-content">

<div class="footer-head">

<h4>information</h4>

<div class="footer-contacts">

<p><span>Tel:</span> +123 456 789</p>

<p><span>Email:</span> srisimhadri345@gmail.com</p>

<p><span>Working Hours:</span> 9am-5pm</p>

</div>

</div>

</div>

</div>

</div>

</div>

</div>

</footer>

<a href="#" class="back-to-top"><i class="fa fa-chevron-up"></i></a>

<!-- JavaScript Libraries -->

<script src="lib/jquery/jquery.min.js"></script>

<script src="lib/bootstrap/js/bootstrap.min.js"></script>

<script src="lib/owlcarousel/owl.carousel.min.js"></script>

<script src="lib/venobox/venobox.min.js"></script>

<script src="lib/knob/jquery.knob.js"></script>

<script src="lib/wow/wow.min.js"></script>

<script src="lib/parallax/parallax.js"></script>

<script src="lib/easing/easing.min.js"></script>

<script src="lib/nivo-slider/js/jquery.nivo.slider.js" type="text/javascript"></script>

<script src="lib/appear/jquery.appear.js"></script>

<script src="lib/isotope/isotope.pkgd.min.js"></script>

<!-- Contact Form JavaScript File -->

<script src="contactform/contactform.js"></script>

<script src="js/main.js"></script>

</body>

</html>

Html code for IOT web page

<html>

<head>

<title>

Atmosphere Data

</title>

<style>

body {

text-align:center;

}

.sensor{

background-color: black;

padding: 20px;

font-size: 35px;

color: white;

margin: 10px;

padding: 10px;

}

table, th, td {

border: 2px solid black;

border-collapse: collapse;

padding: 20px;

margin: 20px;

margin-left:auto;

margin-right:auto;

}

</style>

</head>

<body onload="updateValues()">

<div class="sensor">

<h2 style=" text-align: center;">Atmosphere</h2>

</div>

<table style="width:50%" bgcolor="gainsboro" align="center">

<tr>

<th> <h3 id="temperature">Temperature : </h3> </th>

</tr>

<tr>

<th> <h3 id="humidity">Humidity : </h3> </th>

</tr>

</table>

<script>

function updateValues() {

setInterval(function() {

getData();

},4000);

}

function getData() {

var xhttp = new XMLHttpRequest();

xhttp.onreadystatechange=function() {

if (this.readyState == 4 && this.status == 200) {

let response=JSON.parse(this.response);

document.getElementById("temperature").innerHTML = "Temperature : " + response.temperature;

document.getElementById("humidity").innerHTML = "Humidity : " + response.humidity;

}

};

xhttp.open("GET", "/api/getdata", true);

xhttp.send();

}

function putData(temp,humd){

var xhttp = new XMLHttpRequest();

xhttp.onreadystatechange=function() {

};

xhttp.open("POST", "/api/putdata", true);

xhttp.setRequestHeader("Content-type", "application/json");

xhttp.send(JSON.stringify({temperature:temp,humidity:humd}));

}

</script>

</body>

</html>

This below code is the main source file Index.js to perform whole server operations. This index.js server calls the database table information and acts as an API to connect the webpage and database.

We need to install npm, mysql and express js to run this node js file.

**Npm (node packet manager):** **NPM** basically **is** the package manager for node. It helps with installing various packages and resolving their various dependencies. It greatly helps with your Node development.

**How it is helpful in our project**: Using this npm, we have specified all our project’s dependencies inside the package.json file, when any time we want to run our webserver we can just **run npm install** and all the dependences installed. Manually adding these files kills time and messy.

**MySQL**: AWS support MySQL to store the values into a data base we used MySQL. We can add, delete, retrieve, and modify the values on a database.

**Express Js**: It is a web application framework for nodejs and provides simple API’s to build websites, web-apps, and backends . It is flexible as there are numerous modules available on **npm**, which can be directly plugged into Express.

The database created here is atmospheredb, This atmospheredb contains atmospheretable with field names temperature, humidity and created time.

Mysql DB Creation

#!/bin/bash

# this script is not compatible with dash shell; hence the above shebang is mandatory

# ------------------------------------------------------------------------- #

# This script shall create database and required users.

#INSTRUCTIONS:

#1.This script executes as mysql root

#2.change dbname, pwd values respectively

#3.Depending on VM & mysql installtion edit the dbpath variable

#Scripts to be executed in the following order:

# 1.db\_scripts\_p1.sh script

# 2. create tables, foreign keys etc..

# 3.db\_scripts\_p2.sh script

# Created date : 26-Jan-18

# Initial Author : Sai

# ------------------------------------------------------------------------- #

# ---Variables to be modified at run time. User permissions-------------------------------- #

dbname=atmospheredb

owner\_user='root'

owner\_pwd='passw0rd'

root\_pwd='passw0rd'

dbpath="mysql -u root -e "

#dbpath="$mysql --host=127.0.0.1 -u root -p$root\_pwd -e "

# ------------------------------------------------------------------------- #

# Here $dbname will call node js program, there given database name , it will be recalled and stored in $dbname# ------------------------------------------------------------------------- #

# creating databse steps

$dbpath"CREATE DATABASE $dbname ;"

# ------------------------------------------------------------------------- #

# creating OWNER user / grants steps

$dbpath"CREATE USER '$owner\_user' IDENTIFIED BY '$owner\_pwd' ;"

$dbpath"GRANT ALL ON $dbname.\* TO '$owner\_user' WITH GRANT OPTION;"

#$dbpath "ALTER USER '$owner\_user'@'localhost' IDENTIFIED WITH mysql\_native\_password BY '$owner\_pwd';"

Table Values

Resources to create db:

User.sql=>contains the sql table creation and data to be inserted into created table

create table atmospheretable(

id int auto\_increment,

temperature smallint,

humidity smallint,

created\_time datetime,

PRIMARY KEY (id)

);

insert into atmospheretable(temperature,humidity,created\_time) values(38.6,26,'2018-09-12');

insert into atmospheretable(temperature,humidity,created\_time) values(46.9,12,'2018-09-12');

insert into atmospheretable(temperature,humidity,created\_time) values(37.6,77,'2018-09-12');

insert into atmospheretable(temperature,humidity,created\_time) values(28.4,98,'2018-09-12');

insert into atmospheretable(temperature,humidity,created\_time) values(22.6,32,'2018-09-12');

insert into atmospheretable(temperature,humidity,created\_time) values(59.8,51,'2018-09-12');

------------------------------------------------------------------------

Initialsetupdeploy.sh (Script file for installing node js npm and creating wbsitedeploy directroy)

