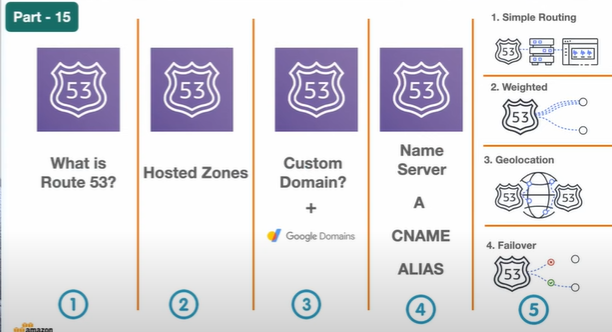
**ROUTE 53**

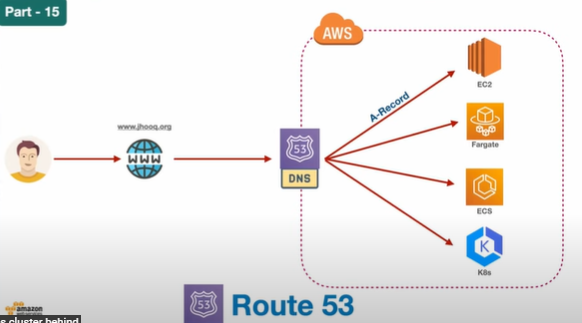
* **What is Route 53?**
* **Hosted Zones**
* **Custom Domain?**
* **Name Server, A record, CNAME, ALIAS**
* **1. Simple Routing**
* **2. Weighted Routing**
* **3. Geolocation Routing**
* **4. Failover Routing**

****

* **What is Route53**

**When ever a user try to a website URL , here its try to enter a web site** [**https://loadec2.shop/**](https://loadec2.shop/) **as soon as the user enter the URL the domain go’s to the domain name system which is our Route53. So as soon as the request lands on our Route53 it tries to resolve that particular request’s based on the name server settings.**

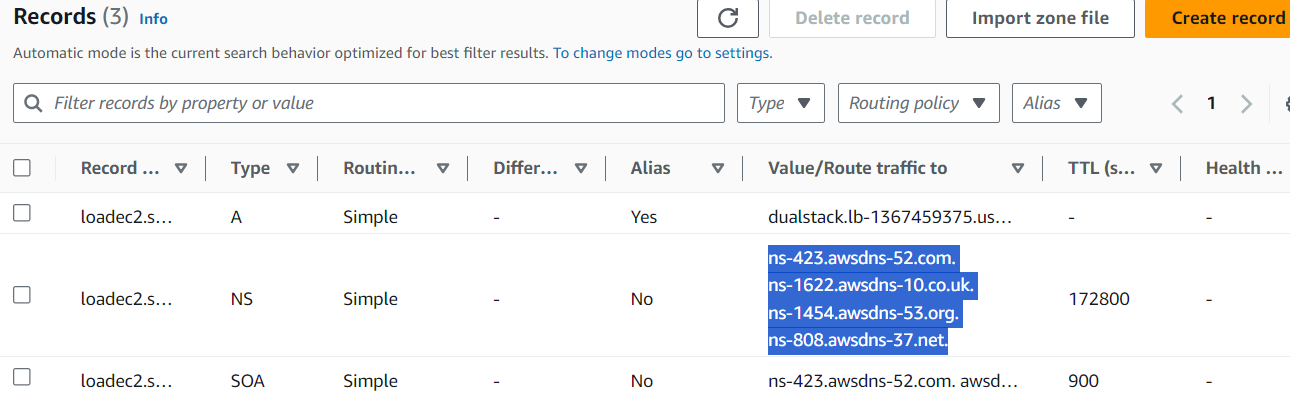
* **once it resolve that request is forwarded to particular resource which has been configured into our AWS environment so that request forwarding is happening with the a record so we configure two things into Route53 which is the name server record for resolving the domain then the A record or the c name record for forwarding the request to the particular resource so here I’m taking an ec2 instance but in AWS you can have a multiple instances or multiple different resources running behind the scenes so you can serve the same request either with EC2 instance or either with the Fargate or you can also serve it with the ECS services or also you can run Kubernetes cluster behind the scene.**

****

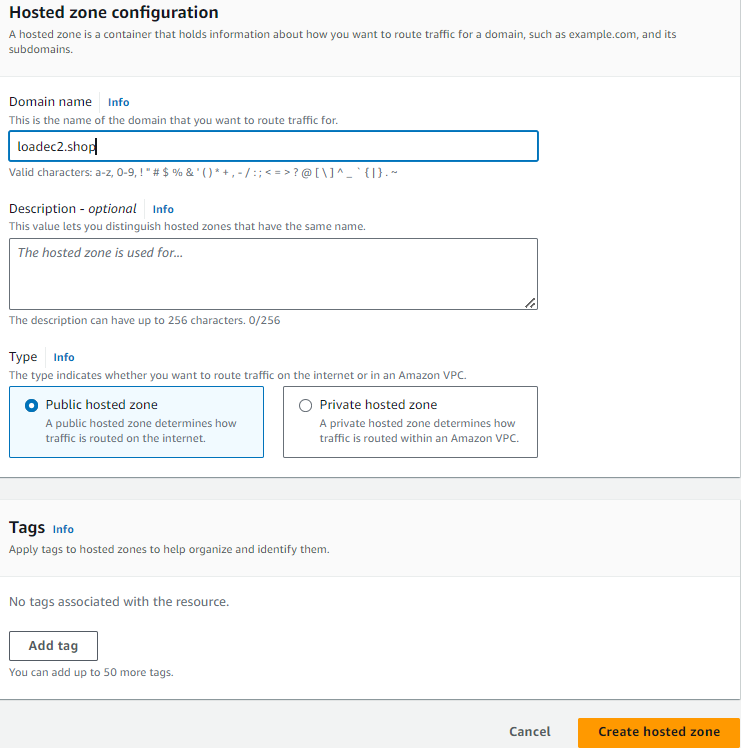
* **You need to set our A record or C name properly so that your request will be served back properly**

**Setting of Route 53**

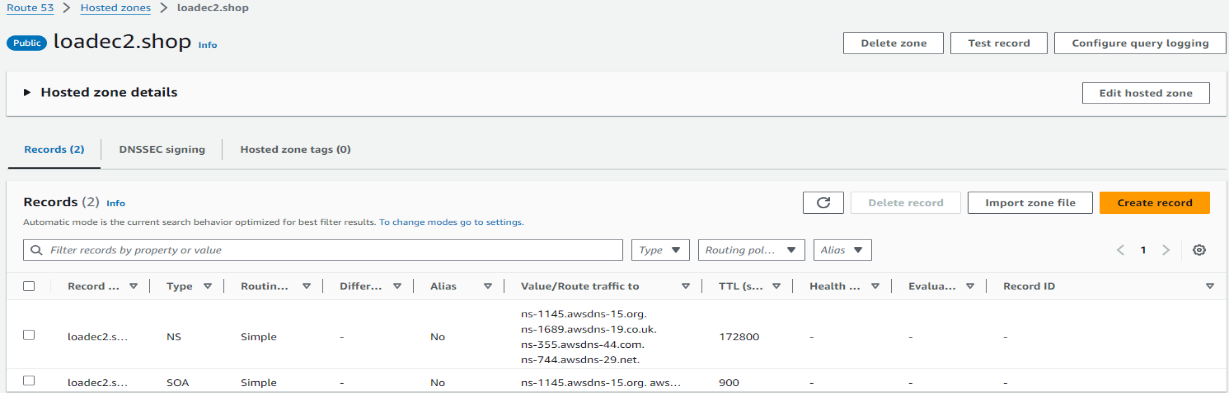
* **Before we proceed I would like to show you the settings which is of our Route53hosted Zone**

****

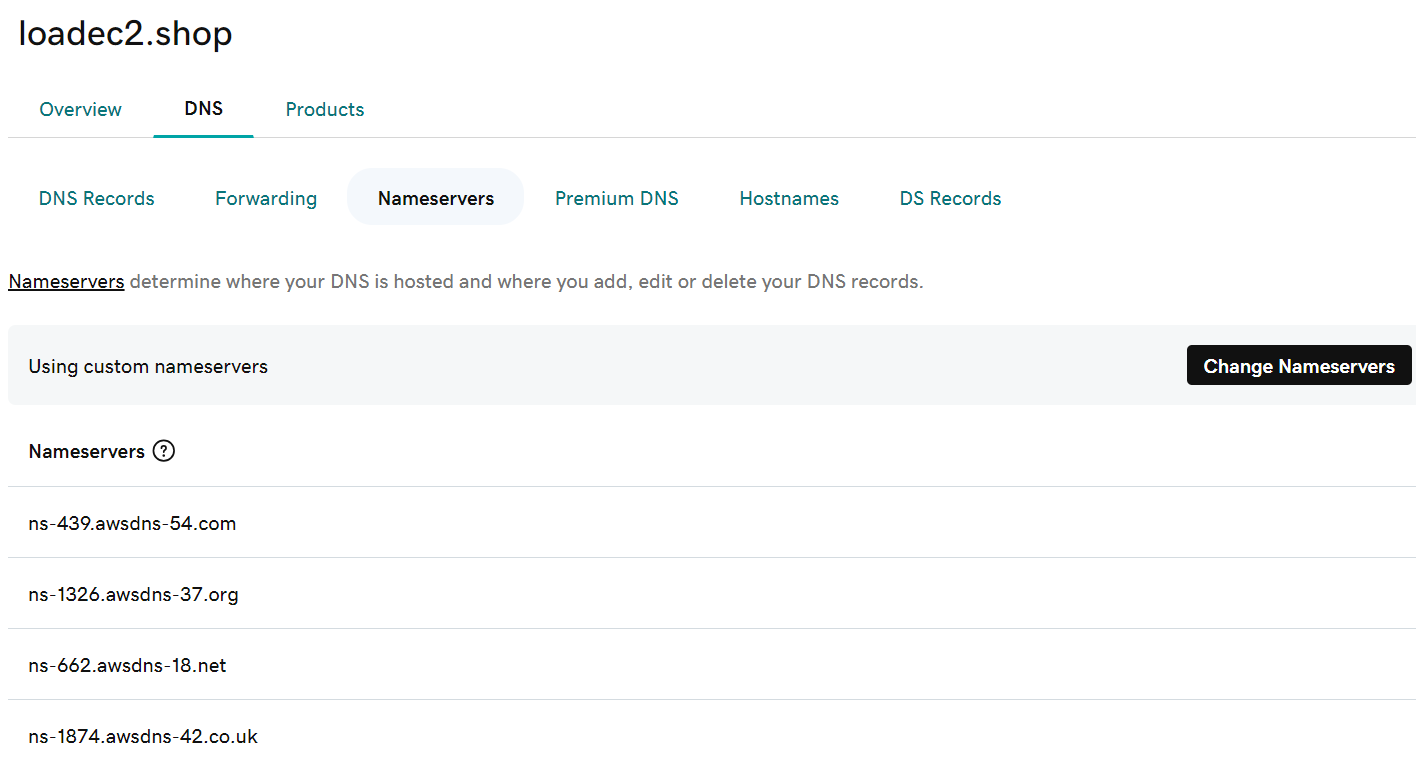
* **So here you can see this is my loadec2.sh hosted zone which I have created into Route53 which you can see over here and these are the settings which own Domain which is loadec2.shop so here you can see there are three types of routing A record and the NS as nameserver also the SOA but we are more consider for nameserver record is responsible for resolving your domain so whenever a user center loadec2.shop then these name server settings will help is to resolve the request which is our given by AWS and this is the A record which will help us to redirect the request to either ec2 instance to farget service to lambda to API Gateway or any other resource so this A record we will be creating multiple time during the demo so this is how our hostage Zone look like now a little bit more knowledgeable about Route53 so the first thing is to create our Route53 hosted Zone.**

****

* **We have to select the public hosted zone to point our public domain to this particular Route53 name server settings so that’s why I’m keeping this as a public hosted zone after that click on create hosted zone as soon as it will be ready**