#!/bin/bash

curl -sL https://rpm.nodesource.com/setup\_10.x | sudo -E bash -;

yum install nodejs --enablerepo=nodesource -y;

npm install npm -y;

mkdir wbsite-deploy;

cd wbsite-deploy;

npm init -y;

npm install express --save;

npm install mysql --save;

node index.js (Main java script file for )

const express = require('express'); // express driver installation

const app = express();

const mysql = require('mysql'); //mysql driver installation

const dbcon = mysql.createConnection({

'host':'localhost', //creating a connection to a database

'port':3306,

'user':'root',

'database':'atmospheredb'

});

app.use(express.json());

app.use(express.static('public'));

dbcon.connect(function(err,res){

if(err) {

console.log("err",err);

}else{

console.log("success");

}

});

app.get('/api/getdata', function(req,res){//

getData(req, res);

});

app.post('/api/putdata', function(req,res){

putData(req.body, res);

});

function getData(req,res){

dbcon.query('select \* from atmospheretable order by created\_time desc limit 1;',function(error,result){

if(error)

{

res.send(error);

}else

{

res.send(result.length?result[0]:result);

}

});

}

**//insert into atmosphere tables the values**

function putData(req,res){

dbcon.query('insert into atmospheretable (temperature,humidity,created\_time) values (?,?,now()) ;',[req.temperature,req.humidity],function(error,result){

if(error) {

res.send(error);

}else {

res.send(result.length?result[0]:result);

}

});

}

app.listen(3001, () => console.log('Server running on port 3001')); //acknowledgement to the user that server is running.

MQTT setup

import ast

import paho.mqtt.client as mqtt

import ConfigParser

import simplejson as json

import logging

import datetime

import pymysql

LOG\_FILENAME = 'FeedbackServer.log'

logging.basicConfig(filename=LOG\_FILENAME,level=logging.DEBUG,format='%(asctime)s, %(levelname)s, %(message)s', datefmt='%Y-%m-%d %H:%M:%S')

class MQtt:

def \_\_init\_\_(self,host,port,subTopic,cur,db,timealive=60):

self.host = host

self.port = port

self.timealive = timealive

self.payload = None

self.subTopic = subTopic

self.pubTopic = "feedback-serv"

self.cur = cur

self.db = db

def \_\_on\_connect(self,client, userdata, flags, rc):

try:

print "Connected with result code "+str(rc)

if self.subTopic!=None:

(result,mid)= client.subscribe(self.subTopic)

print result

except Exception as e:

logging.error("The on\_connect error %s,%s"%(e,type(e)))

def \_\_on\_message(self,client, userdata, msg):

try:

data = msg.payload

message = ast.literal\_eval(data)

temperature = int(message["temperature"])

humidity = int(message["humidity"])

print temperature,humidity

insertStatement = "INSERT INTO atmospheretable (id,temperature,humidity,created\_time) VALUES (default,"+str(temperature)+","+str(humidity)+",now());"

print insertStatement

try:

self.cur.execute(insertStatement)

except Exception as e:

print e

except Exception as e:

print e

logging.error("The on\_message error %s,%s"%(e,type(e)))

def connect(self):

try:

self.mqttc = mqtt.Client()

self.mqttc.on\_connect = self.\_\_on\_connect

self.mqttc.on\_message = self.\_\_on\_message

self.mqttc.connect(self.host,self.port,self.timealive)

if self.subTopic != None:

print "Hu"

self.mqttc.loop\_start()

self.mqttc.loop\_forever()

except Exception as e:

logging.error("The connect error %s,%s"%(e,type(e)))

def send(self,message):

try:

(result,mid) = self.mqttc.publish(self.pubTopic,message,2)

return result

except Exception as e:

print e

logging.error("The send error %s,%s"%(e,type(e)))

if \_\_name\_\_ == '\_\_main\_\_':

# Mysql details

host = "localhost"

database = "atmospheredb"

table = "atmospheretable"

db = pymysql.connect(host="localhost",db="atmospheredb",autocommit=True)

cur = db.cursor()

# MQTT DETIALS

host = "18.232.46.180"

port = 1883

subTopic = "dht-resp"

mq = MQtt(host,port,subTopic,cur,db)

mq.connect()

Storing Atmosphere Details into Database table

import pymysql

import time

host = "localhost"

db = pymysql.connect(host="localhost",db="atmospheredb",autocommit=True)

cur = db.cursor()

temperature = 64

humidity = 36

print temperature,humidity

insertStatement = "INSERT INTO atmospheretable (id,temperature,humidity,created\_time) VALUES (default,"+str(temperature)+","+str(humidity)+",now());"

print insertStatement

try:

cur.execute(insertStatement)

time.sleep(10)

except Exception as e:

print e

sqlQuery = "select \* from atmospheretable;"

#Fetch all the rows - for the SQL Query

cur.execute(sqlQuery)

rows = cur.fetchall()

for row in rows:

print(row)

**Amazon EC2:**

Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the Amazon Web Services (AWS) cloud. Using Amazon EC2 eliminates your need to invest in hardware up front, so you can develop and deploy applications faster. You can use Amazon EC2 to launch as many or as few virtual servers as you need, configure security and networking, and manage storage. Amazon EC2 enables you to scale up or down to handle changes in requirements or spikes in popularity, reducing your need to forecast traffic.

**Features of Amazon EC2**

Amazon EC2 provides the following features:

* Virtual computing environments, known as *instances*
* Preconfigured templates for your instances, known as *Amazon Machine Images (AMIs)*, that package the bits you need for your server (including the operating system and additional software)
* Various configurations of CPU, memory, storage, and networking capacity for your instances, known as *instance types*
* Secure login information for your instances using *key pairs* , Storage volumes for temporary data that's deleted when you stop or terminate your instance, known as *instance store volumes*
* Persistent storage volumes for your data using Amazon Elastic Block Store (Amazon EBS), known as *Amazon EBS volumes*
* Multiple physical locations for your resources, such as instances and Amazon EBS volumes, known as *regions* and *Availability Zones*
* A firewall that enables you to specify the protocols, ports, and source IP ranges that can reach your instances using *security groups*
* Metadata, known as *tags*, that you can create and assign to your Amazon EC2 resources
* Virtual networks you can create that are logically isolated from the rest of the AWS cloud, and that you can optionally connect to your own network, known as *virtual private clouds* (VPCs)

**How to launch an EC2 instance on AWS:**

Now we need to deploy this web portal on our EC2 instance to make this portal available for multiple users who wants to make use of.

Creation of Instance in AWS Console:

Step 1: Choose the Instance option in AWS EC2.

Step 2: Launch Instance that was on the top right side of the screen displayed.

Step 3: Choose an Amazon Machine Image(AMI).We choose (**Amazon Linux AMI 2018.03.0 (HVM), SSD Volume Type** - ami-0080e4c5bc078760e) free tier to launch our web portal.

Step 4: Now choose an Instance type which fits to different use cases. Instances are Virtual Web servers that run our application. We choose (**:** t2.micro (Variable ECUs, 1 vCPUs, 2.5 GHz, Intel Xeon Family, 1 GiB memory, EBS only)) for general purpose.

Step 5: Configure Instance details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the lower pricing, assign an access management role to the instance, and more.

Step 6: Add security groups

Step 7: On the **Review Instance Launch** page, choose **Launch**.

When prompted for a key pair, select **Choose an existing key pair**, then select the key pair that you created when getting set up.

Alternatively, you can create a new key pair. Select **Create a new key pair**, enter a name for the key pair, and then choose **Download Key Pair**.

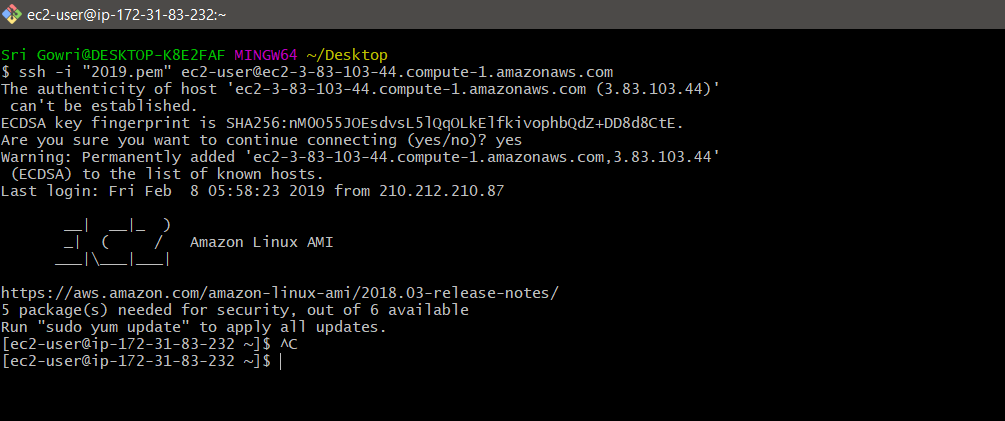
This is the only chance to save the private key file, so be sure to download it. Save the private key file in a safe place. You'll need to provide the name of your key pair when launching an instance and the corresponding private key each time you connect to the instance.

* Now launch the web portal on the created EC2 Instance. To deploy the web portal we need to have a little idea about LINUX.
* Install and run Gitbash, to manage ec2 instances.
* Install filezilla. To move our web code from desktop to cloud ec2 instance.

**Execute the following commands on Gitbash console:**

**1.To connect EC2 instance remotely use the below command.**

**$ssh -i "2019.pem"** [**ec2-user@ec2-3-83-103-44.compute-1.amazonaws.com**](mailto:ec2-user@ec2-3-83-103-44.compute-1.amazonaws.com)



**Procedure for running scripts on server:**

Sudo su //user permissions to access the EC2

Ls

Cd Resources/

ls

chmod 700 db\_setup.sh

chmod 700 initialSetupwensiteDeploy.sh

yum install mysql-server

service mysqld start

./db\_setup.sh

mysql -u root

mysql> show databases;

[root@ip-172-31-84-75 Resources]# ls

db\_setup.sh initialSetupWensiteDeploy.sh User.sql

[root@ip-172-31-84-75 Resources]# mysql atmospheredb <User.sql

[root@ip-172-31-84-75 Resources]# mv initialSetupWensiteDeploy.sh /home/ec2-user/

[root@ip-172-31-84-75 Resources]# ls

db\_setup.sh User.sql

[root@ip-172-31-84-75 Resources]# cd ..

[root@ip-172-31-84-75 ec2-user]# ls

as.pem initialSetupWensiteDeploy.sh Resources

[root@ip-172-31-84-75 ec2-user]# ./initialSetupWensiteDeploy.sh

## Installing the NodeSource Node.js 10.x repo...

[root@ip-172-31-84-75 ec2-user]# cd db

[root@ip-172-31-84-75 db]# ls

index.js public

[root@ip-172-31-84-75 db]# mv index.js /home/ec2-user/website-deploy/

[root@ip-172-31-84-75 db]# mv public /home/ec2-user/website-deploy/

[root@ip-172-31-84-75 db]# ls

[root@ip-172-31-84-75 db]# cd ..

[root@ip-172-31-84-75 db]# npm install express –save

[root@ip-172-31-84-75 db]# npm install mysql --save

[root@ip-172-31-84-75 ec2-user]# cd wbsite-deploy/

[root@ip-172-31-84-75 wbsite-deploy]# ls

index.js node\_modules package.json package-lock.json public

[root@ip-172-31-84-75 wbsite-deploy]# node index.js

**5. ELASTIC LOAD BALANCER**

Elastic Load Balancing automatically distributes incoming application traffic across multiple targets, such as Amazon EC2 instances, containers, IP addresses, and Lambda functions. It can handle the varying load of your application traffic in a single Availability Zone or across multiple Availability Zones. Elastic Load Balancing offers three types of load balancers that all feature the high availability, automatic scaling, and robust security necessary to make your applications fault tolerant.

These are classified into 3 types, they are Application Load Balancer, Network Load Balancer, Classic Load Balancer. In our project we have used Application Load Balancer, this operates at the request level(layer 7), routing traffic to targets – EC2 instances, containers, etc. This provides advanced request routing targeted at delivery of modern application architectures. This simplifies and improves the security of our application ,by ensuring that the latest SSl/TLS ciphers and protocols are used at all times.

**High availability**

Elastic Load Balancing automatically distributes traffic across multiple targets – Amazon EC2 instances, containers and IP addresses – in a single Availability Zone or multiple Availability Zones.