****

* **These are the name server system settings which are available from the AWS Route53 and these are really important to point your domain 🡪I’ll show you like how you like how I have pointed my own domain with these settings, first settings which you ness to keep in mind which is a name server setting and SOA is not going to pretty much but we will be focusing on the name server.**
* **I will show the Domain which I have purchased this particular domain so this are the first we need to purchase a domain and need to create a hosted zone for that particular domain in AWS.So the second part which we have already created the hosted zone for that Domain. You can purchase the domain from godaddy.com available in the market so you can just purchase the domain from those services all right.**
* **Then we need to configure the domain between AWS Route53 & DNS record because if you misconfigure these records then domain name settings will not work.**

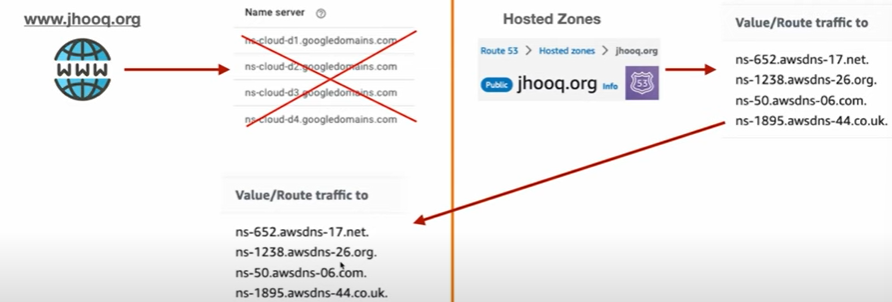
****

* **Here we have the domain default domain name servers in the Godaddy.com, this are the name server settings which we have to change by the Route53 nameserver when you are trying to host somewhere but we are using the AWS.**
* **AWS is provides their own name server settings. Now you will be using the name server setting of your AWS not from the google domain, godaddy.com etc**

****

* **We have to copy this name servers to the domain DNS nameservers settings go to the godaddy.com change the nameserver settings in the godaddy.com DNS name server settings, we don’t need to use the nameservers change according to the Route53**

**Custom Domain**

****

**1.Simple Routing —Route53**

**Setup A record**

* **Now we have configured now that will point our request to the Route53 from Route53 we will create A record to our EC2 instance so that’s going to be our first simple demo for this particular concept now**
* **First create an ec2 instance**
* **Now create the EC2 instance**
* **AMI – UBUNTU, instance type-t2.micro, allow HTTP in the security group, next check to the enable the public IP**
* **Add the user data to install the APACHE in the server after rebooting**

**#!/bin/bash**

**yes| sudo apt update**

**yes| sudo apt install apache2**

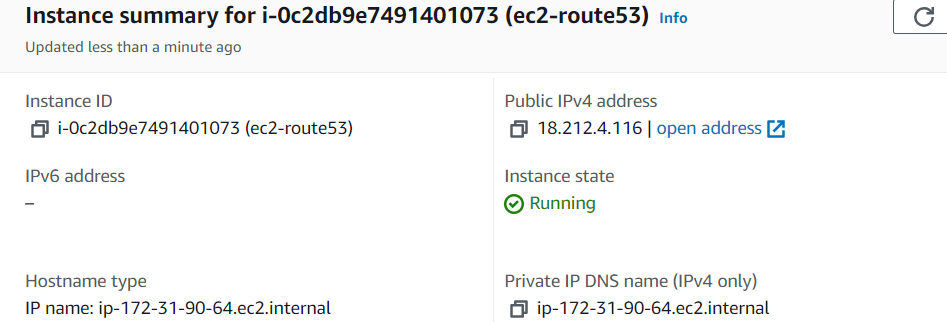
**echo "<h1>server Details</h1><p><strong>Hostname:</strong> $(hostname)**

**</p><p><strong>IP Address:</strong> $(hostname -l | cut -d" " -f1)</p>">**

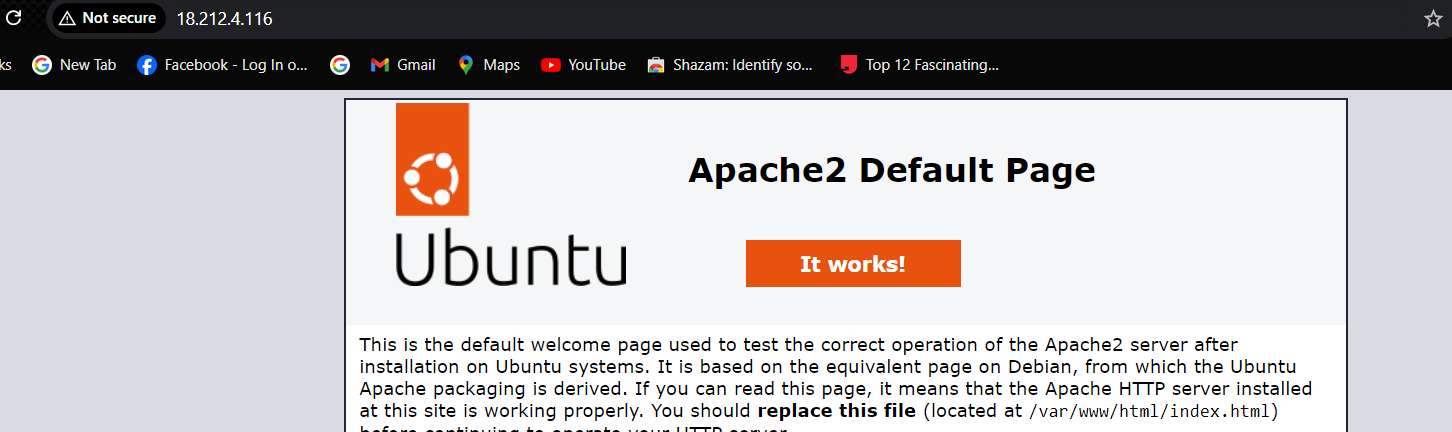
**/var/www/html/index.html**

**sudo systemctl restart apache2**

* **Lunch instance (ec2-route53)**

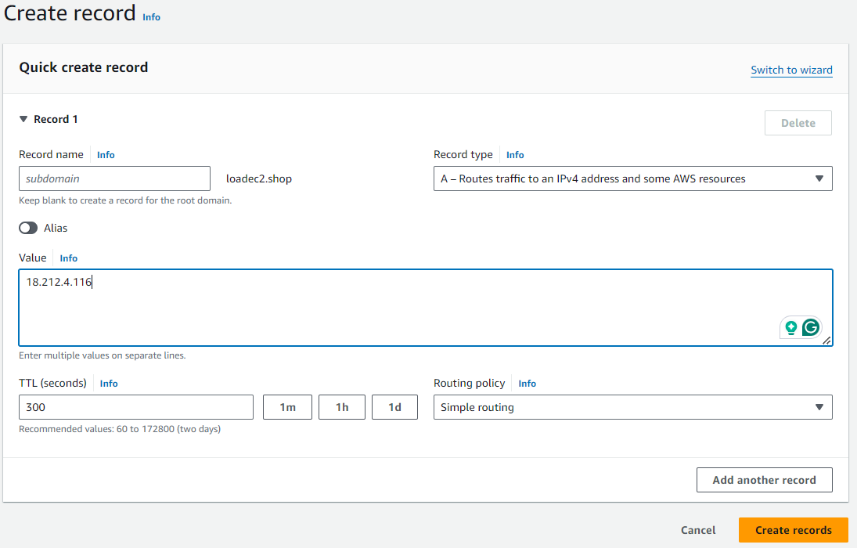
****

* **Now copy the public IP of the instance and past it in the browser to check the default apache page**

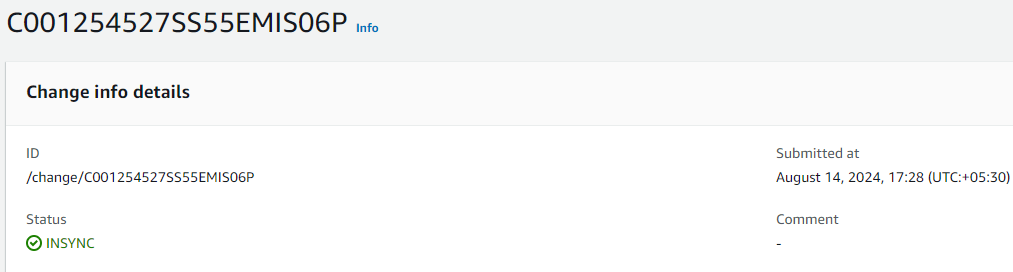
****

* **I don’t want to access my public IP I want to access it with my loadec2.shop for that we have to go Route53 hosted zone create the A-Record to point to the particular ec2 instance.**
* **For that next open the Route53 click on the hosted zone create record**

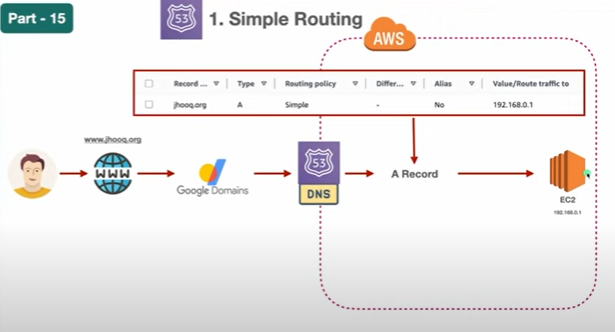
**Copy the public ip of the ec2 instance past it in the value click on create record**

****

* **Wait until the A record is in the force**

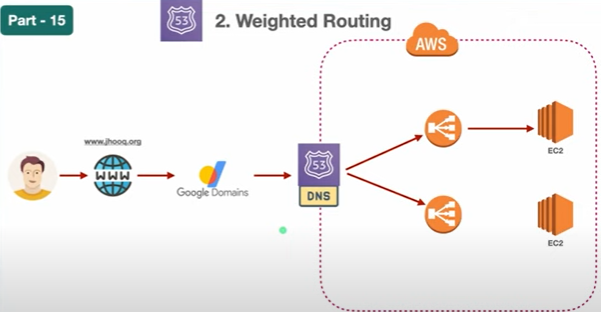
****

* **Now we have created our A-Record which is pointing to the public ip of the EC2 instance after check either the loadec2.shop it showing the apache page till now we have done Simple Routing.**

****

* **Its no a static IP address to fix this we need to use the load balancer so the we are hiding our resources under the load balancer so that we don’t need to provide the static IP address in those cases**

**2.Weighted Routing — Route53**

****

* **The workflow which is quite similar to the one which we have seen previously so here this is a user which is trying to access the website which is the loadec2.shop it going to the loadec2.shop which is going to the godaddy domain.**
* **And the from the godaddy domain it finding the name server record for our Rouet53 and the request is coming to the route53 hosted zone**
* **Once that request enter into our hosted zone of our AWS Route53 then it goes to the load balancer and it forward to the ec2 instances**
* **Here you can see the couple of load balancer see eventually into the weighted routing but the idea is we need to have a load balancer that points to the EC2 instance. So that’s the preferred way using the Route53 with the load balancer and hiding all the resource behind the load balancer.**