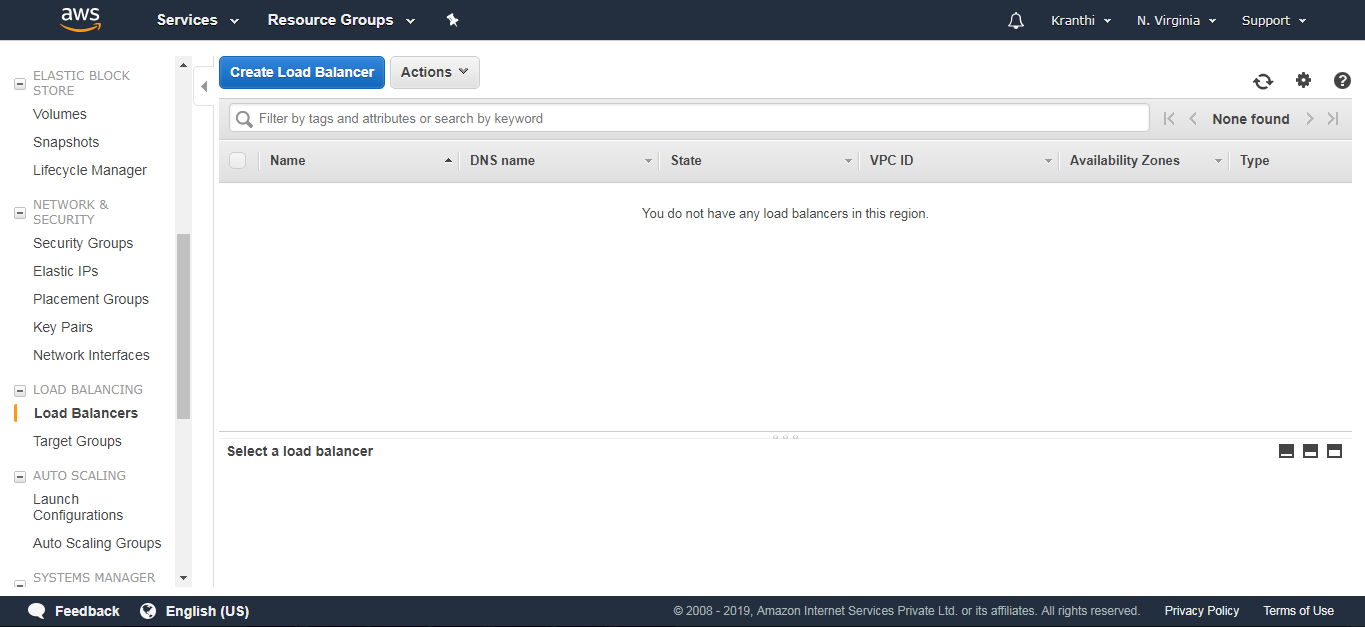
**Health Checks:**

An Application Load Balancer routes traffic only to healthy targets. With an Application Load Balancer, you get improved insight into the health of your applications in two ways: (1) health check improvements that allow you to configure detailed error codes from 200-499. The health checks allow you to monitor the health of each of your services behind the load balancer; and (2) new metrics that give insight into traffic for each of the services running on an EC2 instance.

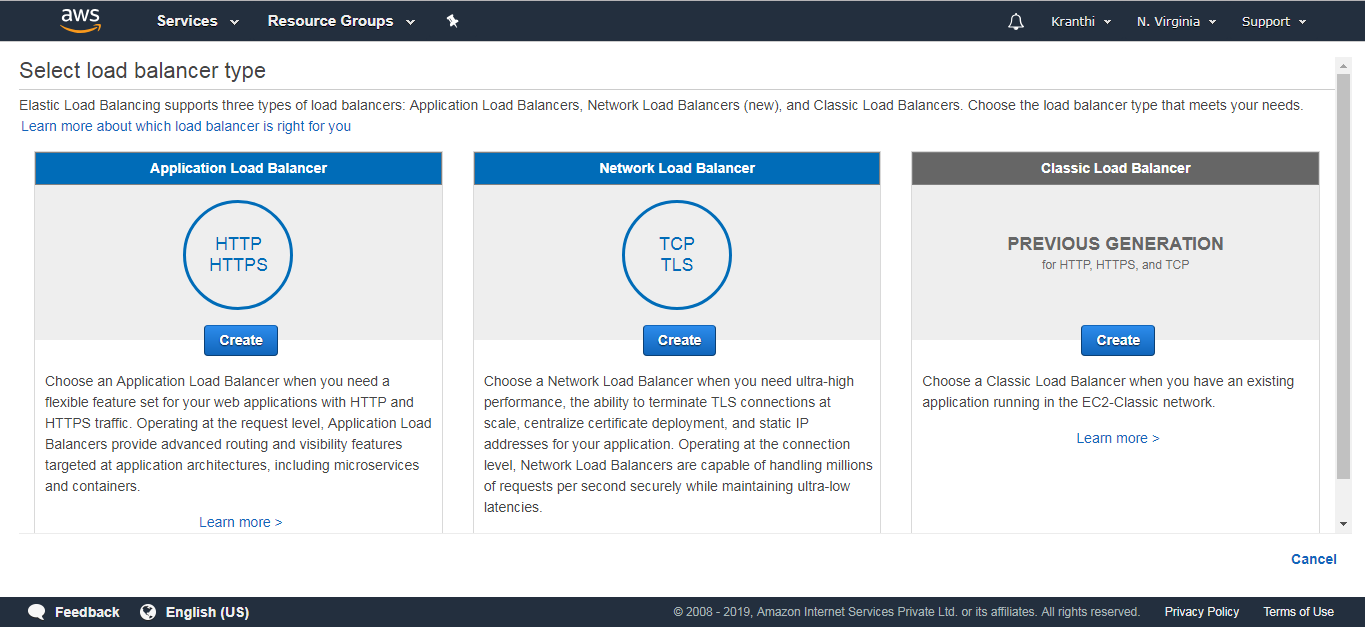
**How to add Load Balancer in AWS :**

Step 1: Login to the AWS Console.

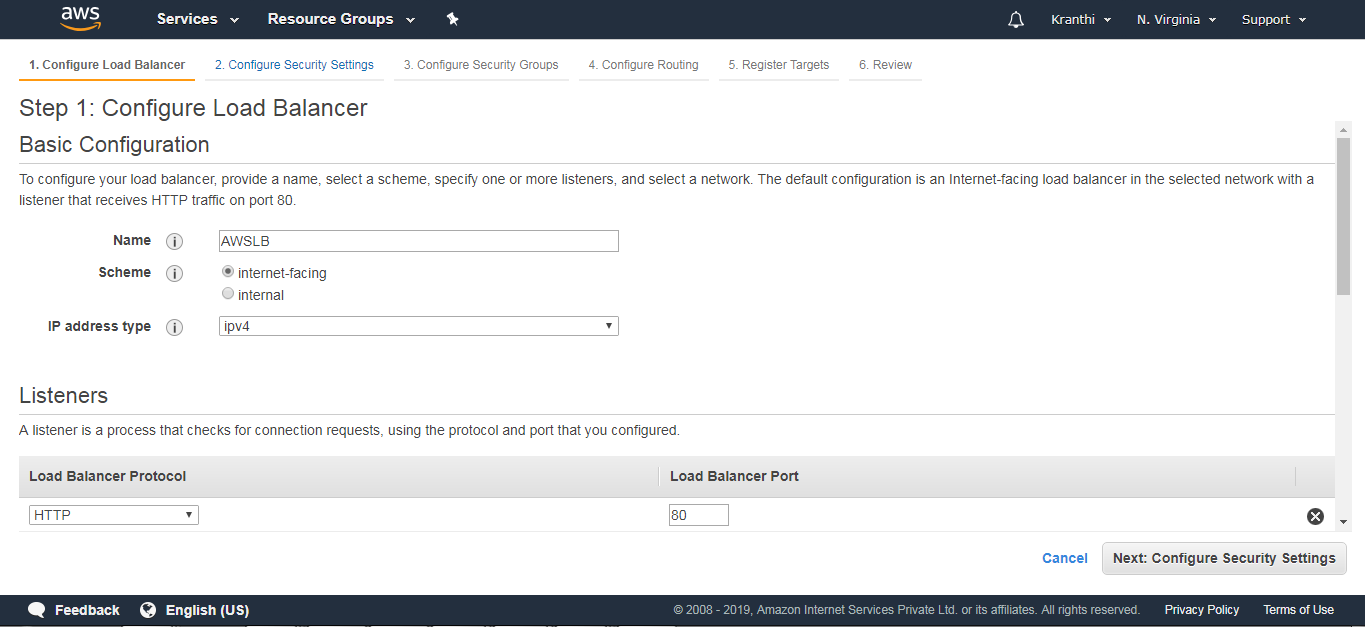
Step 2: Select EC2 Instances, there you find Load Balancer.

Step 3: Choose Create Load Balancer that appeared on the screen below. 

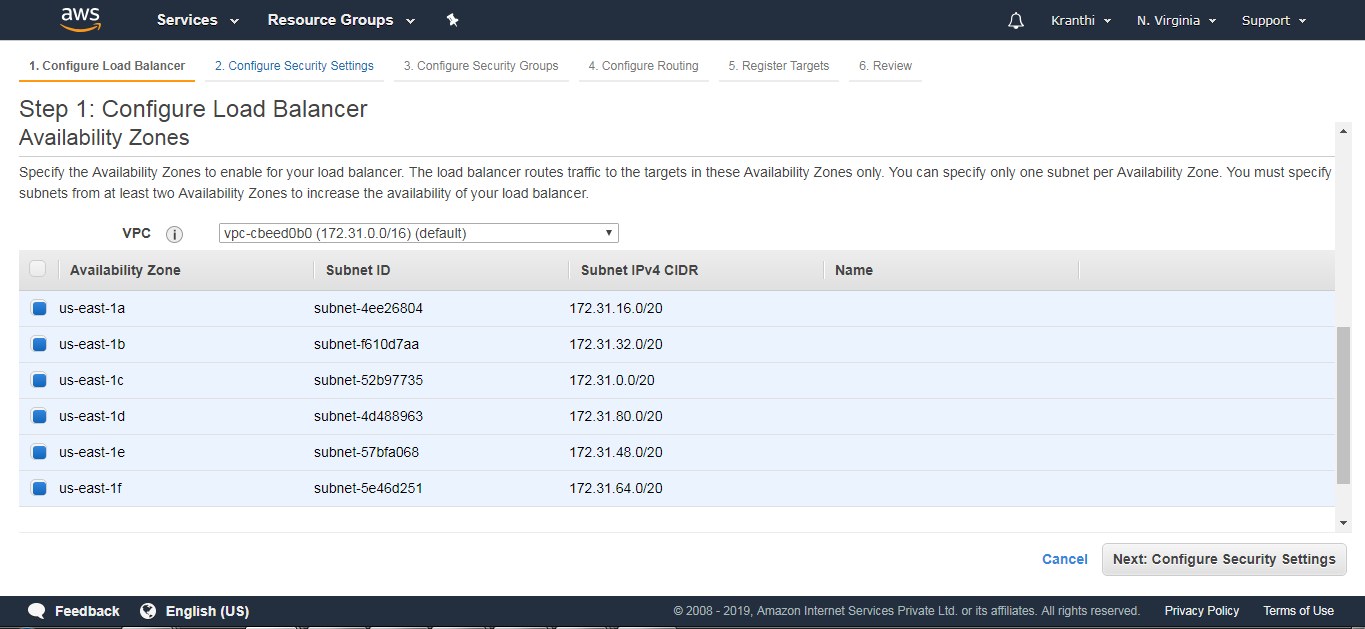
Step 4: Select Application Load Balancer and click on create.



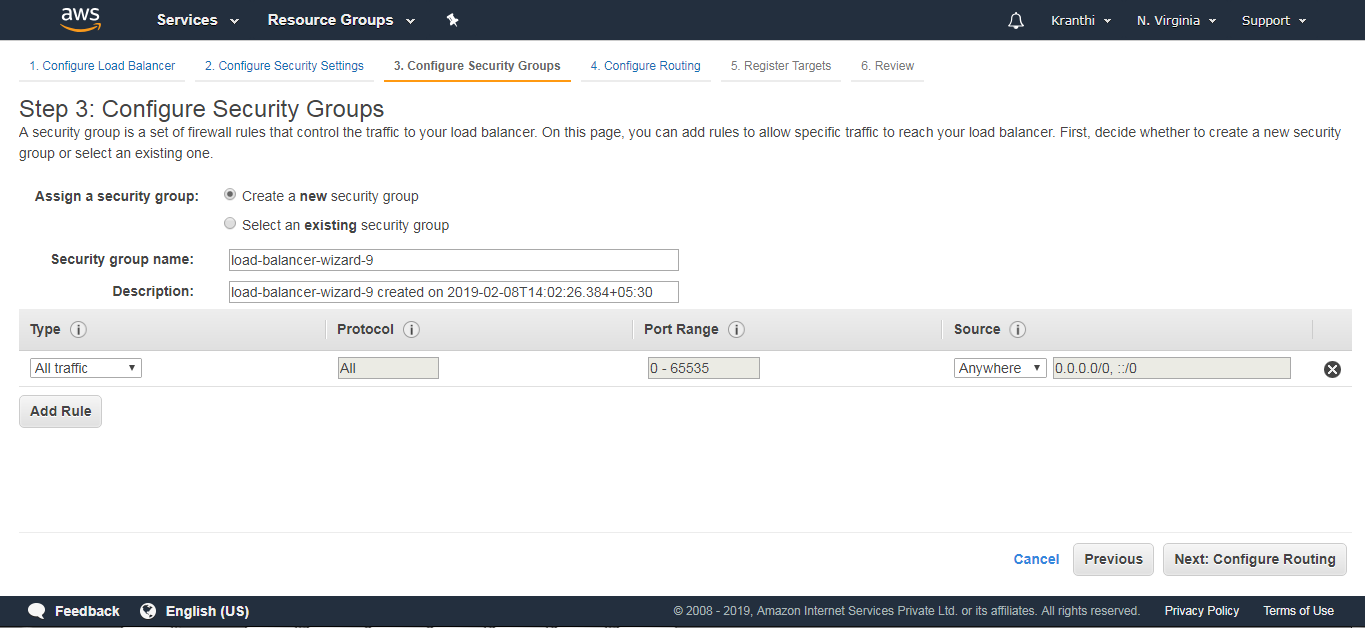
Step 5: Configure Load Balancer.