**🡪Demo:setup Networking**

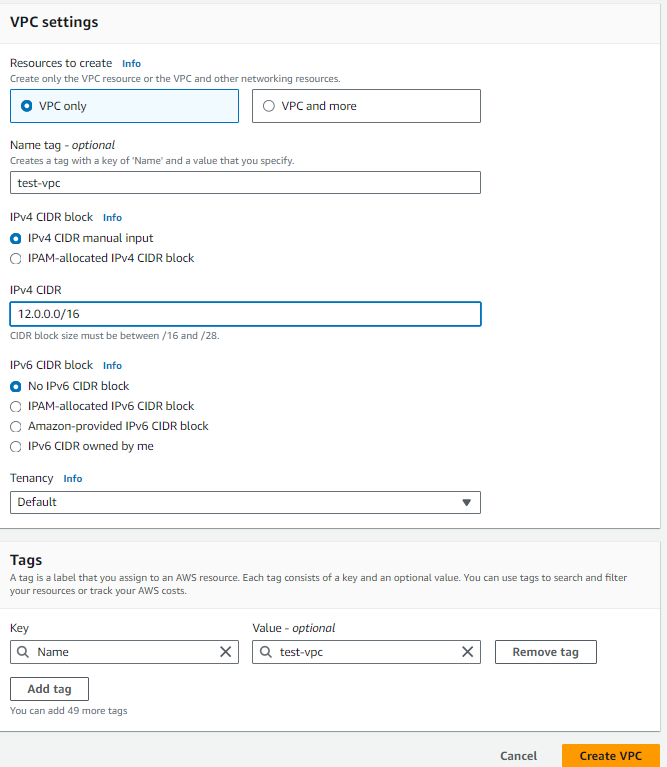
* **All right you have an idea like we need to use a load balancer so that we don’t need to hard code the IP address for our Route53 A-Record**

****

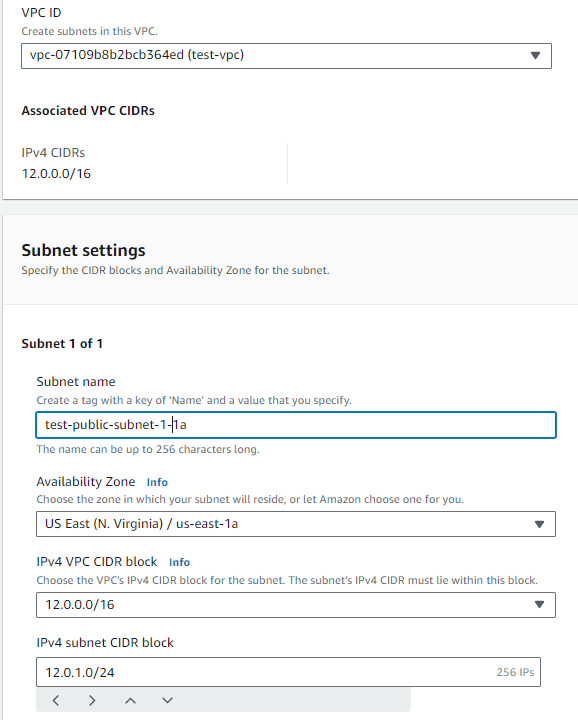
* **Load balancer is one concept which we need to incorporate when our working with Route53 but on top off that you also need to set up your proper networking which means you need to setup VPC although there is a default VPC we aren’t going to use that VPC.**
* **Create our own VPC and inside that VPC we are going to create – public subnet, private subnet, internet Gateway for our public subnet**

**It can access the internet after internet we are going to create a route table and this route table will be responsible for forwarding the request to our public subnet as well as to our private subnet.**

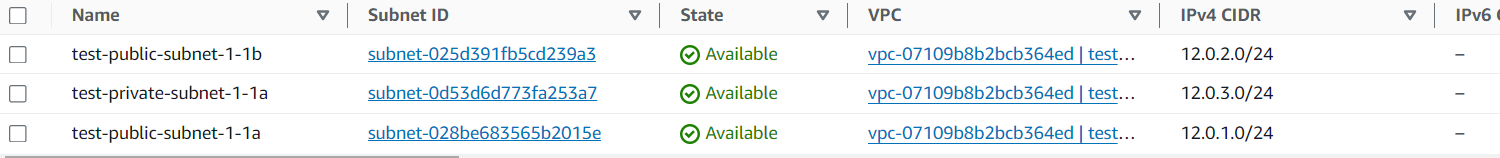
* **Once we configure the route table’s we are going to provision the EC2 instance inside my public subnet as well as my private subnet so this is the minimal network setting you should do so whenever you are working with our Route53 and the load balancer this are most ideal practice which you need to follow to setup this type of environment.**
* **VPC**
* **Public subnet-1a**
* **public subnet -1b**
* **private subnet**
* **Internet gateway**
* **Route table**
* **EC2 instance**
* **Once we have done this then we are going to provision the**
* **Load balancer**
* **Then we are going to point the load balancer to our route53**

****

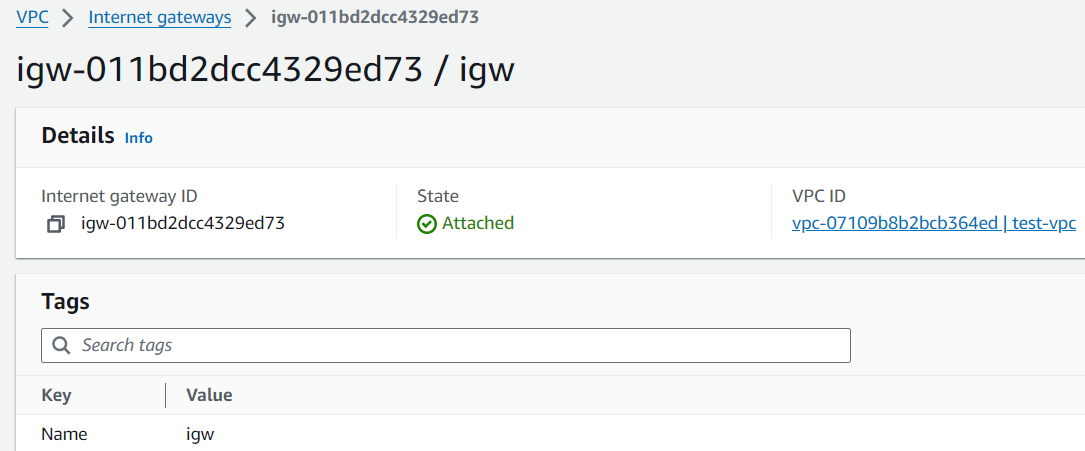
* **Click on create VPC with the VPConly –12.0.0.0/16 (test-vpc)**
* **Next create a public subnet with the test-vpc – 12.0.1.0/24 (test-public-1-1a)**

****

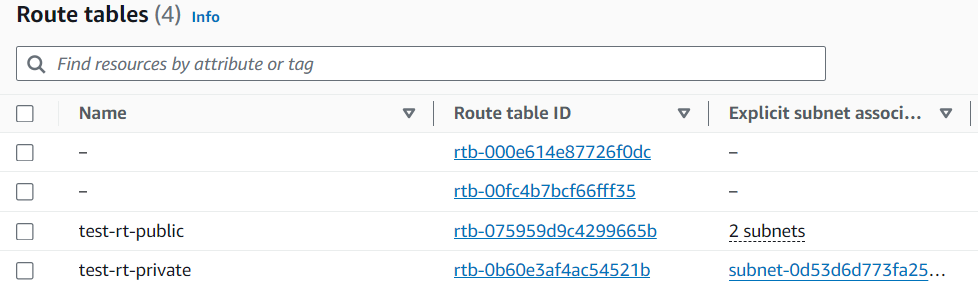
* **Create a second public subnet for the maximum availability**
* **First public subnet -12.0.1.0/24 (test-public-1-1a)**
* **Second public subnet -12.0.2.0/24 (test-public-1-1b)**
* **Private subnet -12.0.3.0/24 (test-private-1-1a)**



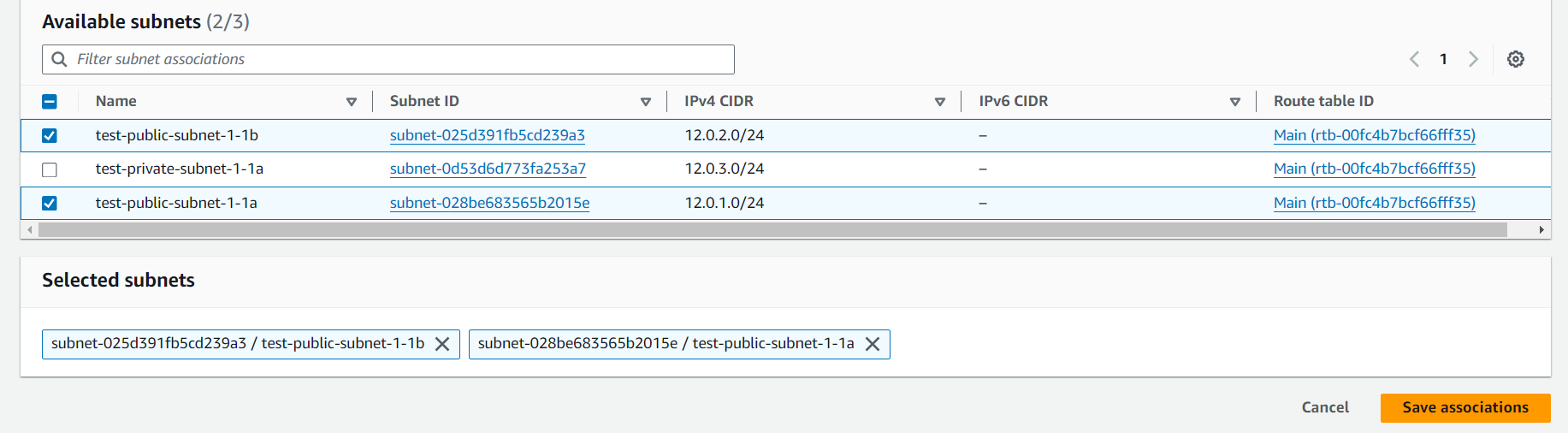
* **Next create the internet gateway (igw) and attach the (test-vpc)**



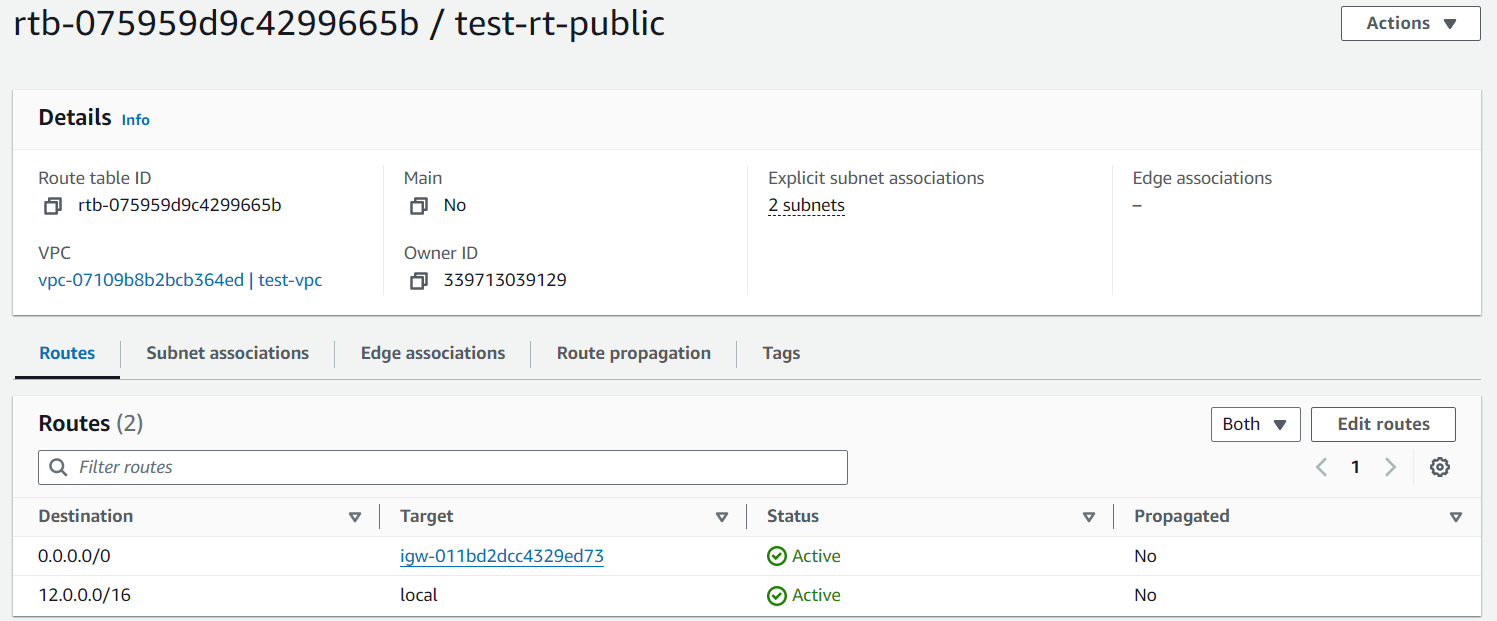
* **Now create the public route table & and the private route table and do the public subnet association 1a and 1b to the public route table with the internet gateway (igw) and the private subnet to the private route table**