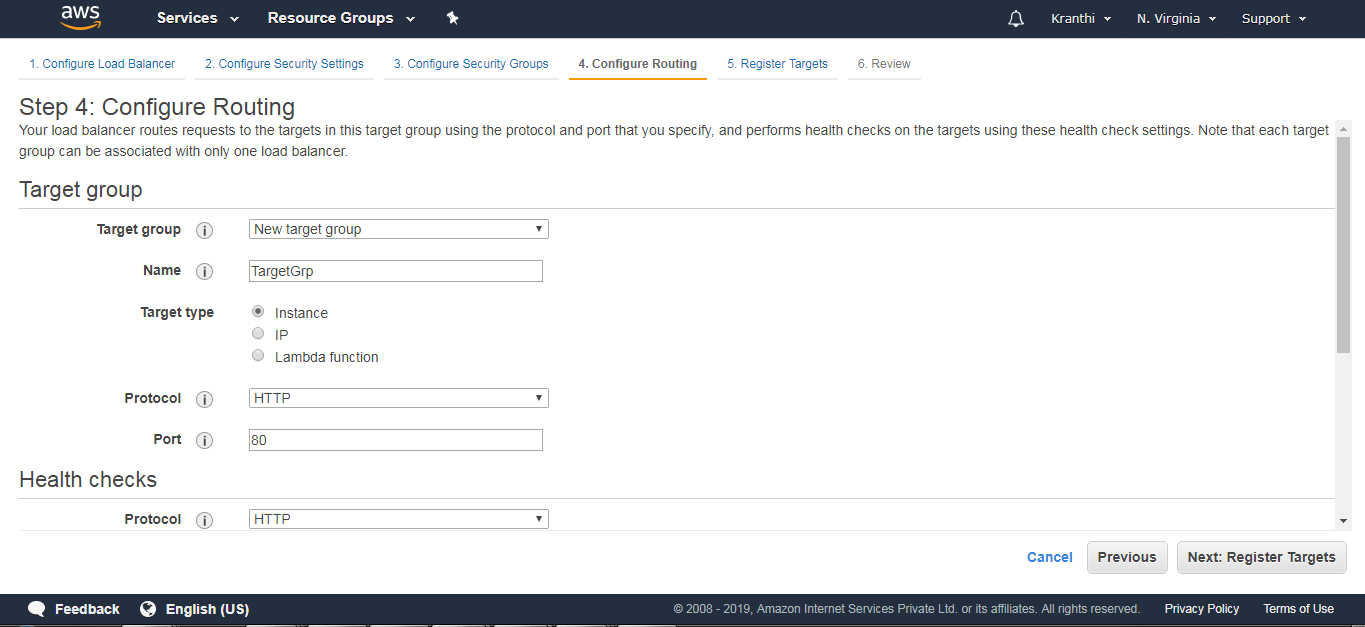
Add the availability Zones

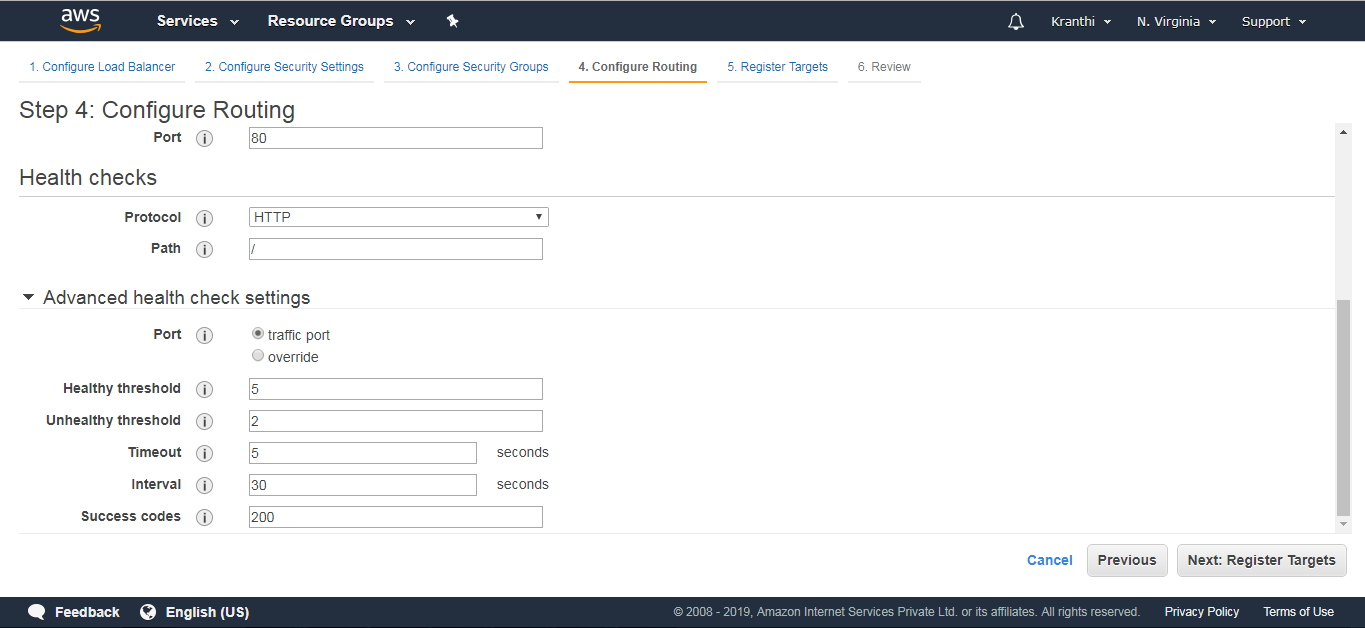


Step 6: Configure Security groups.

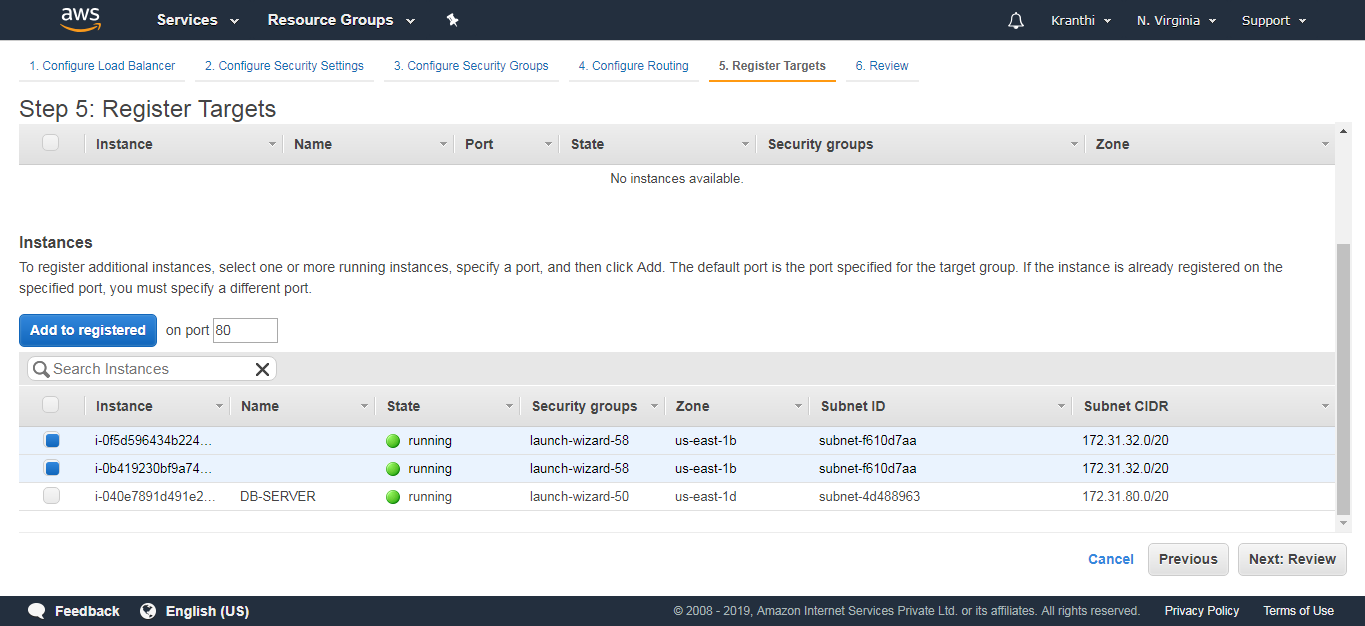


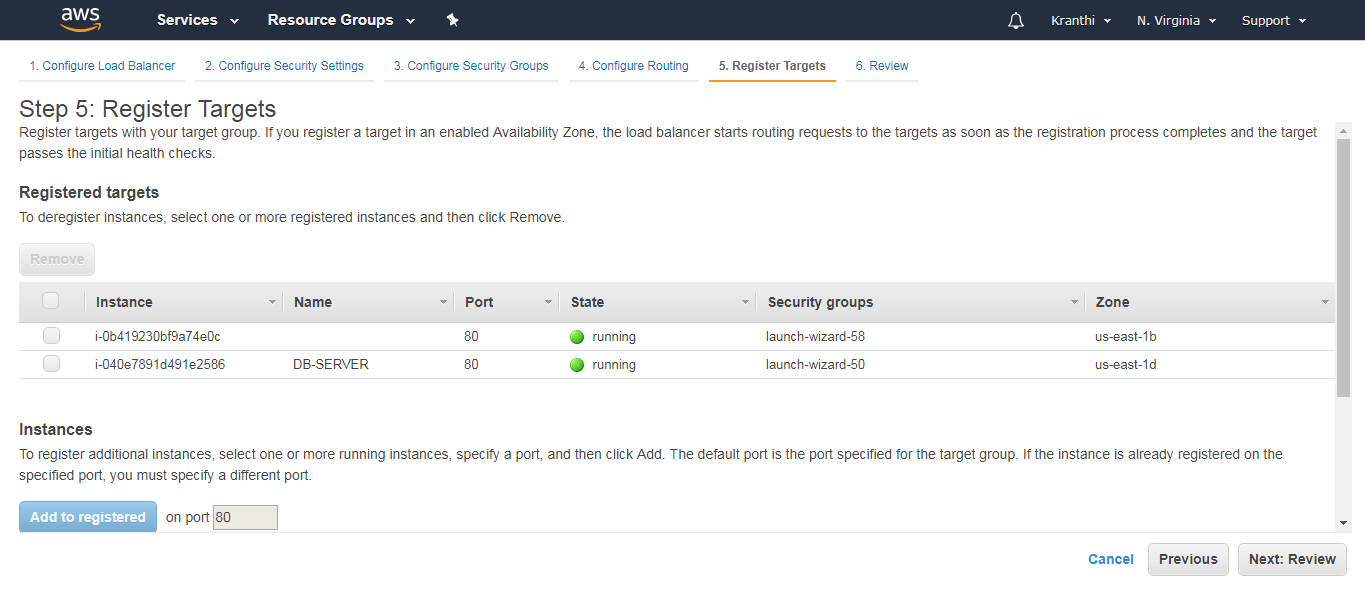
Step 7:Configure Routing



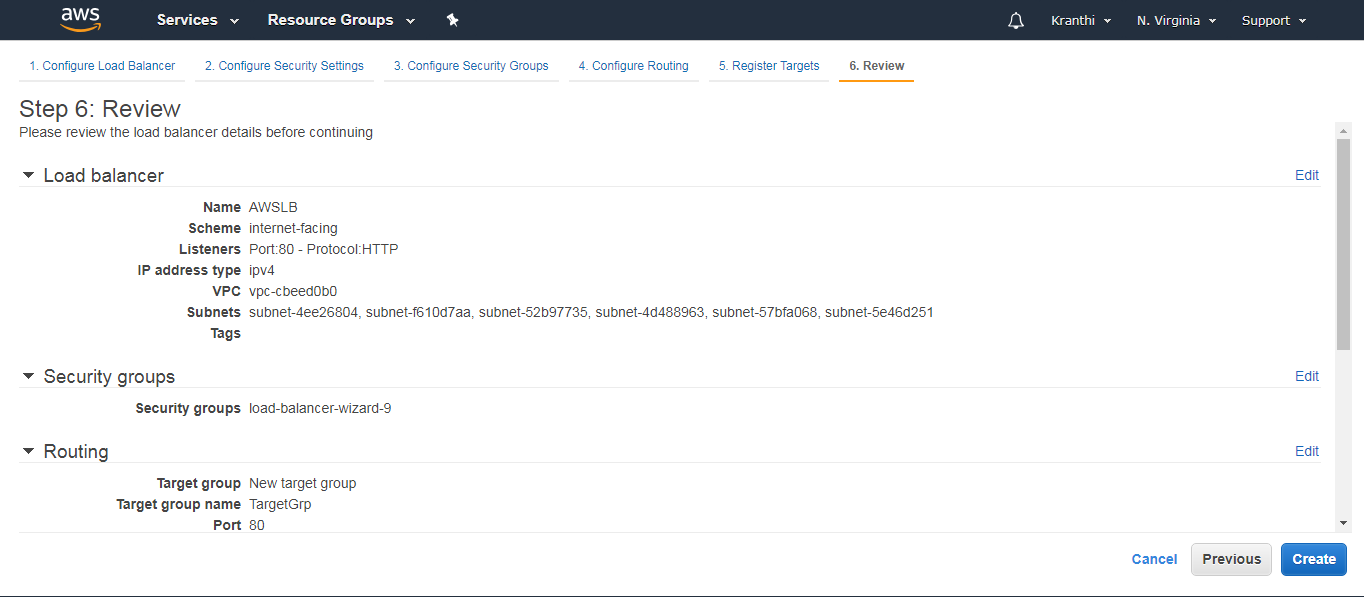


Step 8: Register Targets (nothing but EC2 instances that are created).