****

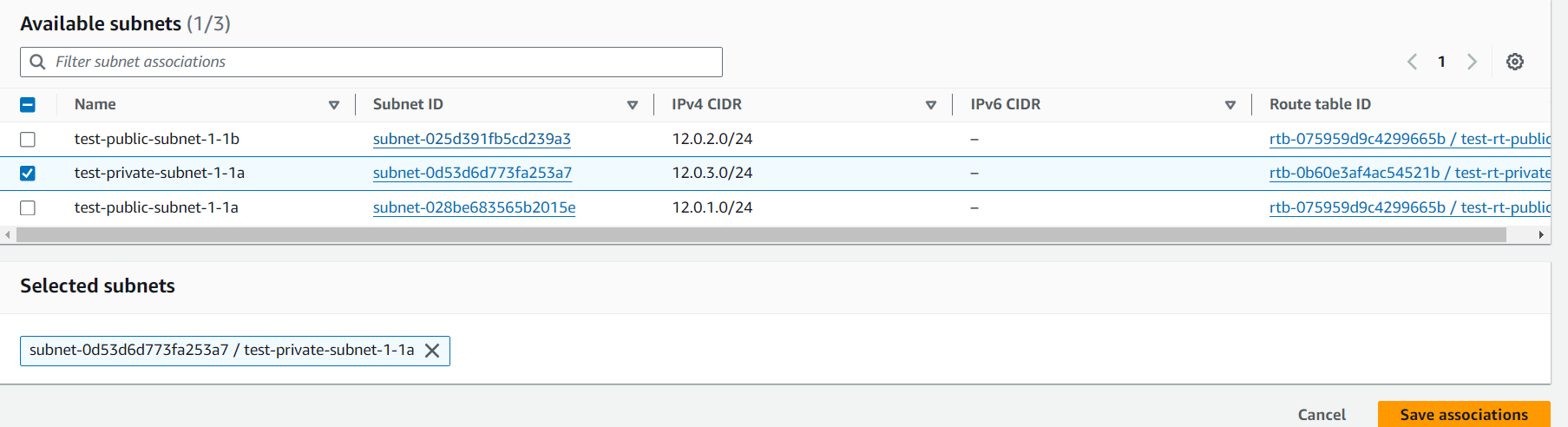
* **Subnet association for the public route table**



* **And click on save associations and next click on edit routes and add the internet gateway (igw) to the route table for the internet access from- any where**

****

* **Also do the private subnet association for the private route table**



* **Click on save subnet associations and next we are going to create an EC2 instance**
* **Creating ec2 instance – (test-ec2-instance)**

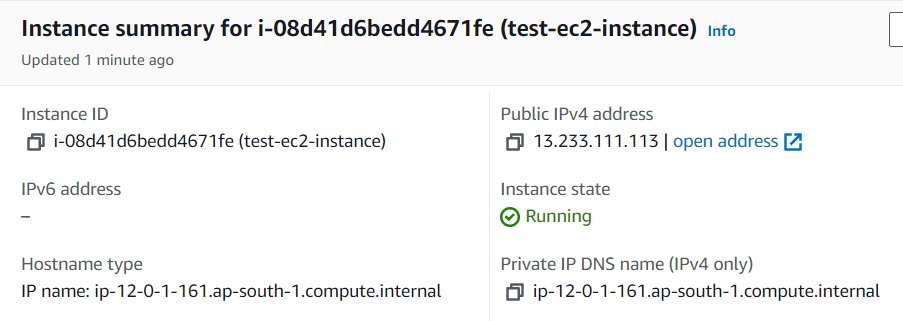
* **AMI-UBUNTU**
* **Instance type – t2.micro**
* **Add the VPC – (test-vpc)**
* **Add the test-public-subnet-1-1a**
* **And add the security group HTTP – port number -80 – 0.0.0.0/0 (any -where)**
* **Add the user data to install nginx after rebooting ec2 instance.**

**#!/bin/bash**

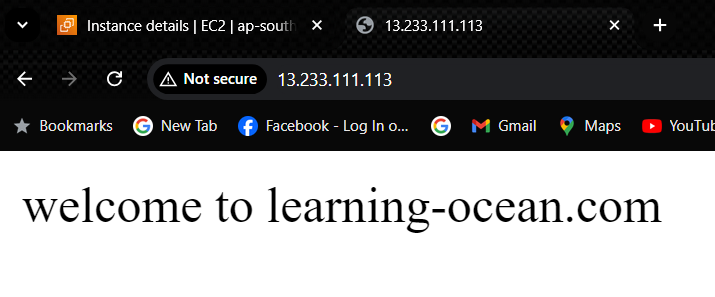
**apt-get update**

**apt-get install nginx -y**

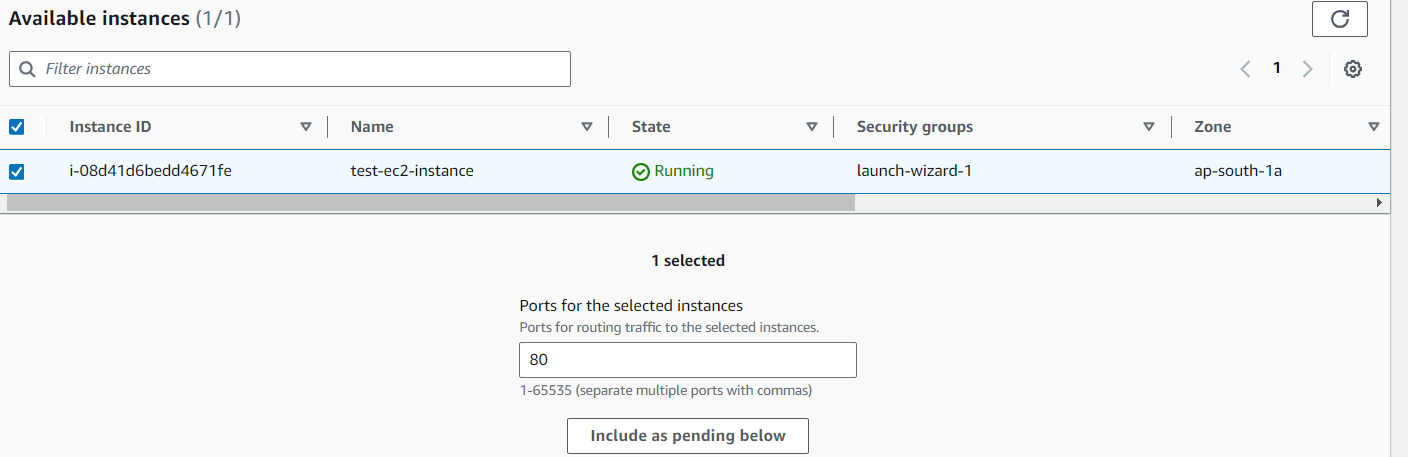
**echo "welcome to learning-ocean.com" > /var/www/html/index.html**

****

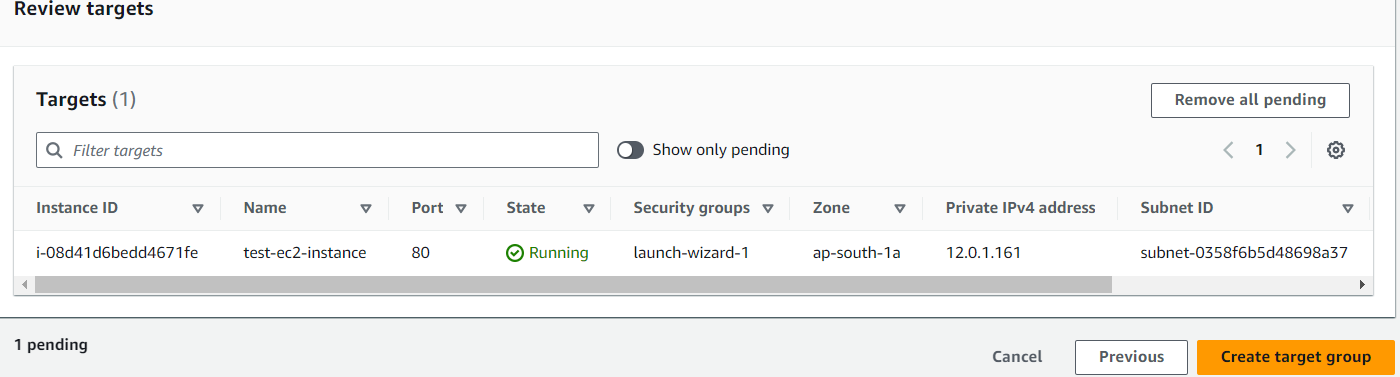
* **Next click on lunch instance and copy the public IP of the instance and past it in the browser ex: 13.233.111.113.**

****

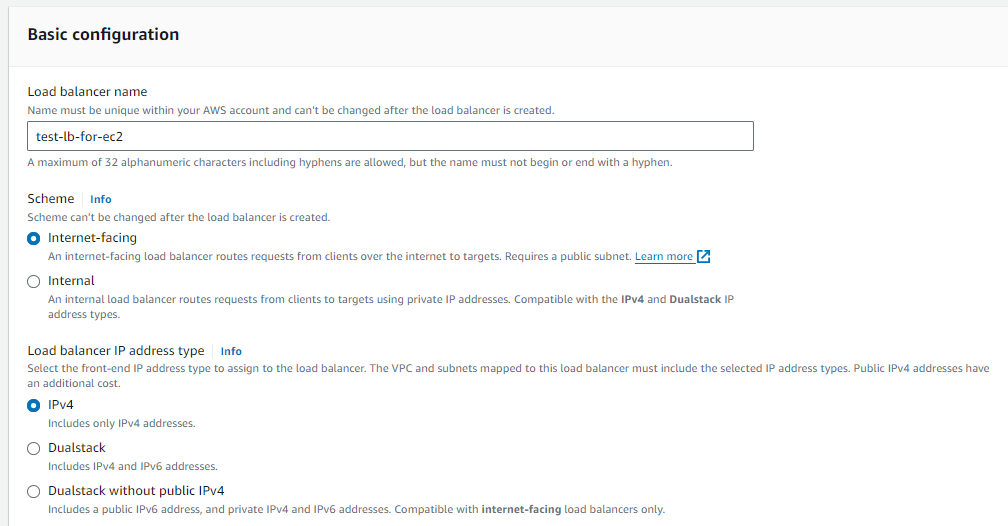
* **Now we need to configure the load balancer for this EC2 instance so we don’t need to access the public IP of the instance.**
* **For creating load balancer first we need to create a target group.**
* **Target group**
* **Target type: instance**
* **Target group name: test-tg-ec2**
* **Protocol: port**
* **HTTP 80**
* **IP address type: IPV4**
* **VPC- (test-vpc)**
* **And click on next**



* **Select the instance that is to be targeted and click on include as pending below in the above picture**



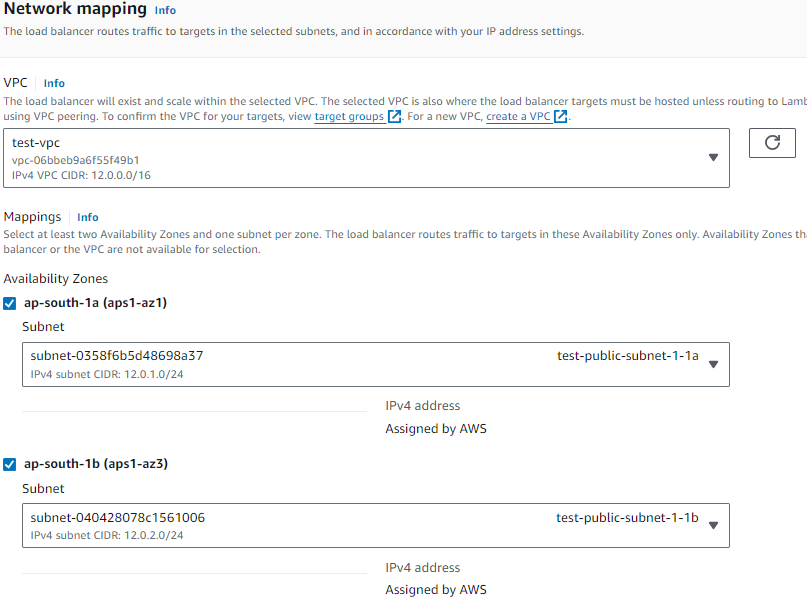
* **Next click on create target group & now we need to configure the load balancer which is the Application load balancer**
* **Load balancer**
* **Application Load Balancer**
* **Load balancer name: test-lb-for-ec2**
* **Scheme: Internet facing**
* **IP address type: IPV4**

****

* **Networking mapping**

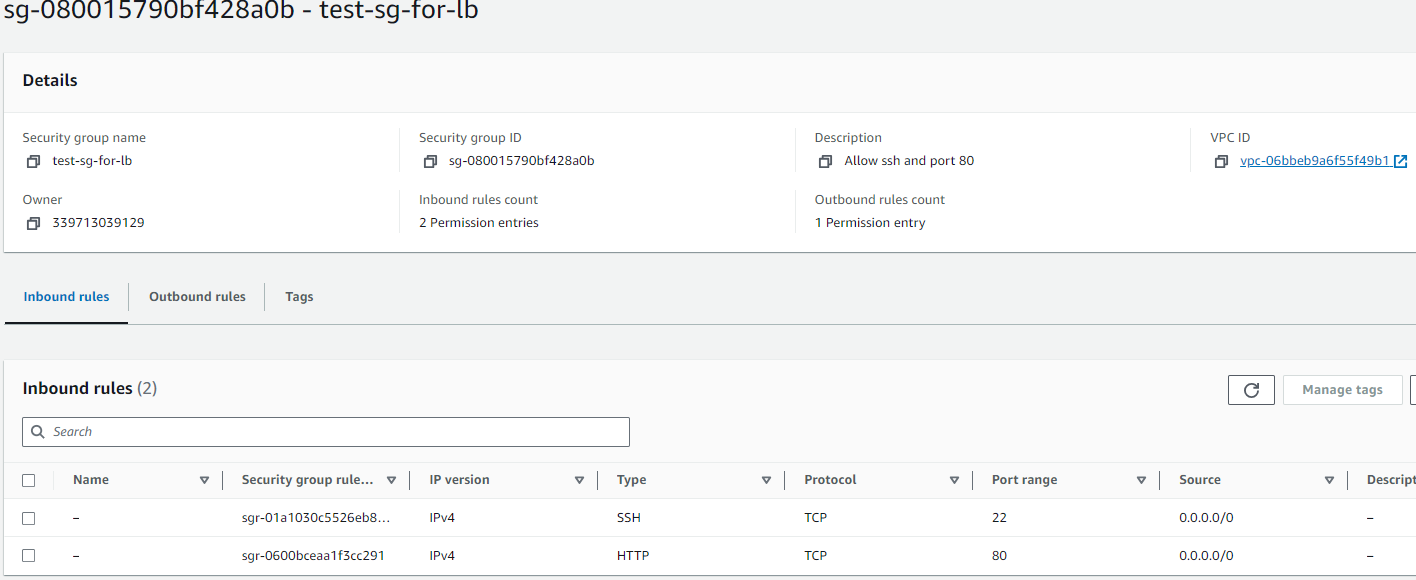
**VPC – (test-vpc)**

**Mapping: test-public-subnet-1-1a & test-public-1-1b**

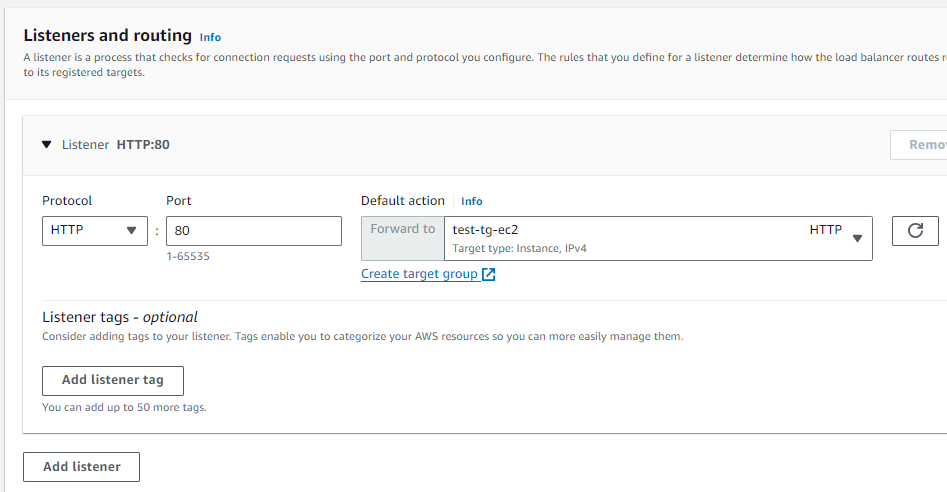
****

**Security group: create a new security (test-sg-for-lb)**

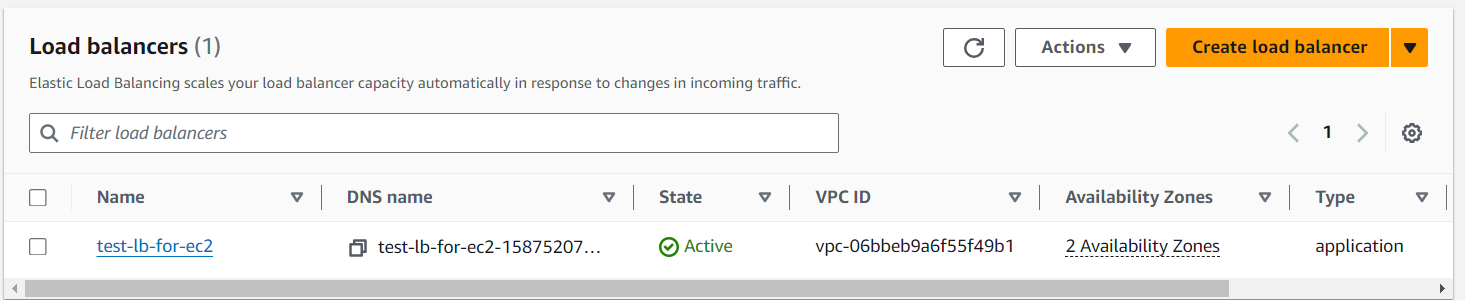
* **Description: Allow ssh and port 80**
* **VPC:-(test-vpc)**
* **Add rules: ssh, Anywhere-IPv4 & HTTP, Anywhere-IPv4 & click on create security group**

****

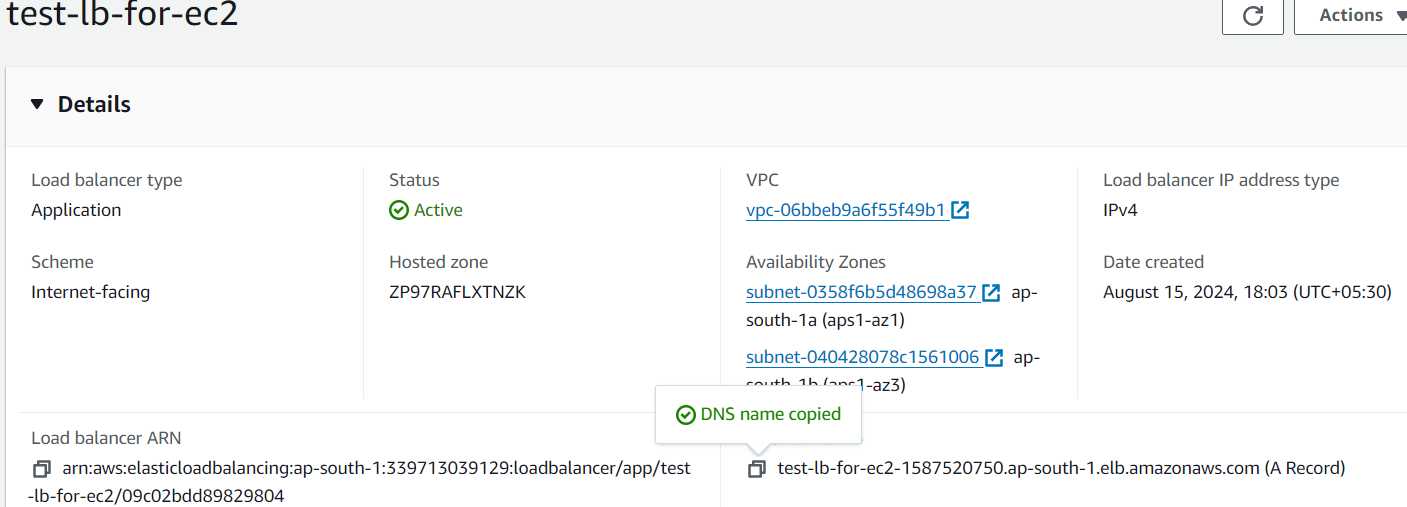
* **Add target group in the Listeners and routing**

****

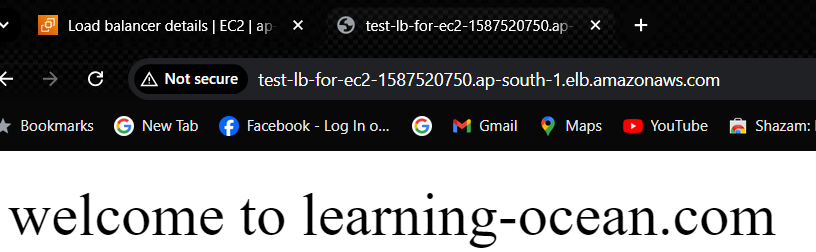
* **Next click on create load balancer wait until its active**

****

* **Now open the load balancer and copy the DNS name**

****

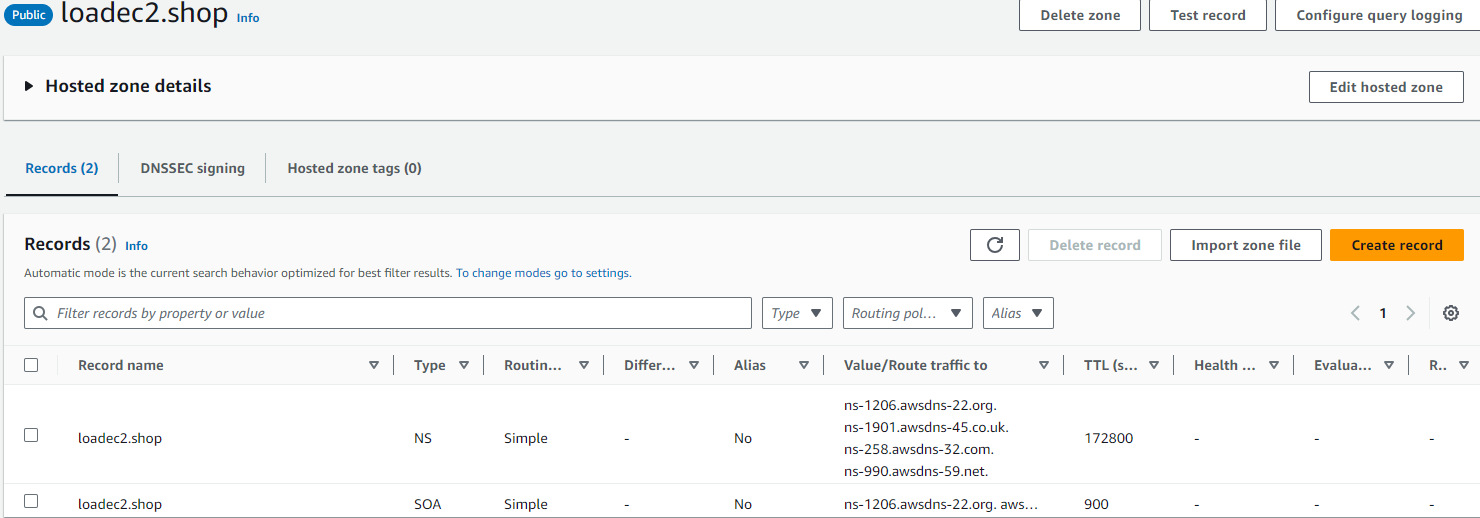
* **Open the browser and past the DNS name it hits to the “welcome to learning-ocen.com” in ec2 instance directly**

****

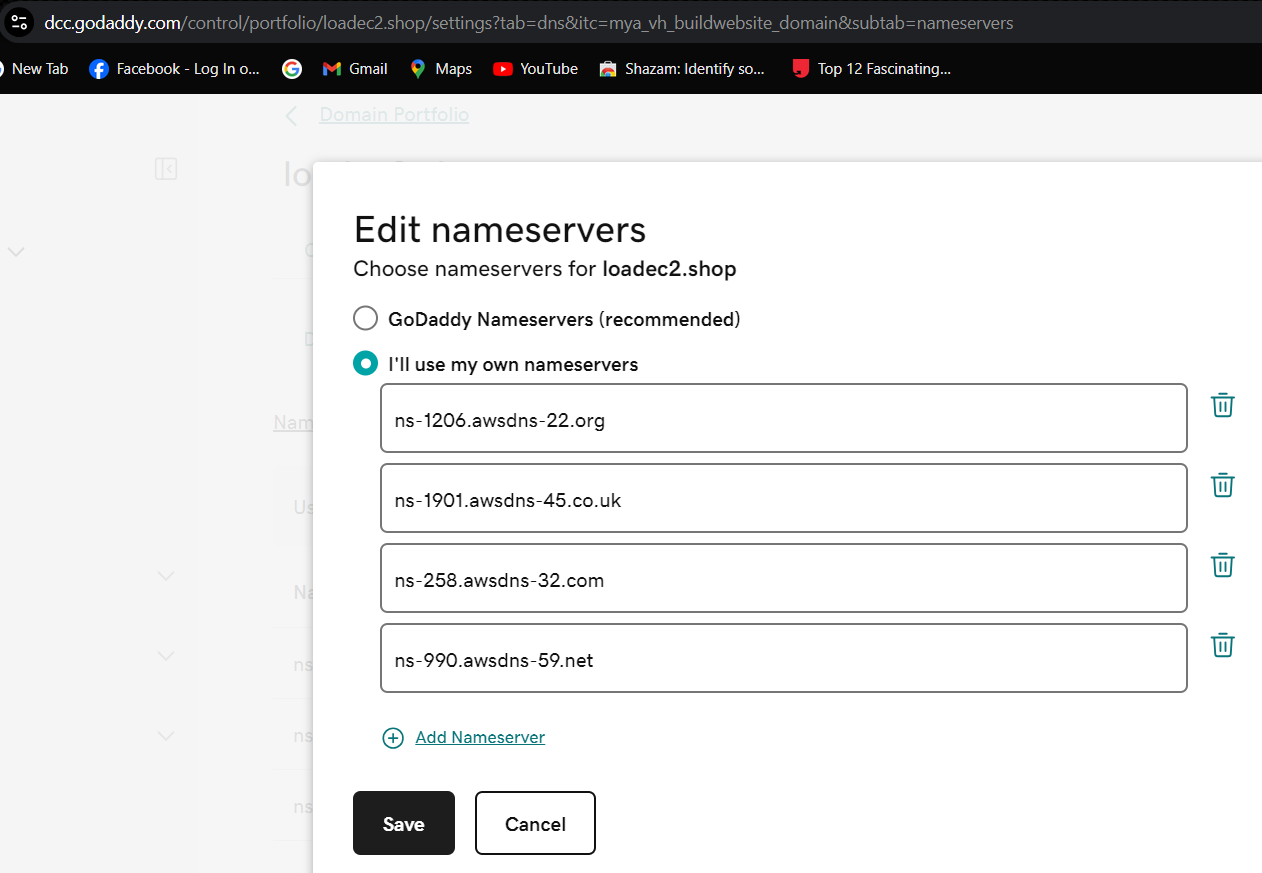
* **Now we have to do the linking between Route53 and the load balancer**

**We linked ec2 instance with the load balancer so we are able to access our ec2 instance with load balancer but we need to create an A-Record so that our Route 53 redirect the request to our load balancer will forward the request to ec2 instance**

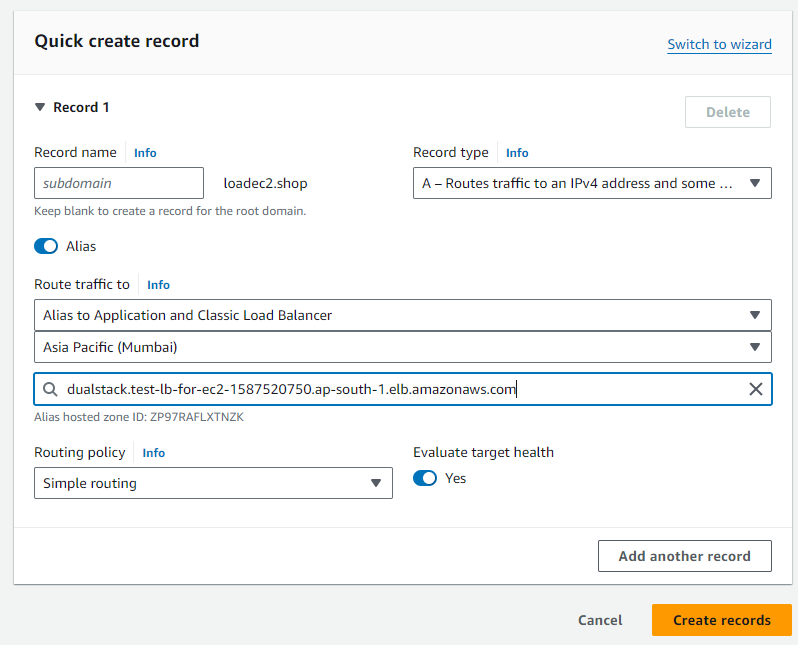
* **Route53**
* **create Hosted zone – loadec2.shop**
* **Type – Public hosted Zone**
* **Next click on create hosted zone**

****

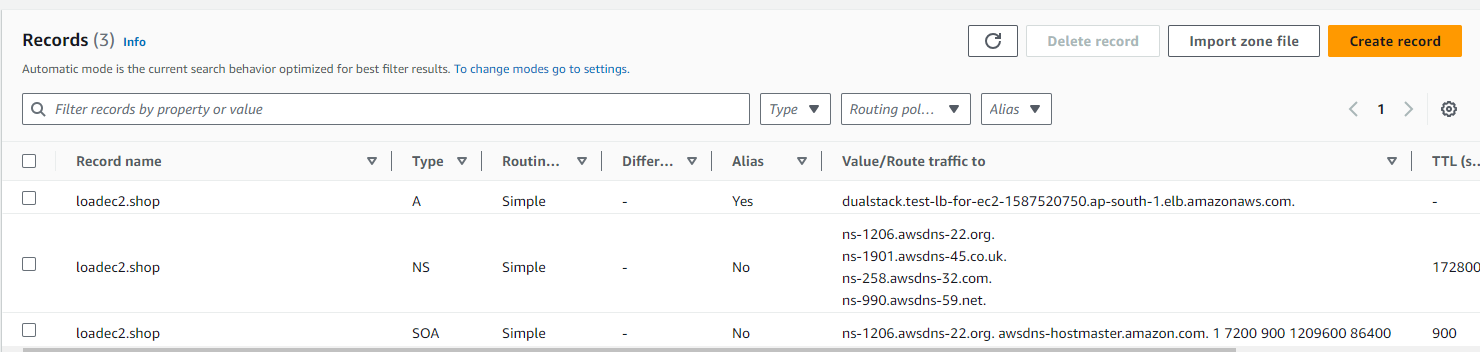
* **Next copy and past the nameservers from Route53 to our Domain DNS settings in godaddy.com, googledomain.com ect.**

****

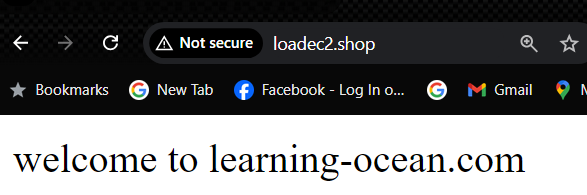
* **Save and continue the changes in our domain portfolio**
* **Create the A-Record with enabling Ailas-application load balancer, Region, and select the load balancer & click on create record**

****

* **The A-Record is directly pointing to our load balancer**

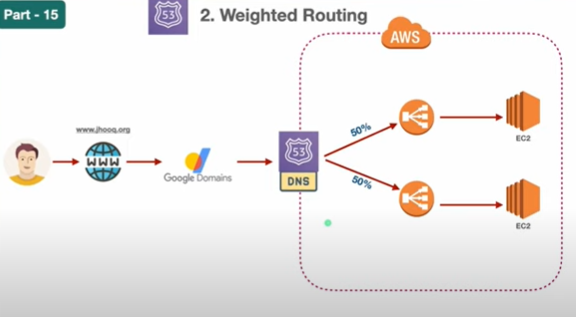


* **By using loadec2.shop now it should point me directly to “welcome to learning-ocean.com” in the EC2 instance**

****

**Concept of Weighted Routing:**

* **And now you know like how to work with the A-Record and how to set up your load balancer so that the Route53 can redirect that request to our load balancer and then eventually you will be able to access your ec2 instance.**

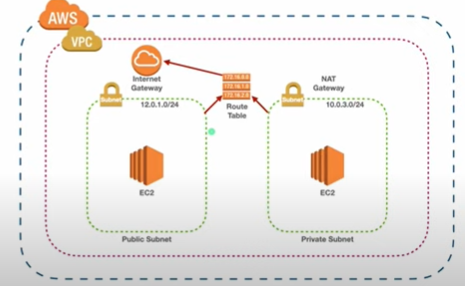
****

* **As the name suggests the weighted routing which means we will be putting a percentage to share the traffic load onto multiple load balancer.**
* **So you have already configure a one load balancer but in the example we are going to configure multiple load balancer so our Route53 will have a multiple load balancer and then it will redirect those request to the EC2 instances.**
* **Now here we can see the first load balancer will forward to the first ec2 instance and the other load balancer will forward that request to another ec2 instance.**

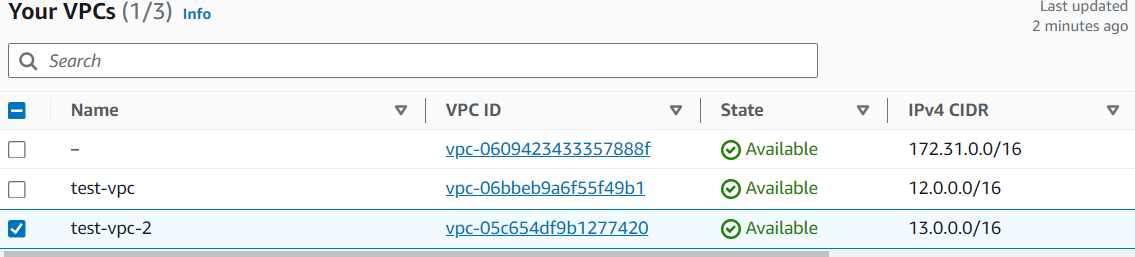
**Second load balancer:**

* **Also we can configure the percentage like how much traffic of percentage should be redirected to the first load balancer and how much it should be redirected to the second load balancer. So that’s what we call it as a weighted Routing.**
* **But before we setup another load balancer I need to set up**

**Another VPC, subnets, internet gateway, route tables:**

****

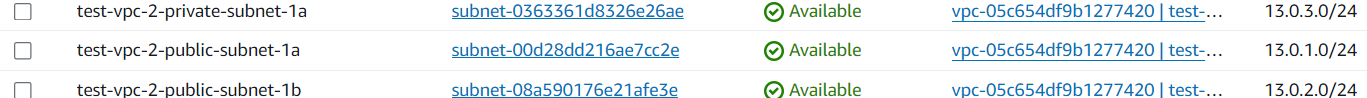
* **So I’m just taking the same diagram once again and I’m not going to repeat the same process again.**
* **Just follow all the sets previously while I am creating the (test-vpc) so I’m just going to use the exact same steps but the only thing which I’m going to change is the IP range so since I’m already using the 12.0.0.0/16 for (test-vpc).**
* **Instead of that I am going to use 13.0.0.0/16 for my test VPC 2. So going to create the whole test VPC, subnet, route table, internet gateway.**
* **VPC- 13.0.0.0/16-(test-vpc-2)**

****

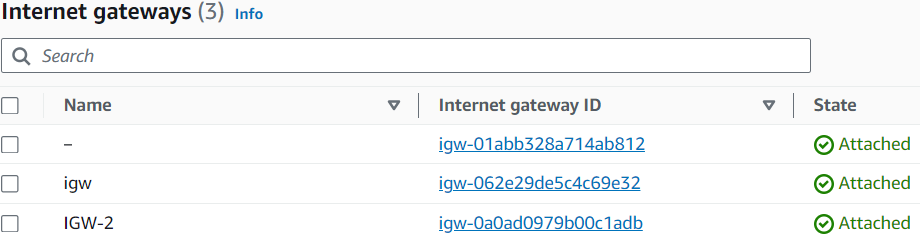
* **Subnet-** **First public subnet -13.0.1.0/24 (test-public-1-1a)**

**Second public subnet -13.0.2.0/24 (test-public-1-1b)**

**private subnet -13.0.3.0/24 (test-private-1-1a)**

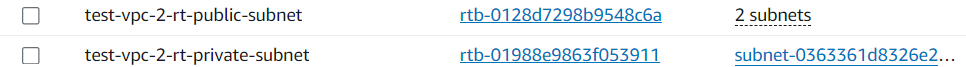


* **internet gateway- IGW-2- Attach to VPC (test-vpc-2)**

****

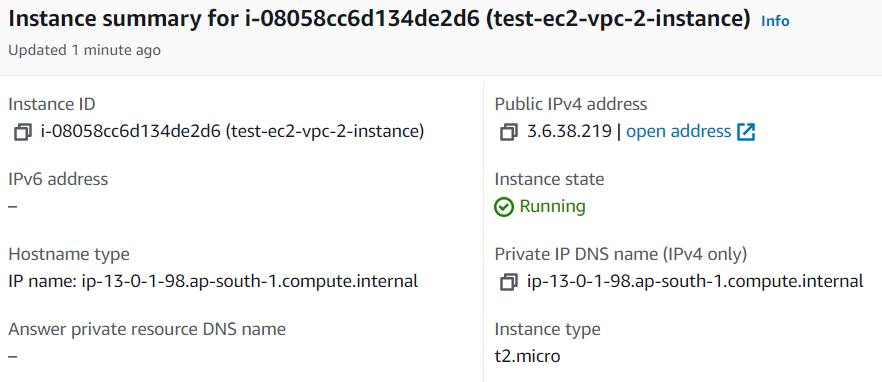
* **Route table: test-vpc-2-rt-public**

**test-vpc-2-rt-private**

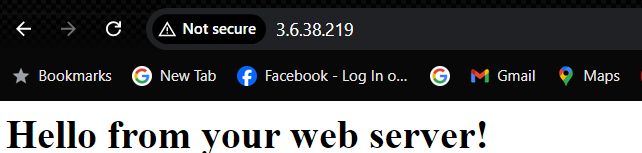


* **Now we are going to create an ec2 instance**
* **Creating ec2 instance – (test-ec2-vpc-2-instance)**
* **AMI-UBUNTU**
* **Instance type – t2.micro**
* **Add the VPC – (test-vpc-2)**
* **Add the test-vpc-2public-subnet-1-1a**
* **And add the security group HTTP – port number -80 – 0.0.0.0/0 ( any -where)**
* **Add the user data to install nginx after rebooting ec2 instance.**

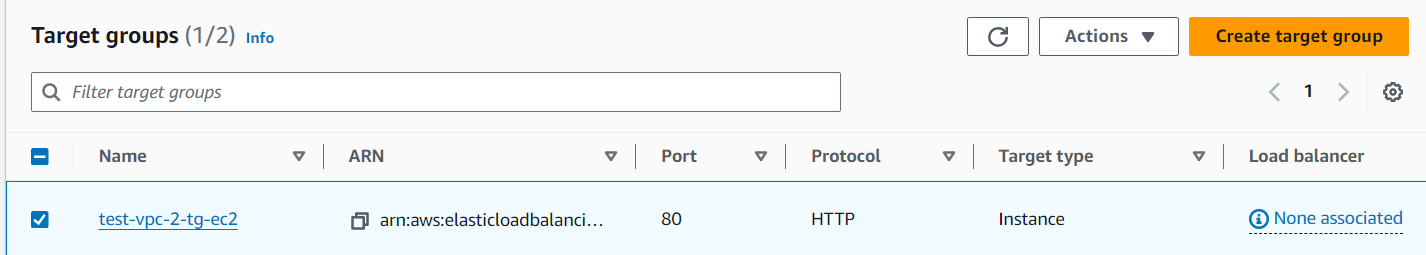
**#!/bin/bash  
  
# UPDATE PACKAGE MANAGER  
apt update --fix-missing  
  
# INSTALL, START and ENABLE NGINX  
apt install -y nginx  
systemctl start nginx  
systemctl enable nginx  
  
# CHANGE FILE PERMISSION TO PERMIT MODIFICATION OF DEFAULT WEB FILE  
chmod 0777 /var/www/html/index.nginx-debian.html  
  
# MODIFY DEFAULT WEB DOCUMENT  
echo "<html><h1>Hello from your web server!</h1></html>" > /var/www/html/index.nginx-debian.html  
# RESTART NGINX  
systemctl start nginx**

****

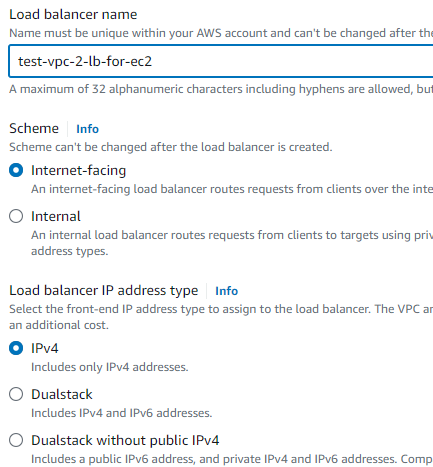
* **Copy the public IP and past the IP in the browser**

****

* **Next my ec2 is up and running so next go and setup load balancer for test-vpc-2-ec2-instance**
* **Target group**
* **Target type: instance**
* **Target group name: test-vpc-2-tg-ec2**
* **Protocol: port**
* **HTTP 80**
* **IP address type: IPV4**
* **VPC- (test-vpc-2)**
* **And click on next**
* **Select the ec2 instance which we have created Click on including as pending and click on create the target group**

****

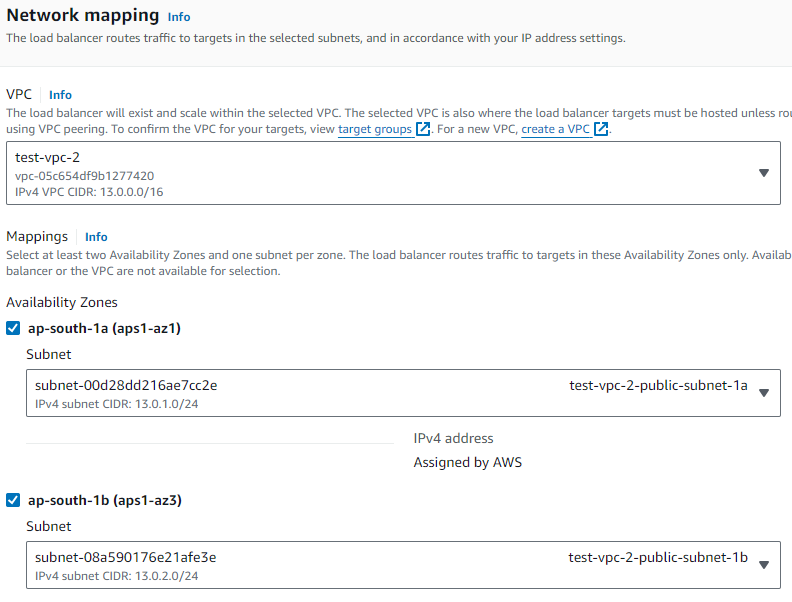
* **Next load balancer**
* **Load balancer**
* **Application Load Balancer**
* **Load balancer name: test-vpc-2-lb-for-ec2**
* **Scheme: Internet facing**
* **IP address type: IPV4**

****

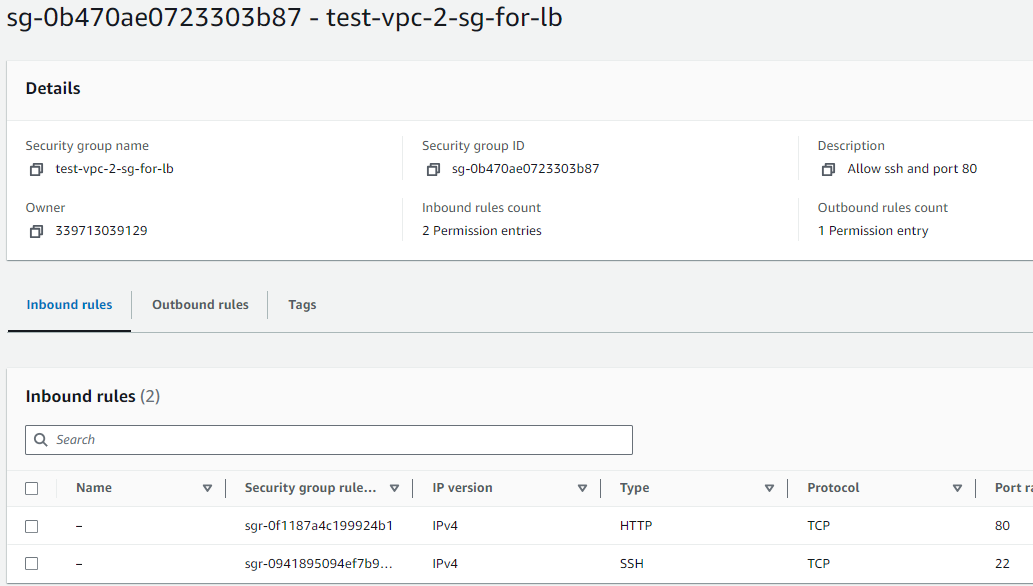
* **Next add the network mapping**

**VPC – (test-vpc-2)**

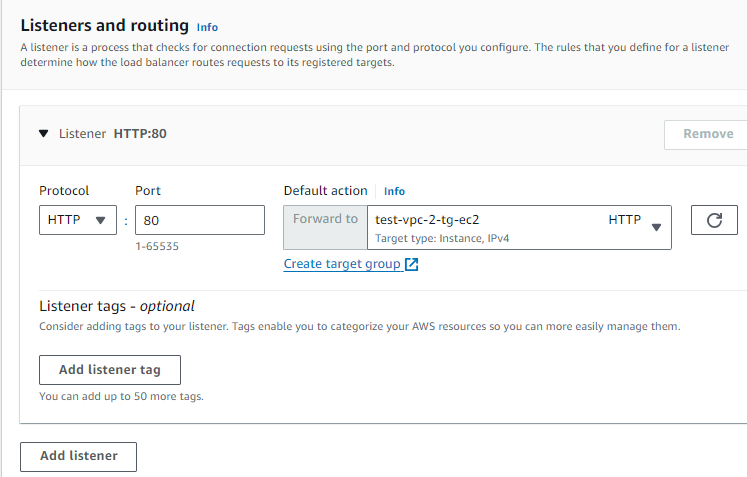
**Mapping: test-vpc-2-public-subnet-1-1a & test-vpc-2-public-1-1b**

****

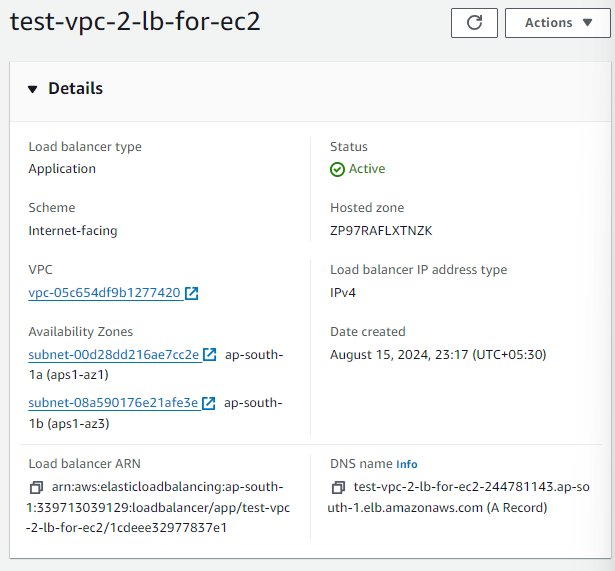
* **Security group: create a new security (****test-vpc-2-sg-for-lb)**
* **Description: Allow ssh and port 80**
* **VPC:-(test-vpc-2)**
* **Add rules: ssh, Anywhere-IPv4 & HTTP, Anywhere-IPv4 & click on create security group**

****

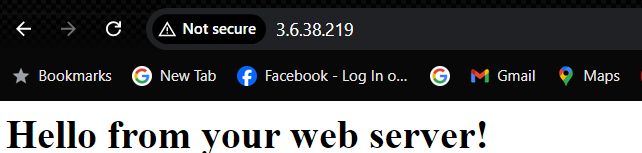
* **Add the target group in the Listeners and routing**

****

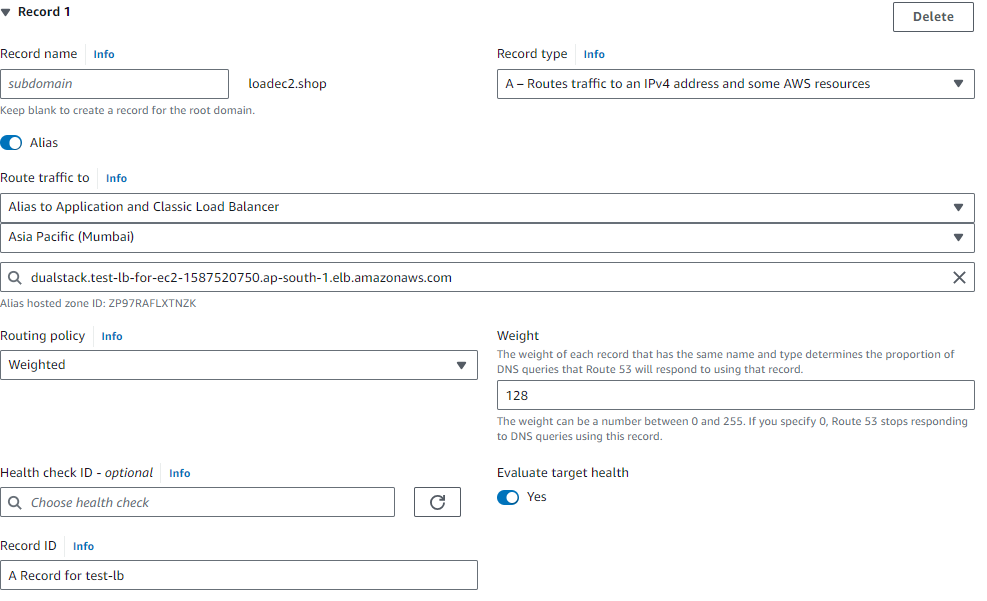
* **Next click on create load balancer**

****

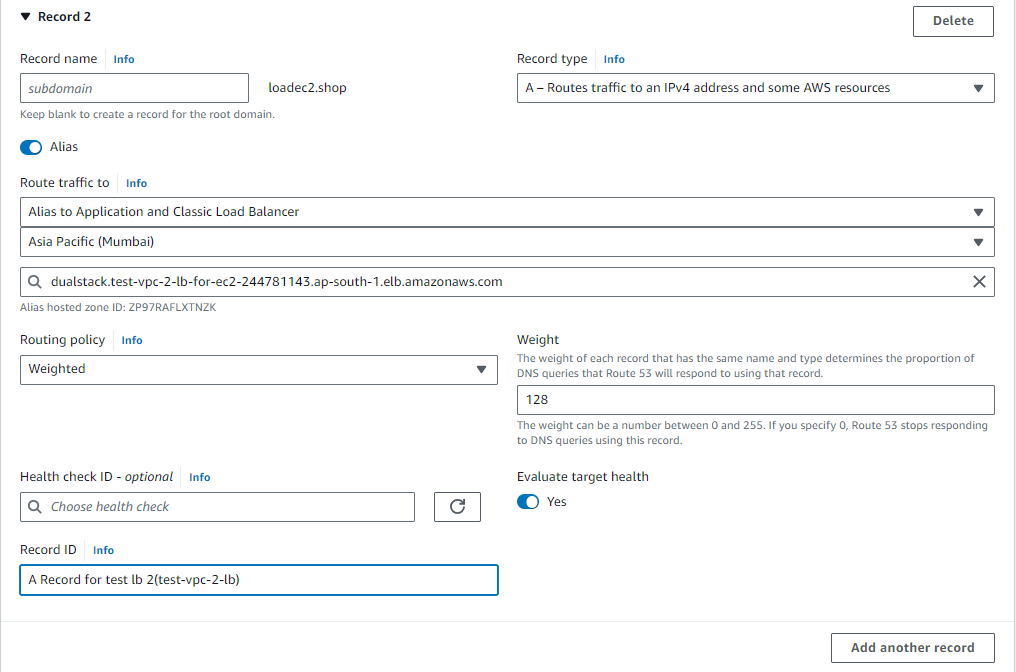
**• Open the browser and past the DNS name it hits to the “welcome to learning-ocen.com” in ec2 instance directly**

****

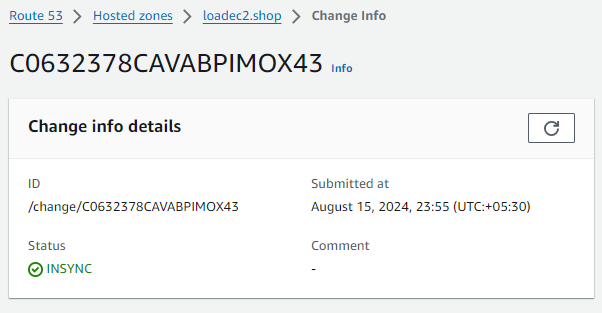
* **Right now our setup is ready loan balancer connection with ec2 instance next we need to create the A-record inside my Route53.**
* **So that we can divide the traffic 50/50 % between this load balancer to tweak that parameter instead of sending 50/50 we will try to change it to make it like a 90/10 or 80/20.**
* **Route53**
* **Now delete record because it’s a simple routing we don’t need this**
* **We are going to create a weighted routing**
* **Enable Alias-Application and classic load balancer-Region-load balancer**
* **Select weighted in the Routing policy**
* **Record 1**
* **Weight-128 (because it half in the 255) – it range should be between 0 – 255**
* **Record ID – A Record for test-lb**

****

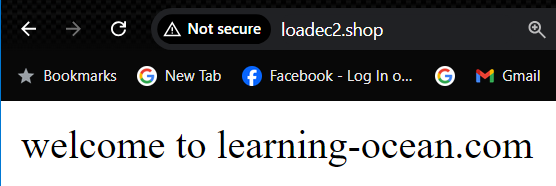
* **Record 2**
* **Weight-128 (because it half in the 255) – it range should be between 0 – 255**
* **Record ID – A Record for test-lb 2(test-vpc-2-lb)**

****

* **Next click on create record and wait until its up and running**

****

* **Now lest go and check the loadec2.shop and see where it goes**

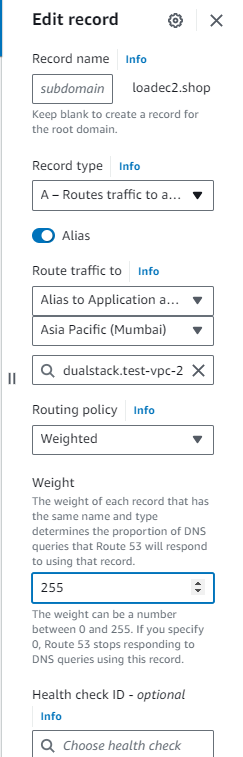
****

* **It go’s to the test-ec2-instance**
* **Let’s go and change the weight by editing the weight in the record-1 & record-2**

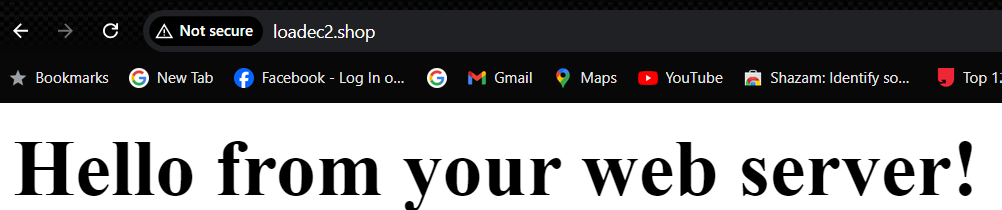
**Record – 1 = 0**

**Record – 2 =255**

* **Put the full weight for the record - 2**

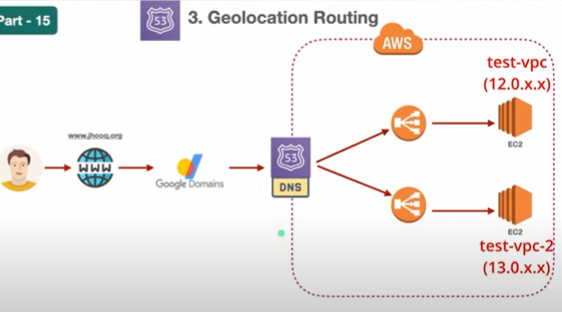
****

* **And click on save**
* **So also why I did 0 to 255 because I’m just running everything into a limited environment.**
* **I don’t have a multiple number of requests, thousands of request coming to ec2 instance so through those split cannot be visible significant in if we go with the 50/50 that’s why I have just gone with the zero and 100%.**
* **So that you can see the difference quite immediately, so that’s the reason behind splitting the traffic instead of 50/50.**
* **I’m just going with a zero and 100%**
* **Next let’s go and check the loadec2.shop it should point the request to the test-vpc-2-ec2-instance**

****

* **This how we are going to using the load balancer you just need to define the weighted in terms of 0 to 256 just chose the suitable number and divide the traffic between those 2-load balancer**

**Geolocation based – Routing**



* **Here all the setup is still the same we are still using the same load balancer or same ec2 instance but we just going to modify our A-Records instead of weighted A-Record now we be creating a geolocation based a record.**