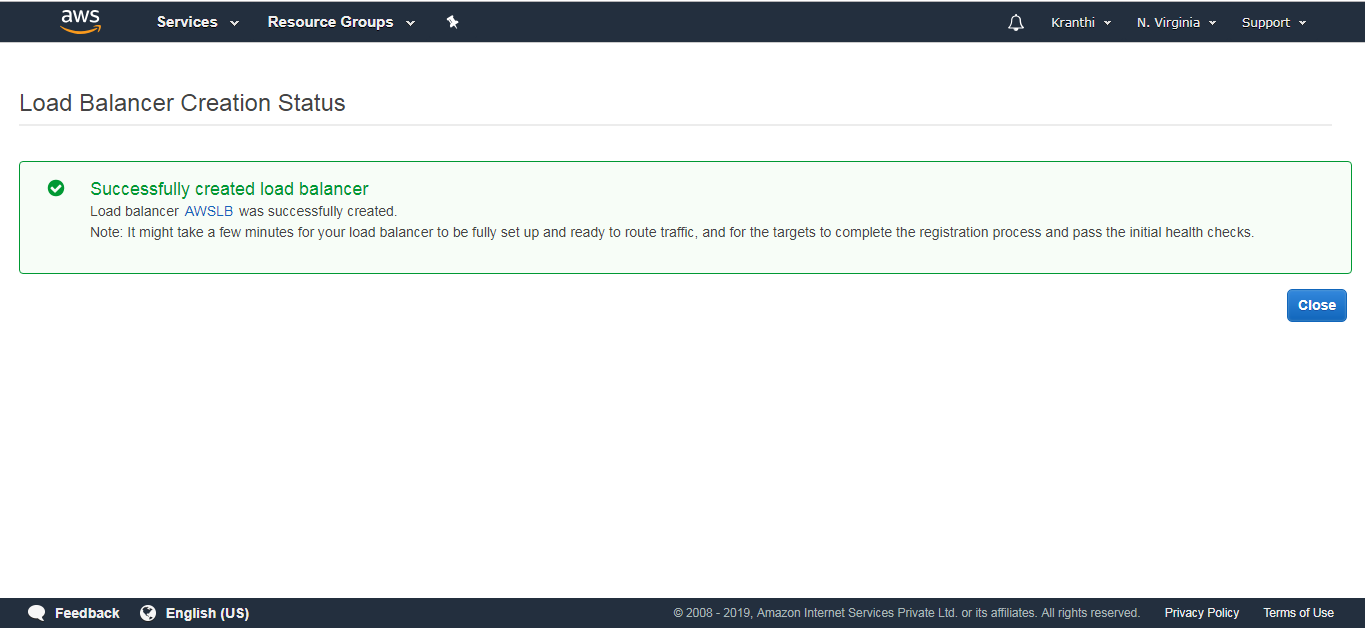




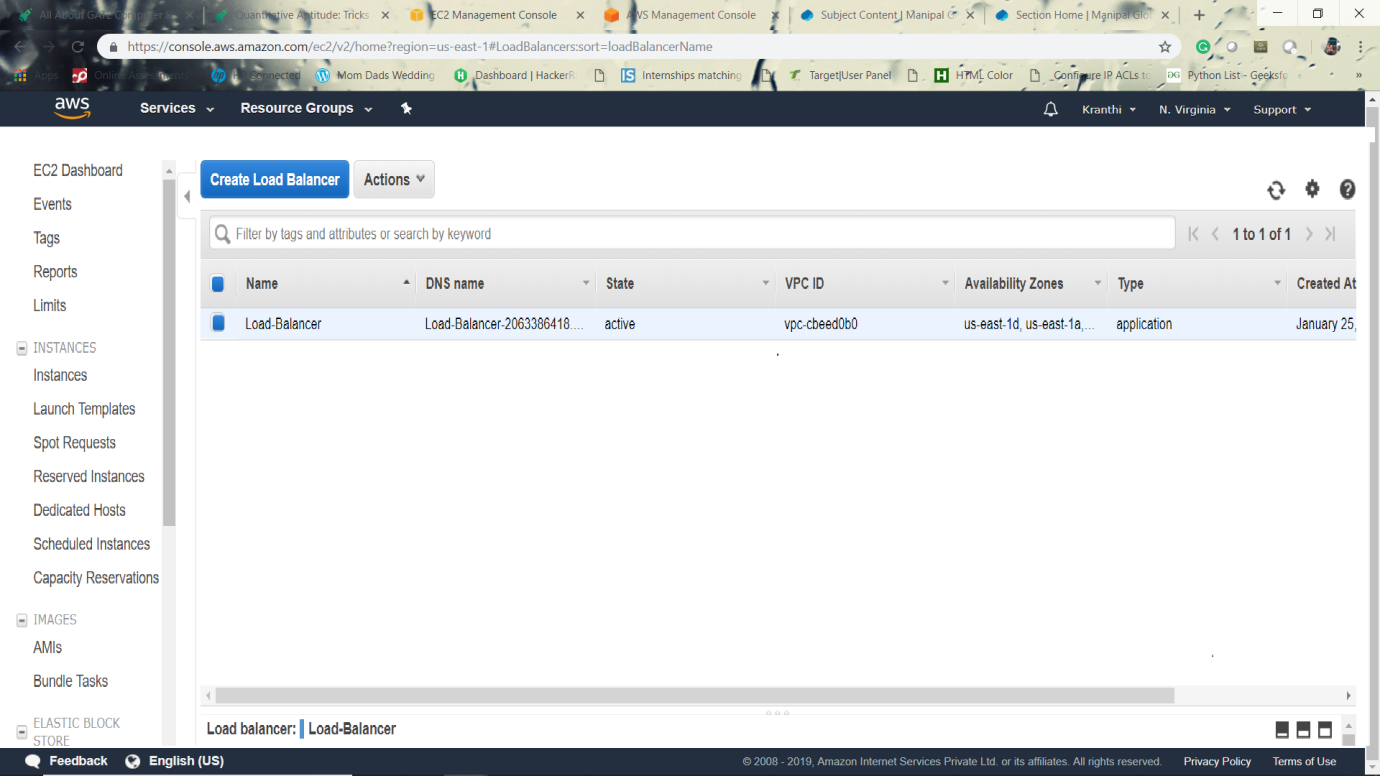
Review: View all the information of an Elastic Load Balancer.



Step 9: We can now notice the notification that Load Balancer is created.



Step 10: This is the Load Balancer that we have created.



Usage of Load balancer:

