Securing Custom Domains with HTTPS on AWS

In this turorial, we are going to study Domain Management with Route 53 and SSL/TLS Certificate Handling with AWS Certificate Manager.

DNS, or the Domain Name System, translates human readable domain names (for example, www.amazon.com) to machine readable IP addresses (for example, 192.0.2.44).

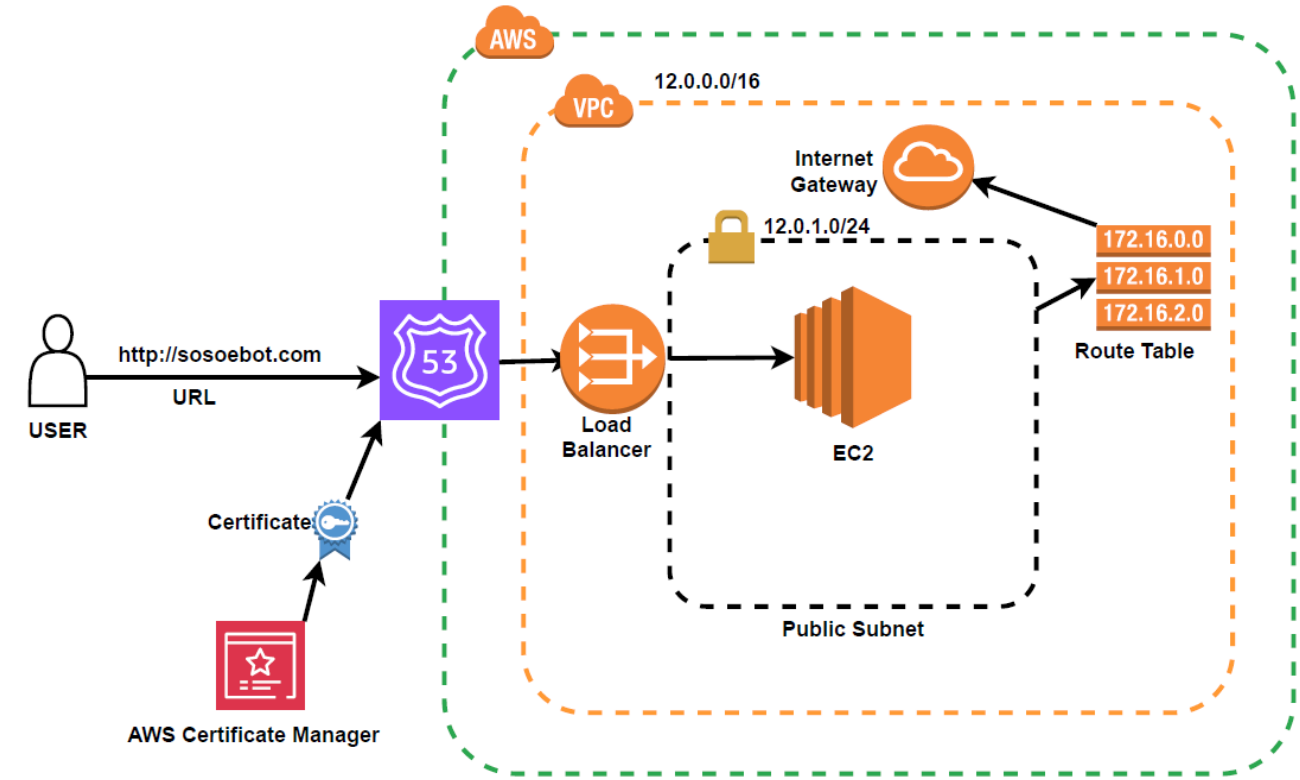
All computers on the Internet, from your smart phone or laptop to the servers that serve content for massive retail websites, find and communicate with one another by using numbers. These numbers are known as IP addresses. When you open a web browser and go to a website, you don't have to remember and enter a long number. Instead, you can enter a domain name like example.com and still end up in the right place.

A DNS service such as **Amazon Route 53** is a globally distributed service that translates human readable names like www.example.com into the numeric IP addresses like 192.0.2.1 that computers use to connect to each other. The Internet's DNS system works much like a phone book by managing the mapping between names and numbers. DNS servers translate requests for names into IP addresses, controlling which server an end user will reach when they type a domain name into their web browser. These requests are called queries.

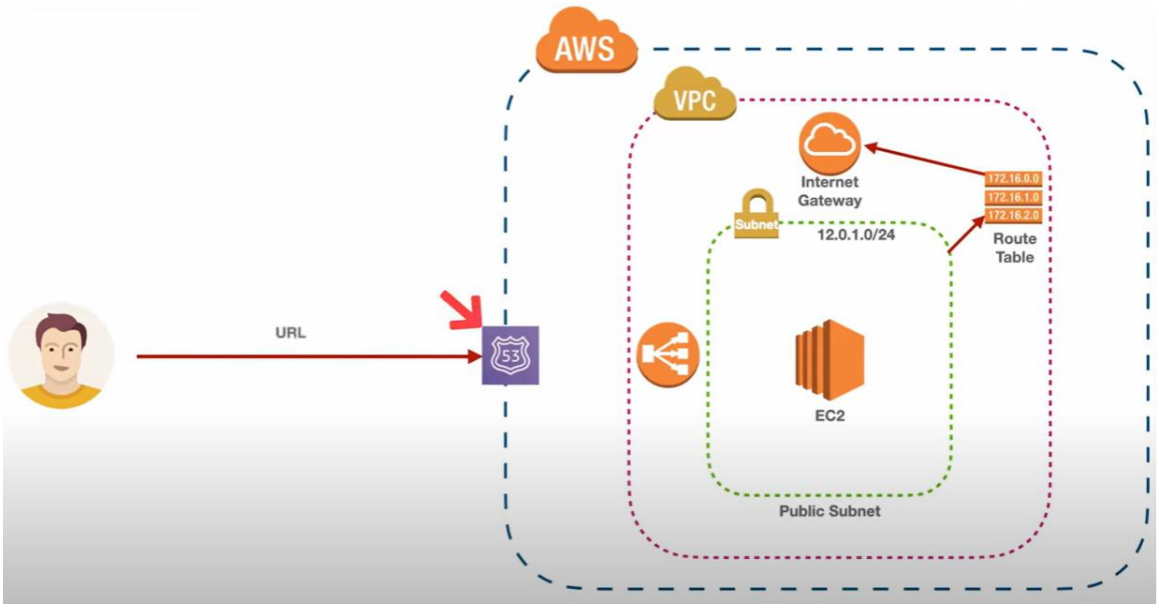
DEMO PREVIEW

In this tutorial, we are going to talk about AWS Certificate Manager. Here is what we are going to implement in our AWS Certificate Manager demo.

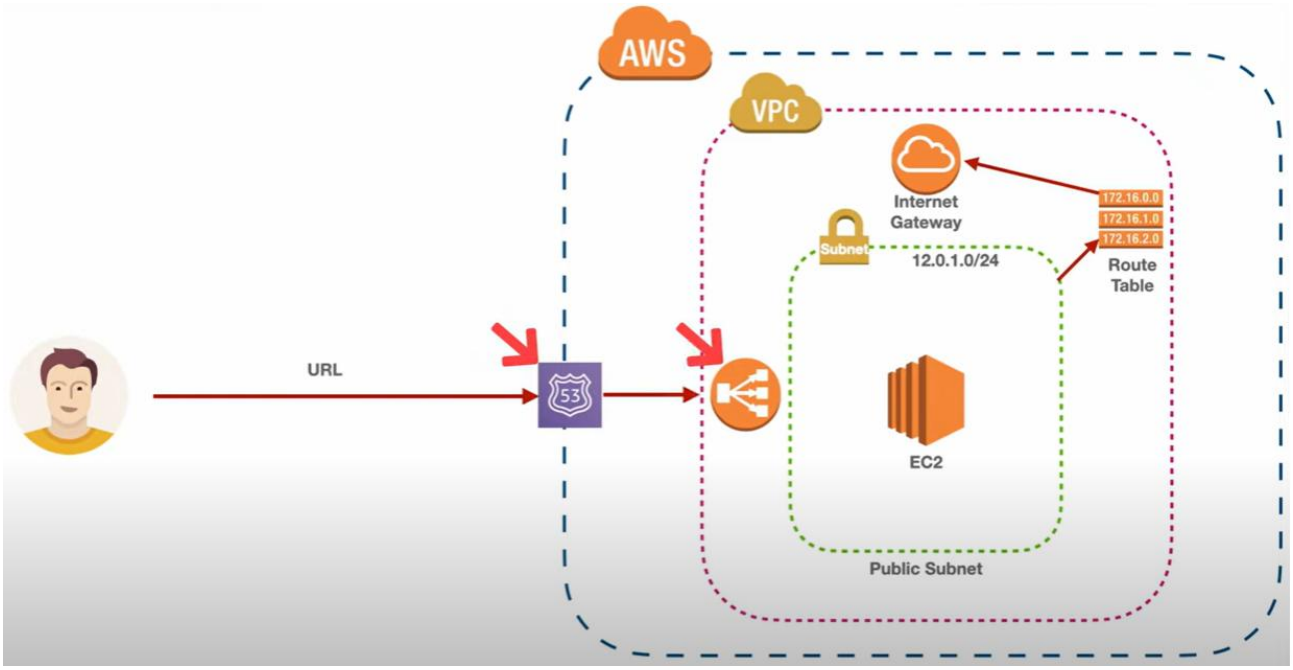
ARCHITECTURE



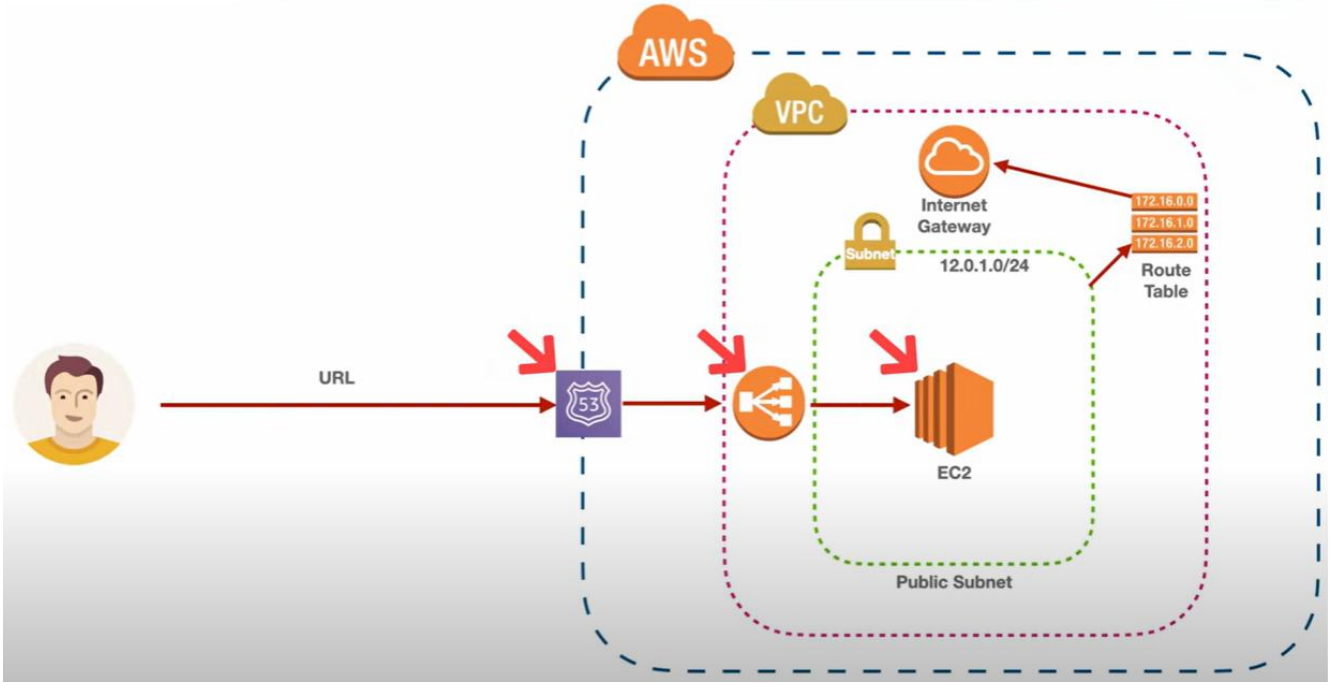
First, the user is going to make a request using the URL that request is going to land on our **AWS Route 53**.



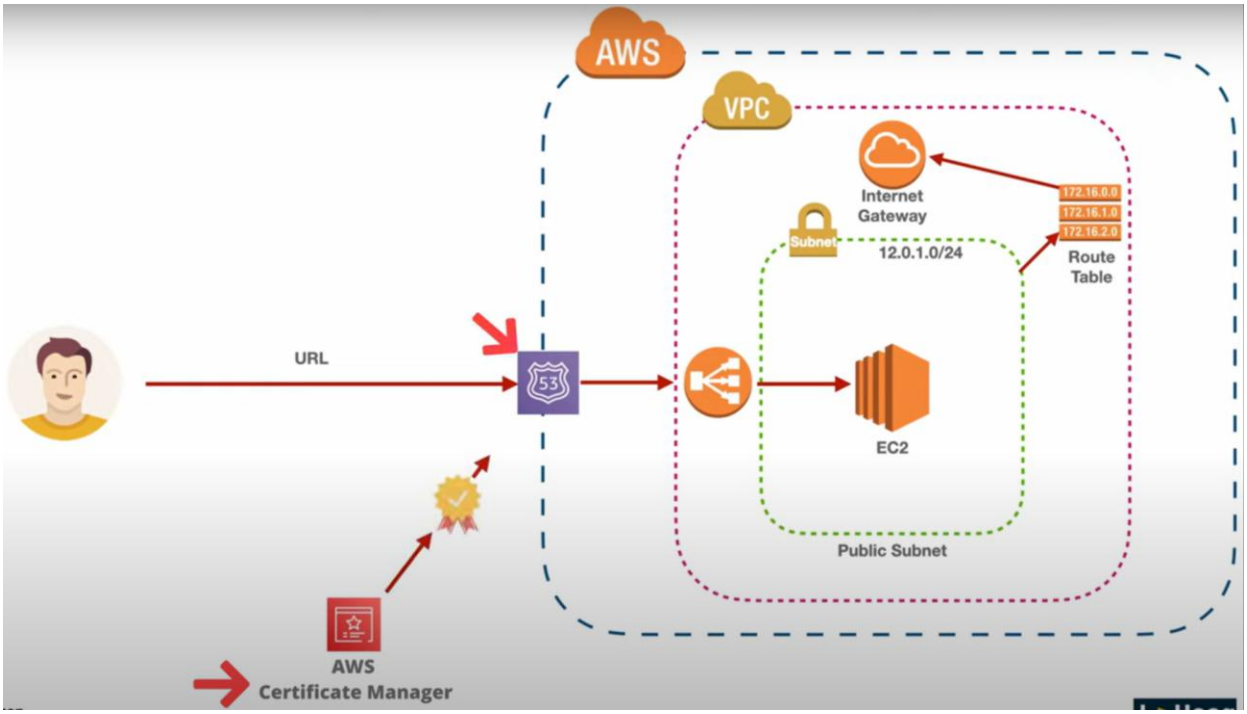
Route 53 is going to forward that request to our **application load balancer**.



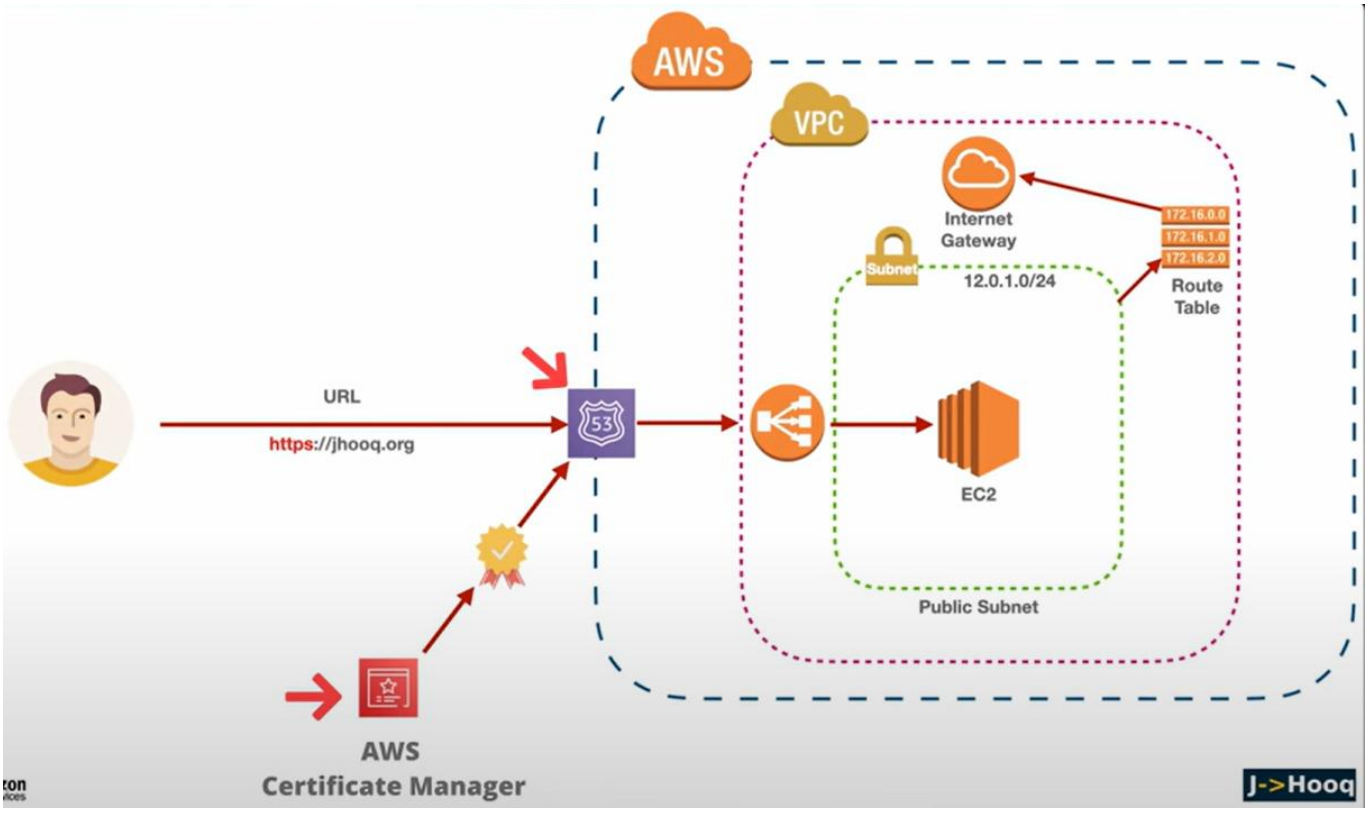
That Load Balancer is going to point to our **EC2 instance** where our application is running.



To secure that URL, we are going to use **AWS Certificate Manager**, which is going to issue the certificate to us and we are going to use those certificate records inside our AWS Route 53.

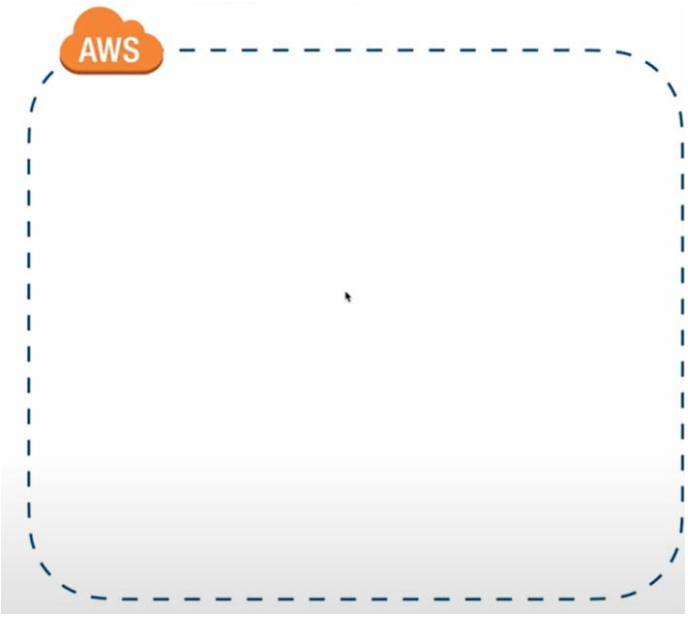


Once we have imported those records inside our Route 53, then we are going to use the same URL, but this time we are going to use the **HTTPS**.

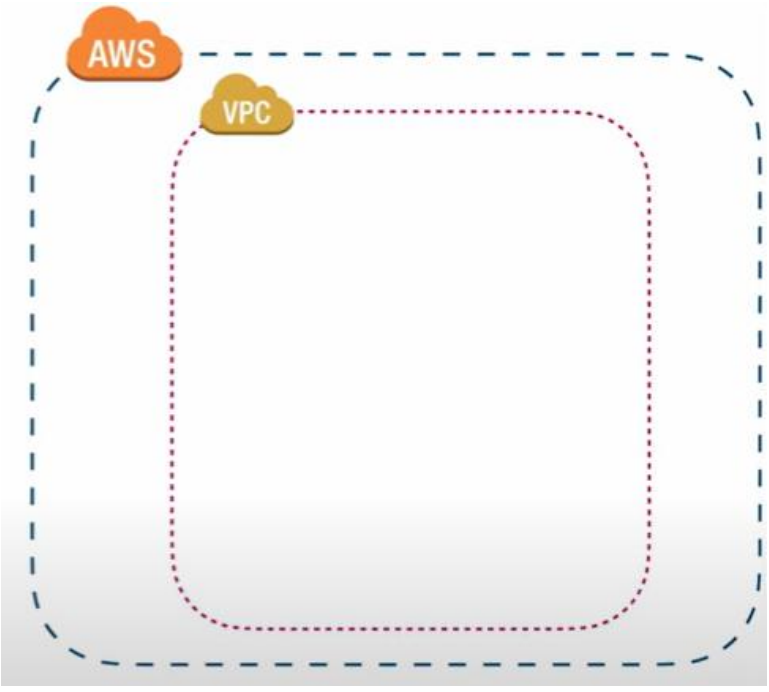


SETTING UP ARCHITECTURE COMPONENTS

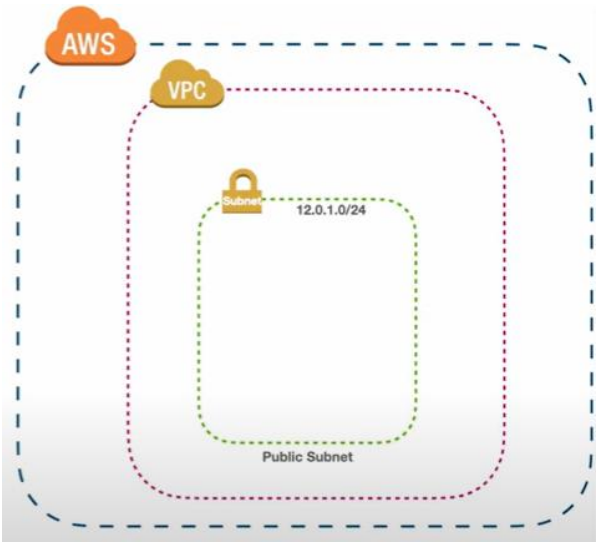
Let us break down this whole implementation into multiple steps. The first thing we will start with is our **AWS Account**.



Secondly, we are going to set up our **VPC** within the AWS Account.

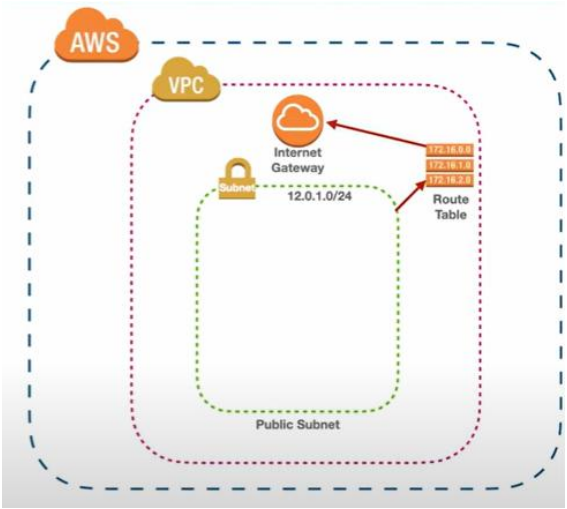


Thirdly, we are going to create a **subnet** inside that particular VPC.



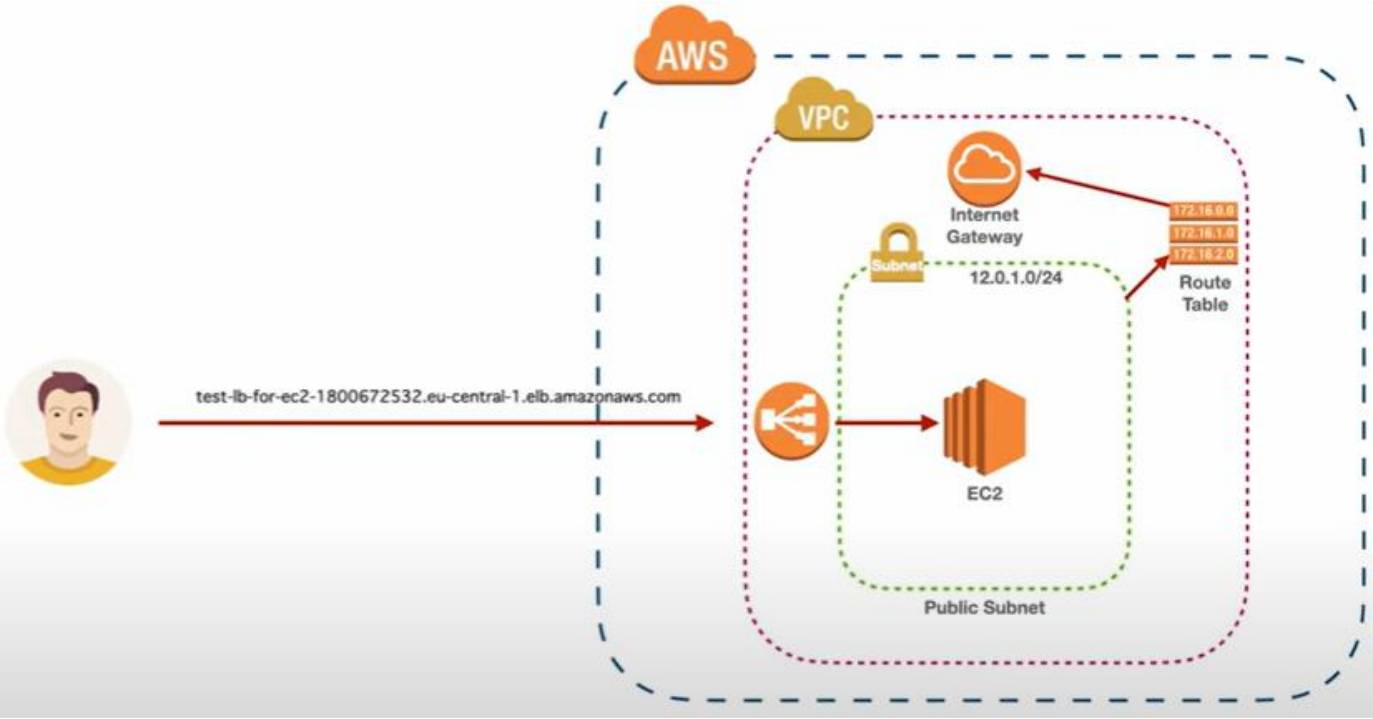
Fourthly, we are going to we are going to set up our **Internet Gateway**.

Fifth, we are going to set up our **Route Table**.

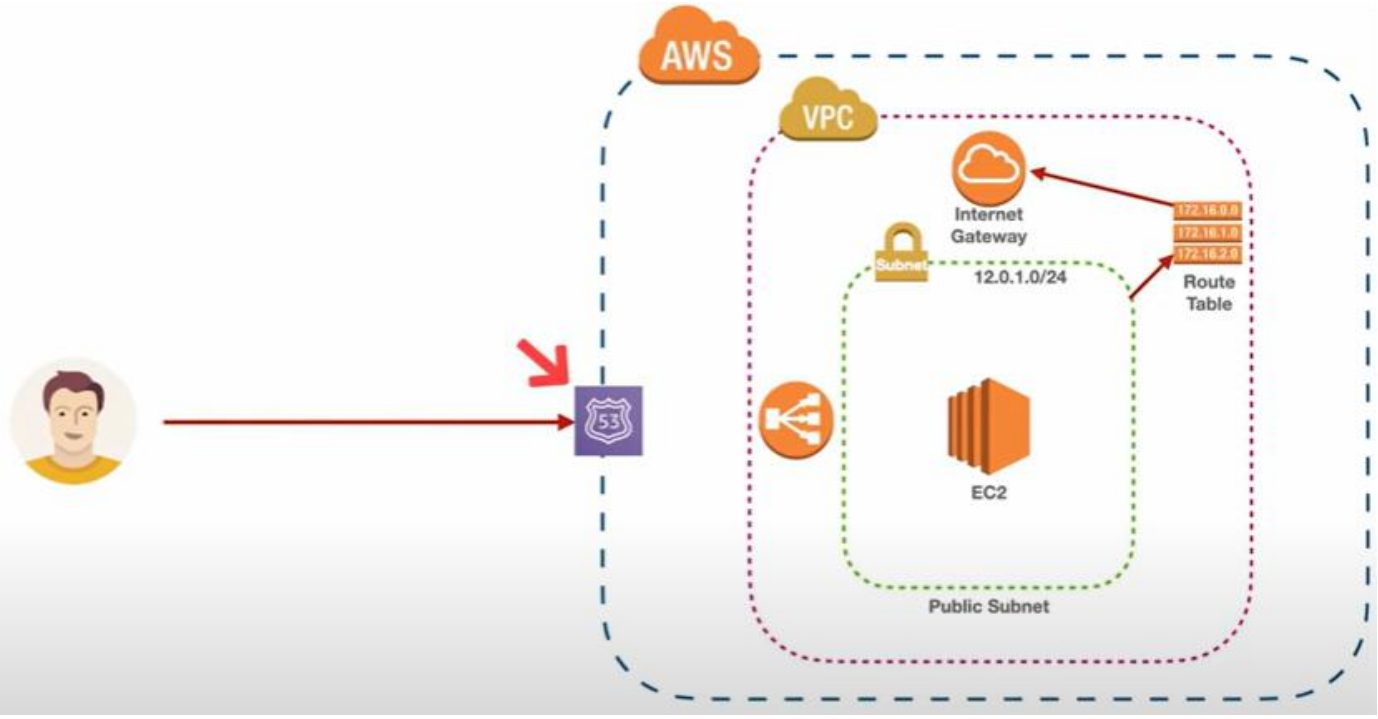


Sixth, we are going to set up our **EC2 instance**.

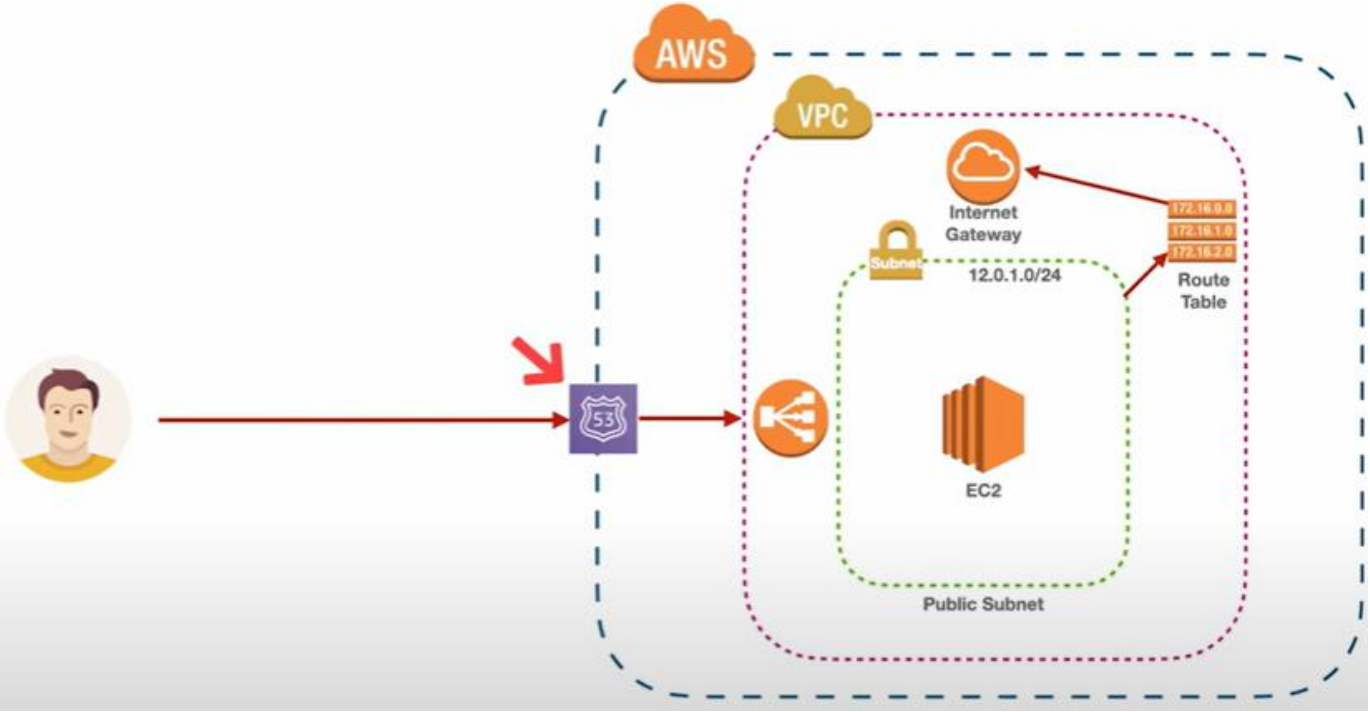
After setting up our EC2 instance, then we are going to create an **Application Load Balancer** so that a user can access our application running in the EC2 instance via the load balancer. For that we are going to use a DNS name of our Load Balancer.



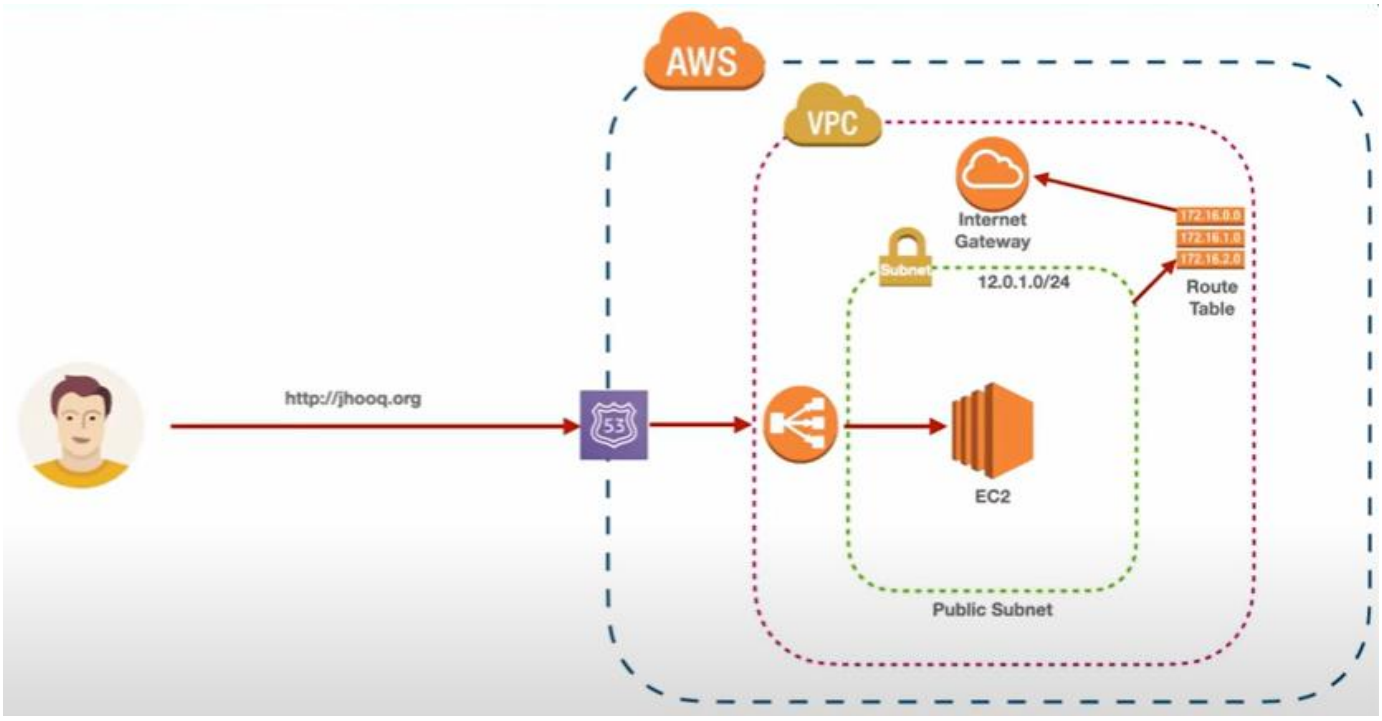
After setting up the load balancer, the next thing we are going to configure is our **AWS Route 53**.



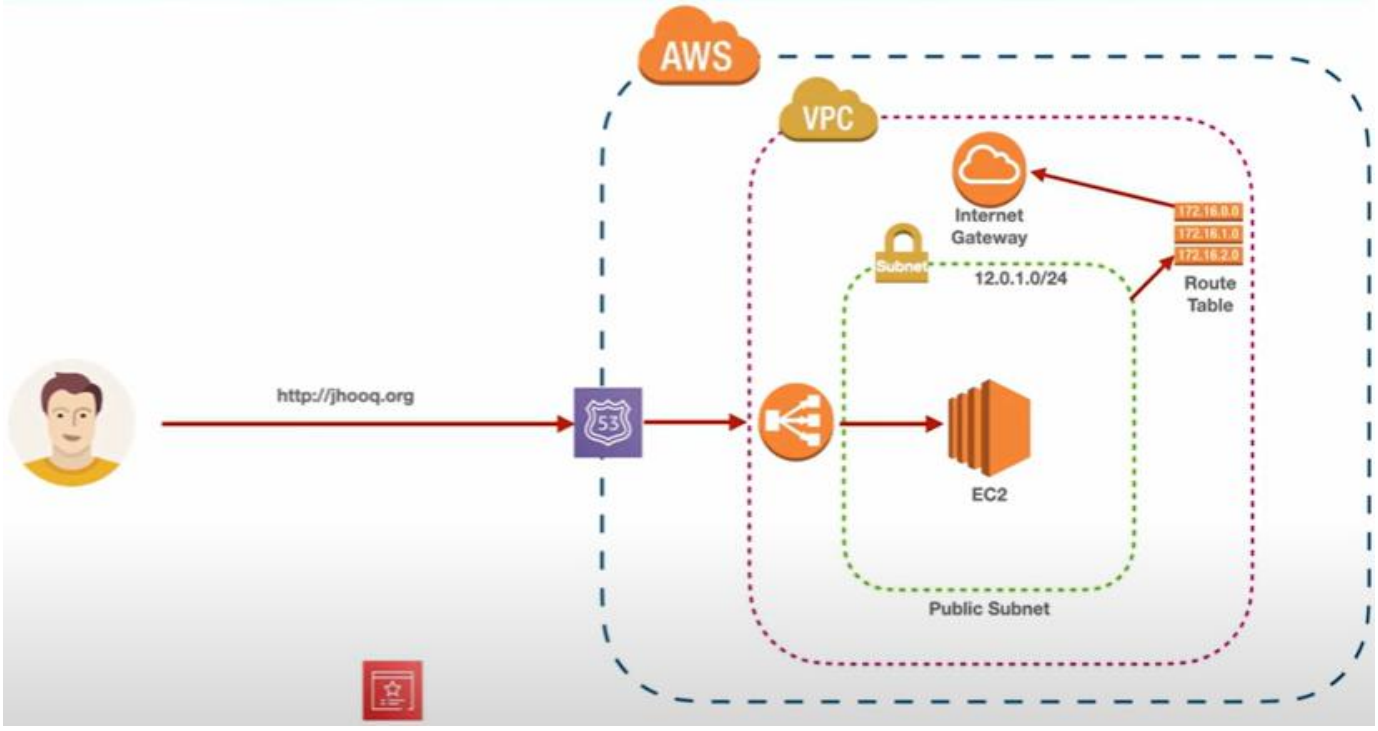
Once we configure our Route 53, then the user can make a request using the URL.



Here you can see in the diagram, we will have a Route 53 on top of our Load Balancer and that user can use the URL like **http://sosoebot.org** which is my own domain to access the particular service that is running on to my EC2 instance.



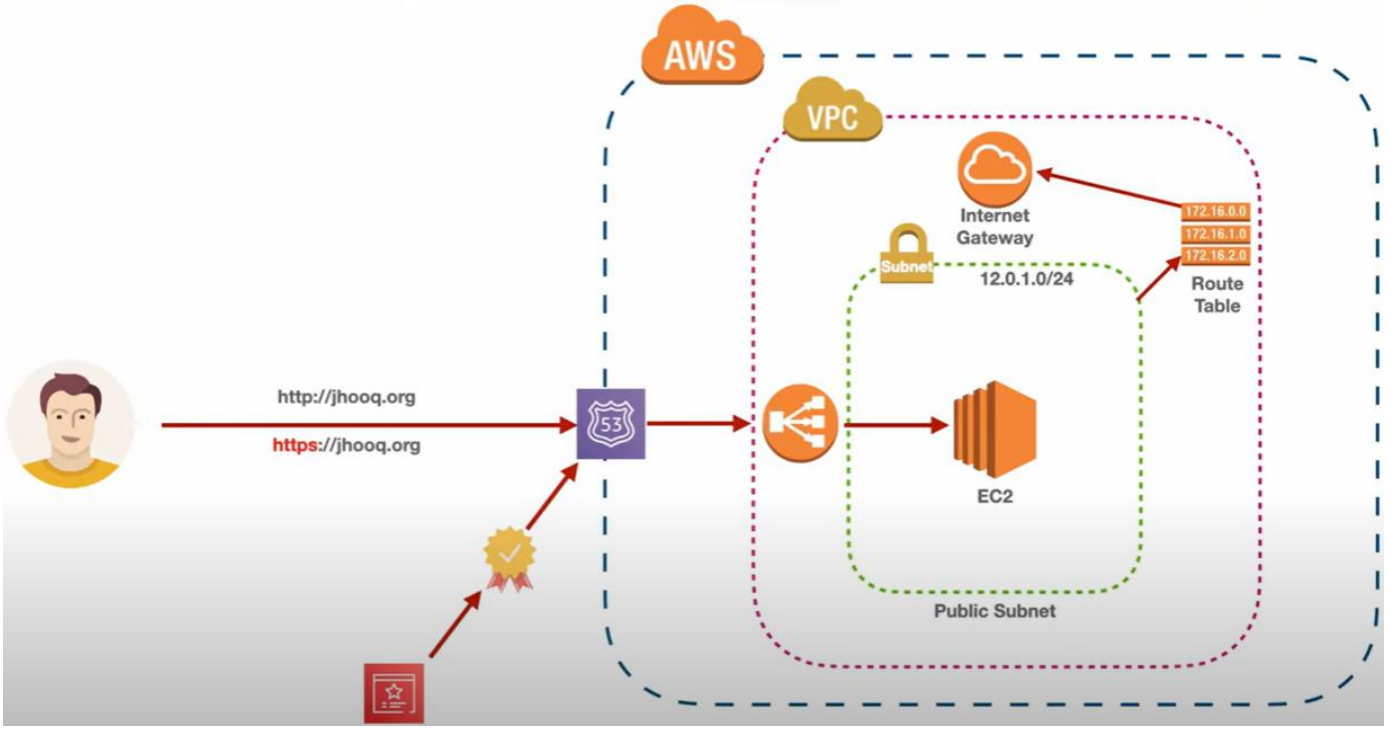
But as you have noticed in the diagram, the URL I am using is http and not https. To introduce HTTPS into the URL, then we need to use the AWS Certificate Manager.



So, we are going to use the AWS Certificate Manager to generate those certificates

And we are going to use those certificate records to insert into our AWS Route 53.

So, once we create those records inside our AWS Route 53 after generating those certificates, then we can use the URL that is HTTPS along with my domain.

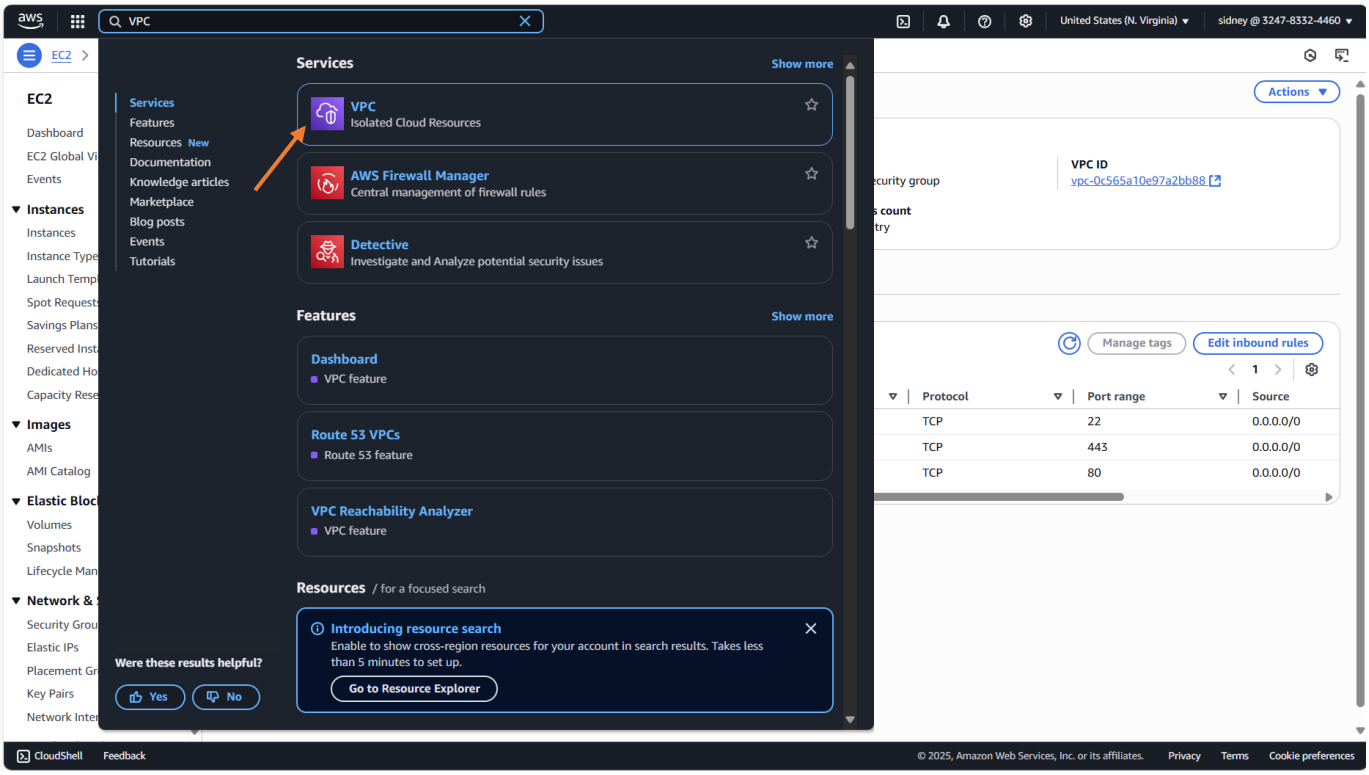


The entire process is broken down into four simple steps for better understanding:

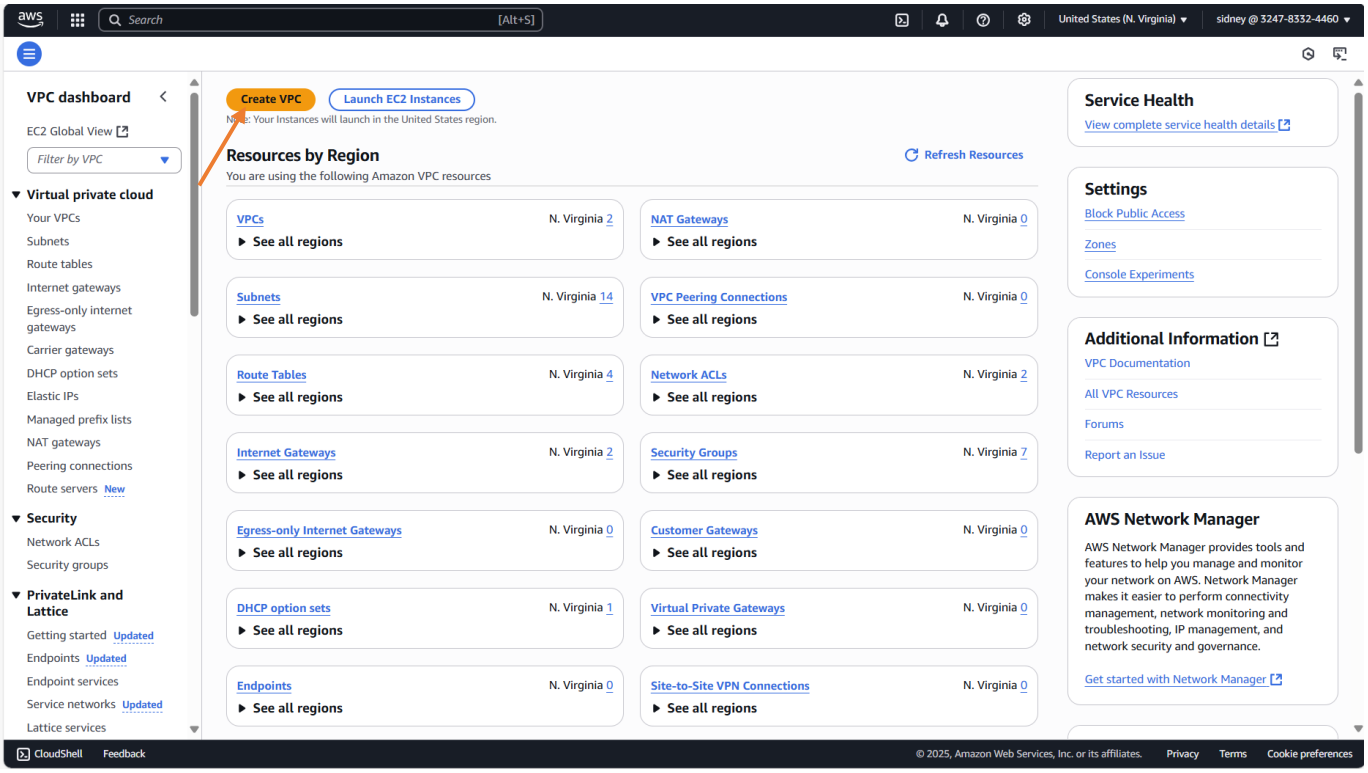
- Set up our VPC
- Create a Public subnet inside the VPC
- Set up our Internet Gateway
- Set up our Route Table
- Set up our EC2 Instance
- Create an Application Load Balancer
- Configure our Route 53
- Set up AWS Certificate Manager

STEP 1: Set up our VPC

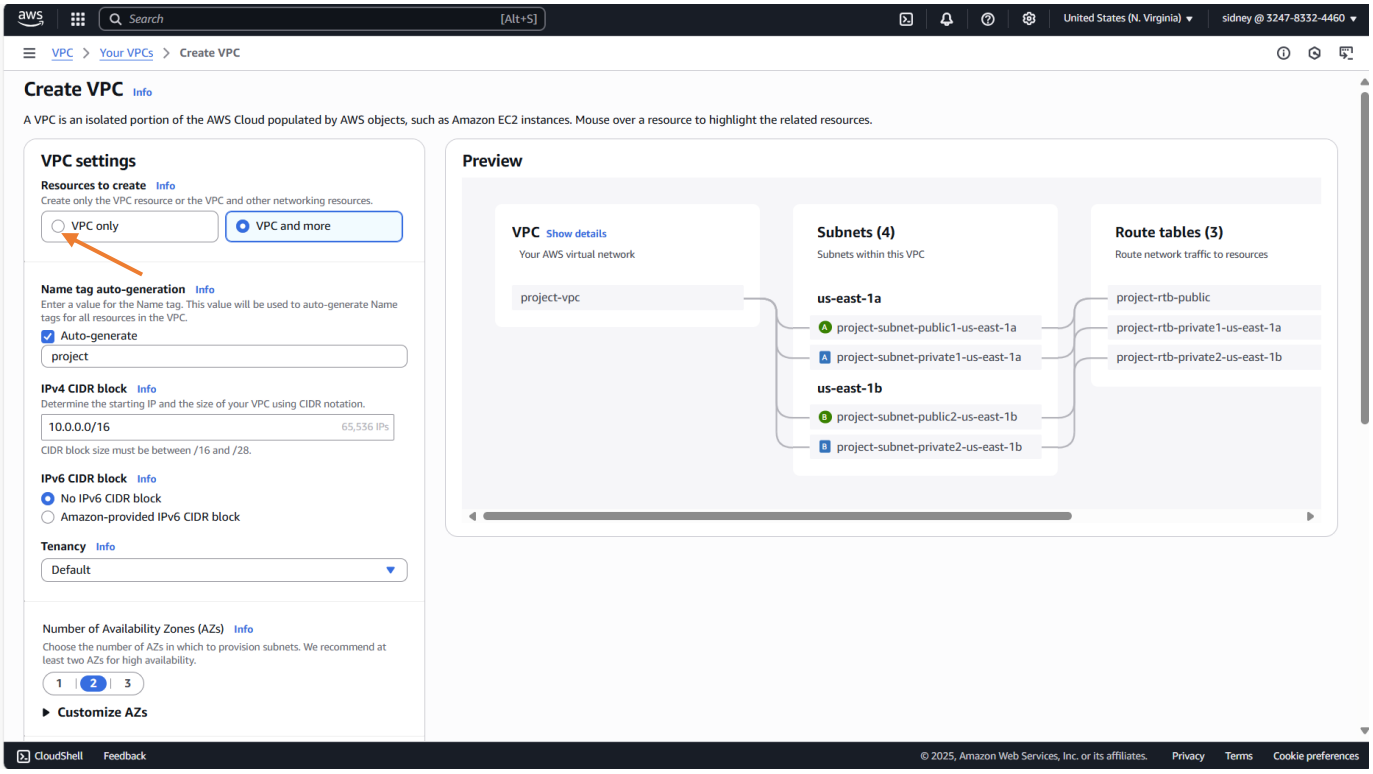
Go to AWS Management console and search for “**VPC**”



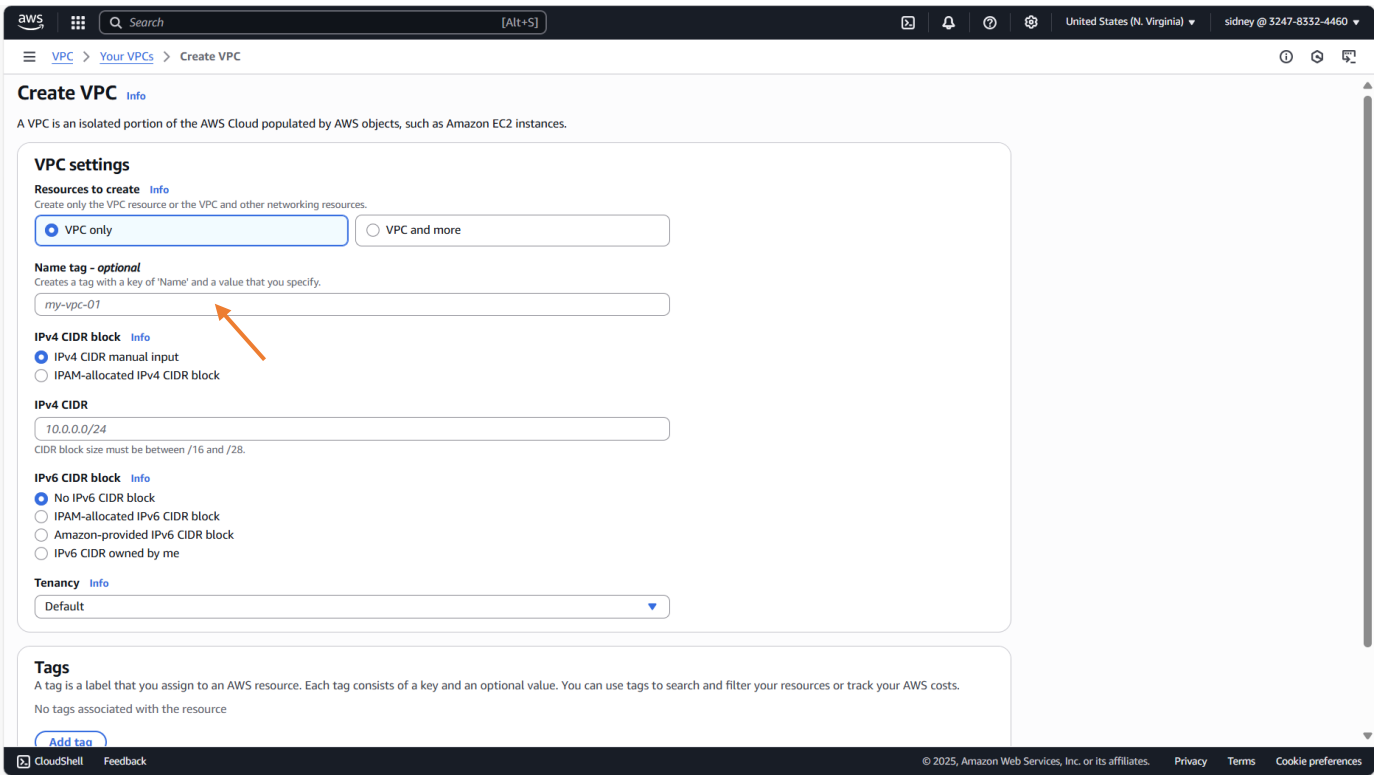
Click on “**VPC**” under services



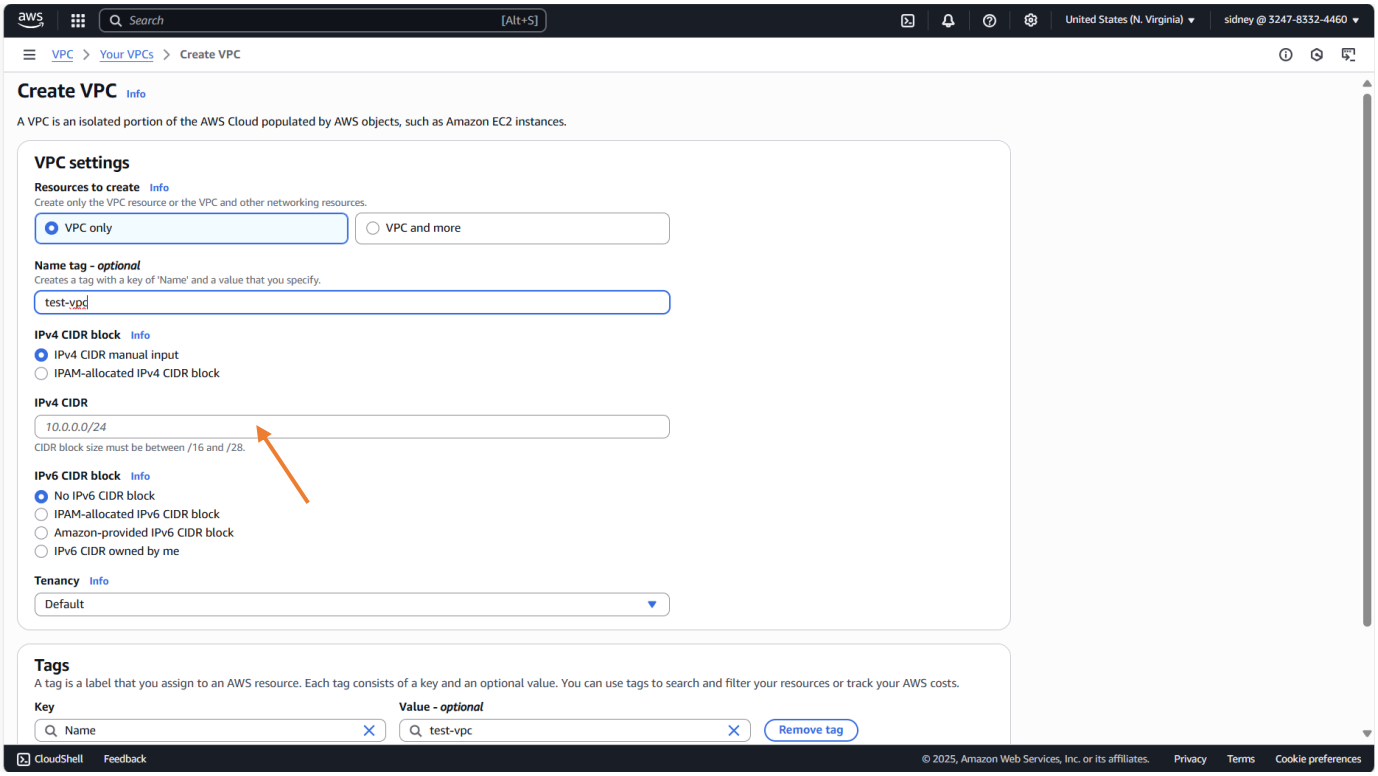
Click on “**Create VPC**”



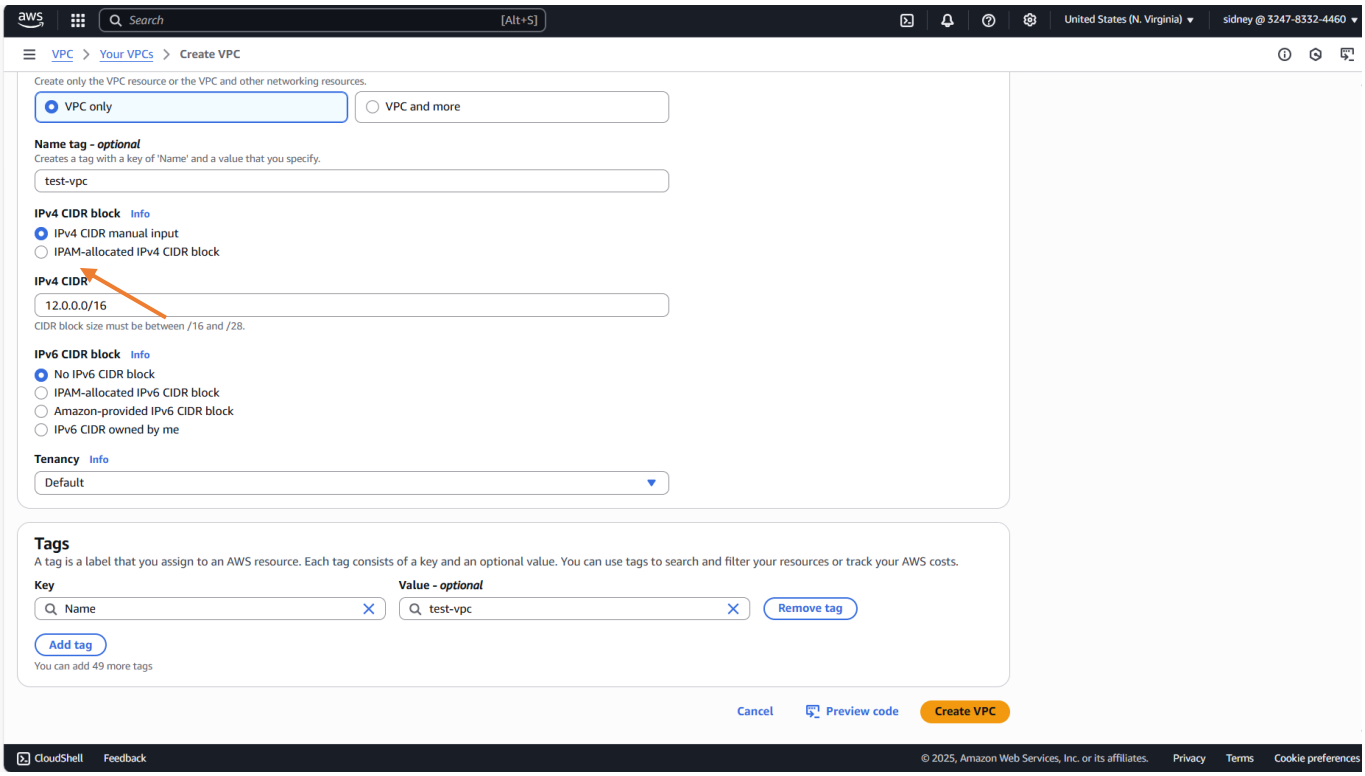
Choose “**VPC Only**”



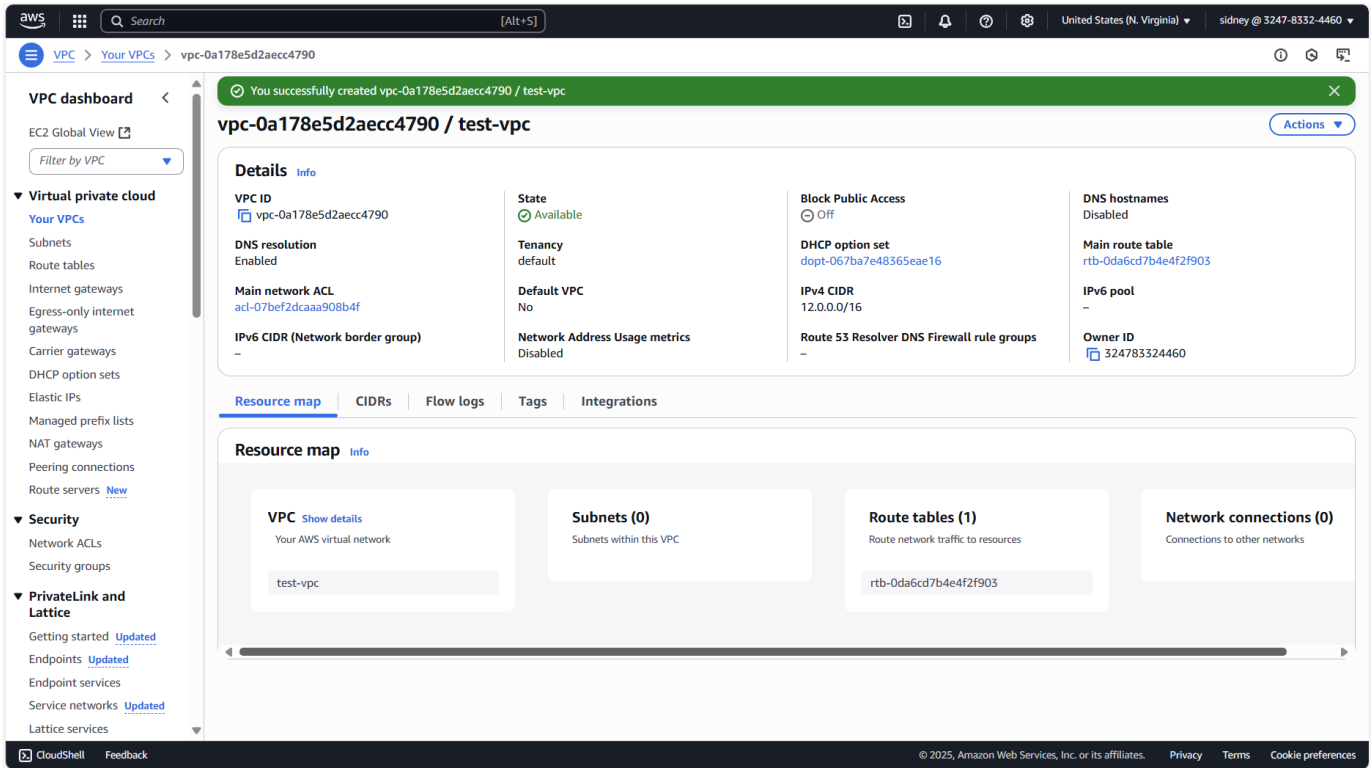
Then we have to give the VPC a name, we will call it “**test-vpc**”



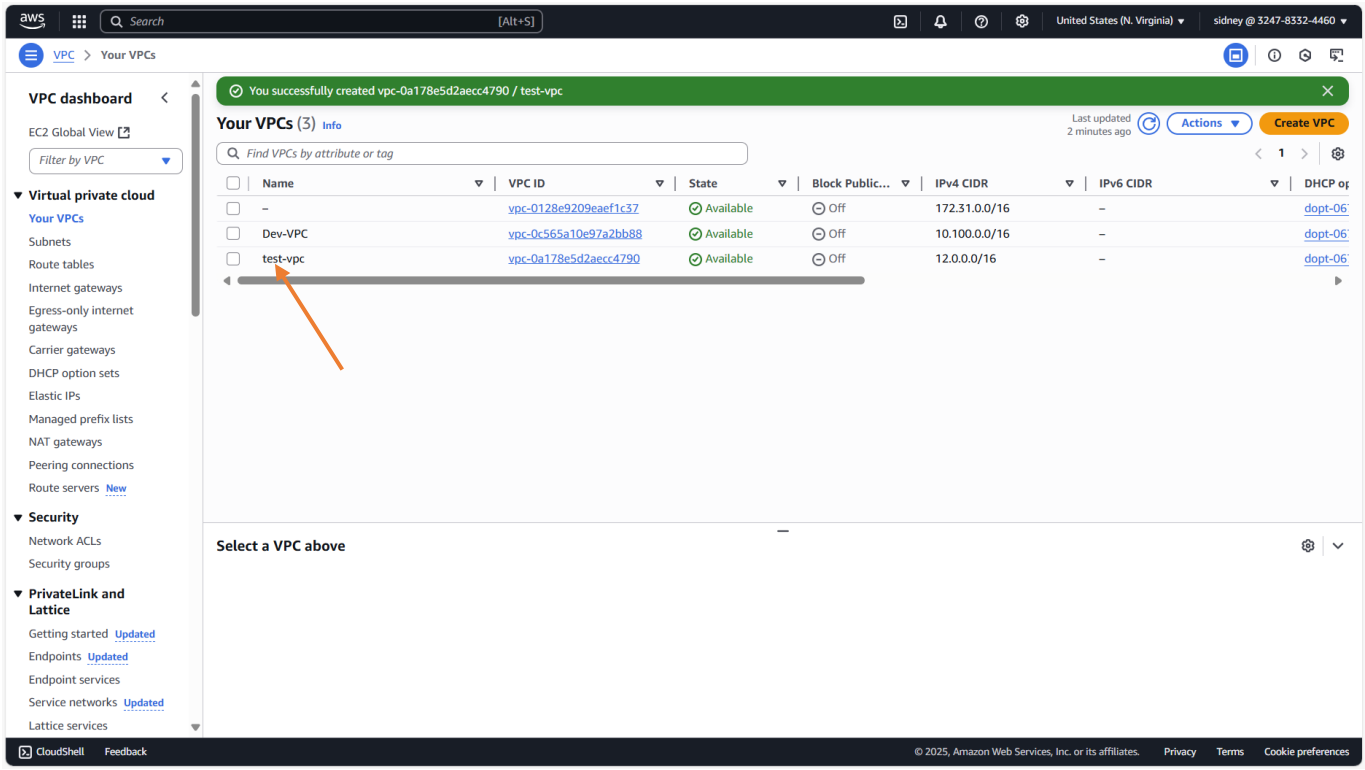
Next, we have to give our IPv4 CIDR range. We will use "**12.0.0.0/16**"



Click on "**Create VPC**"



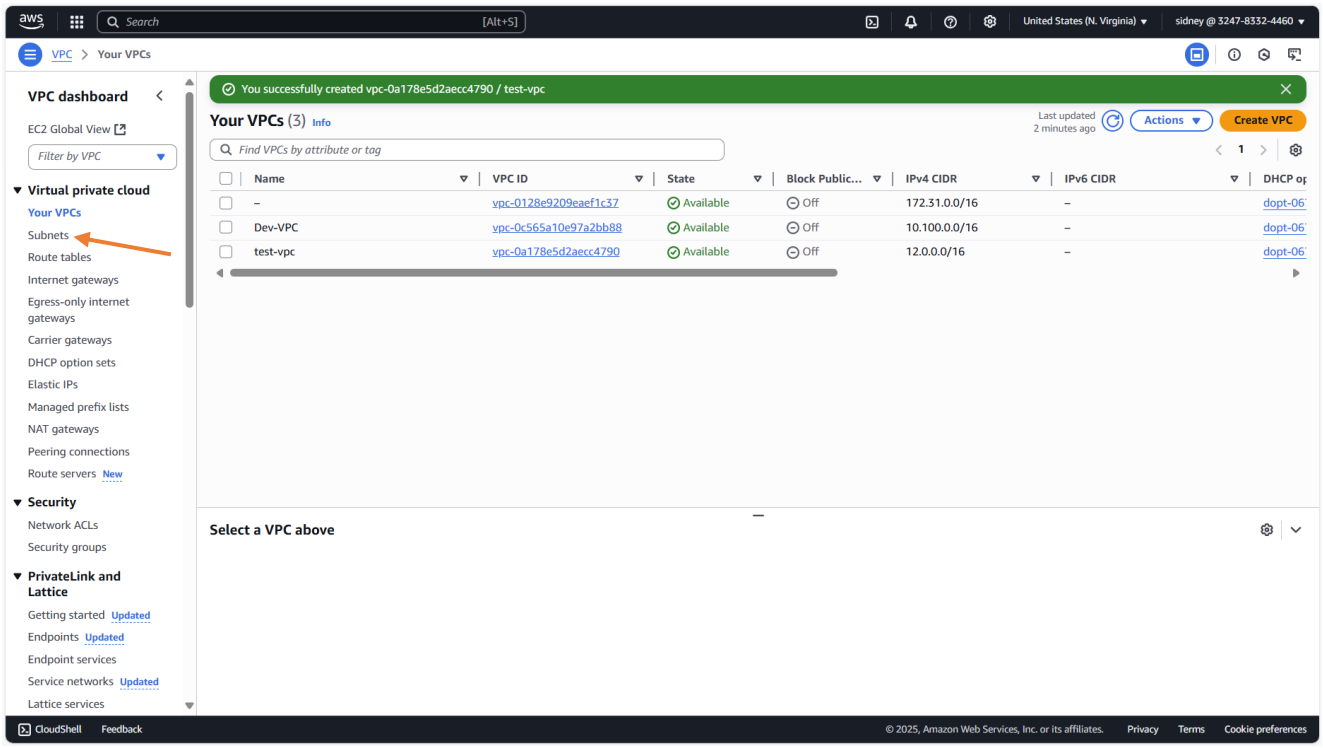
Our “**test-vpc**” has been created. This can be verified by clicking on “**Your VPCs**”



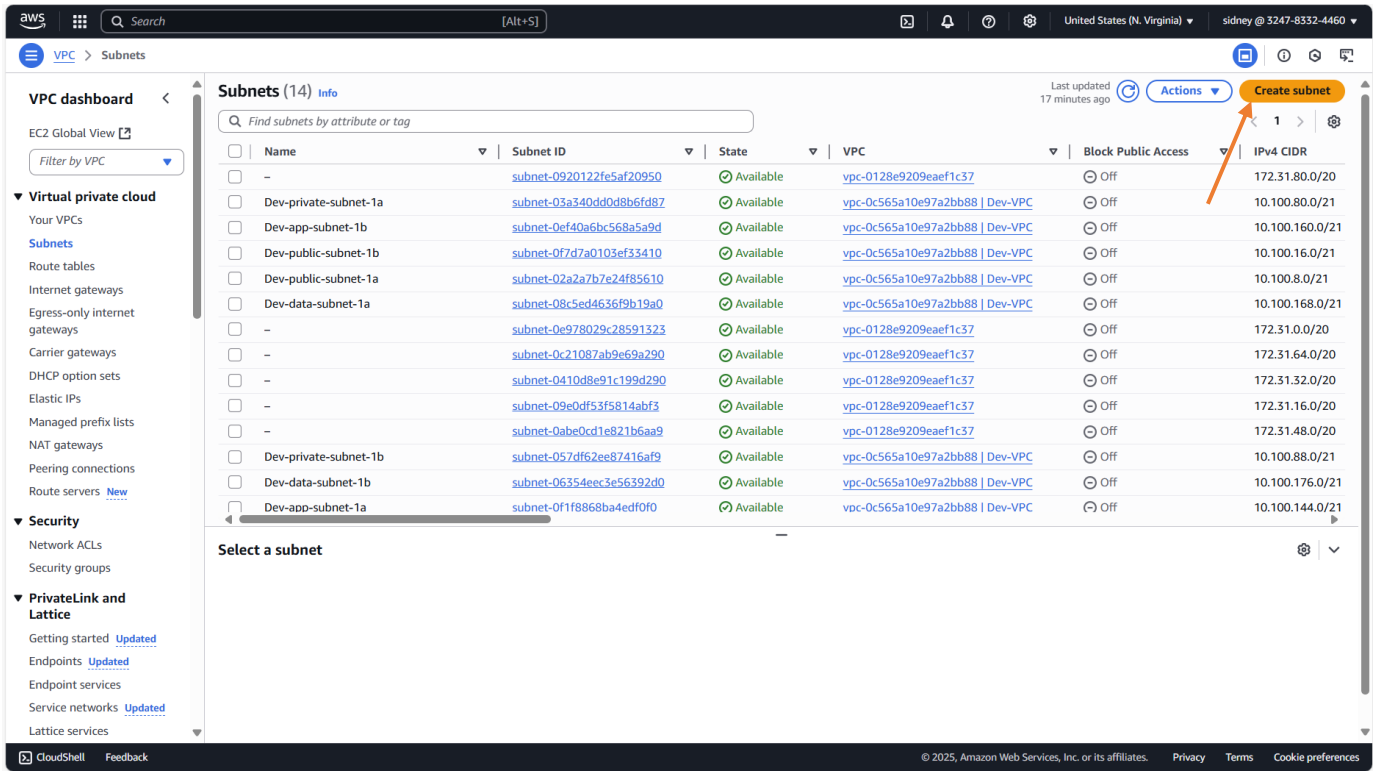
You can see the VPC we just created.

STEP 2: Create a Public subnet inside the VPC

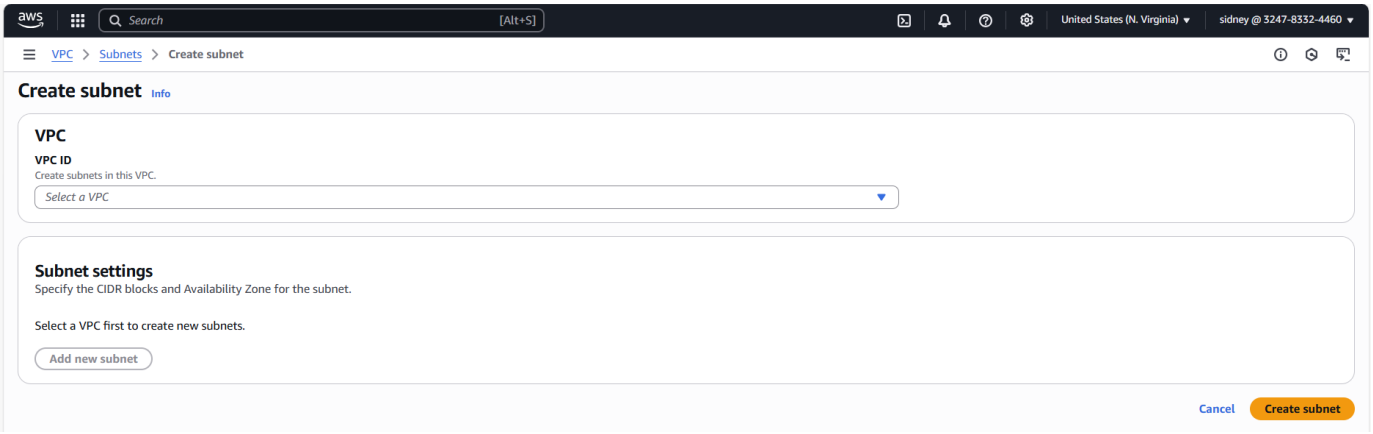
We are going to create a public subnet inside our VPC with IPv4 CIDR range of "**12.0.1.0/24**"



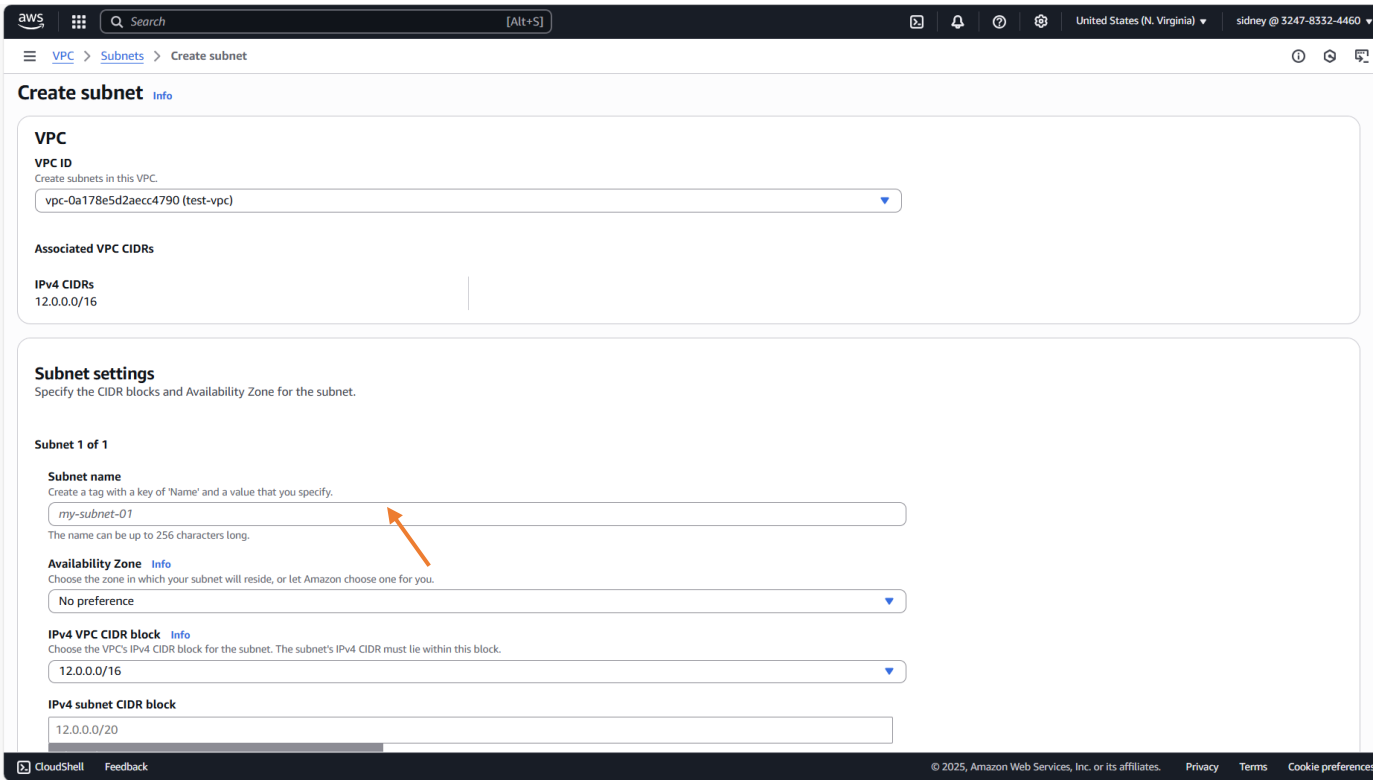
Click on "**Subnets**"



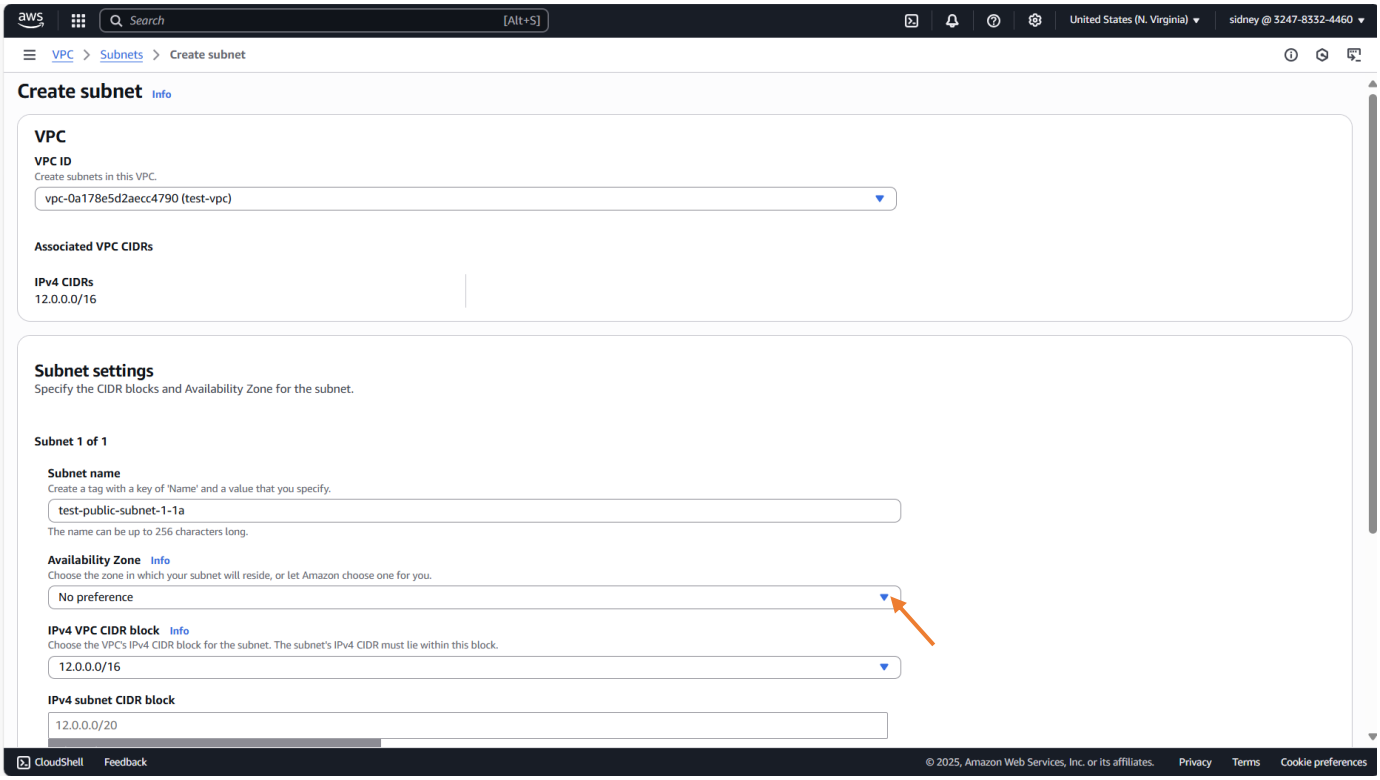
Click on "**create subnet**"



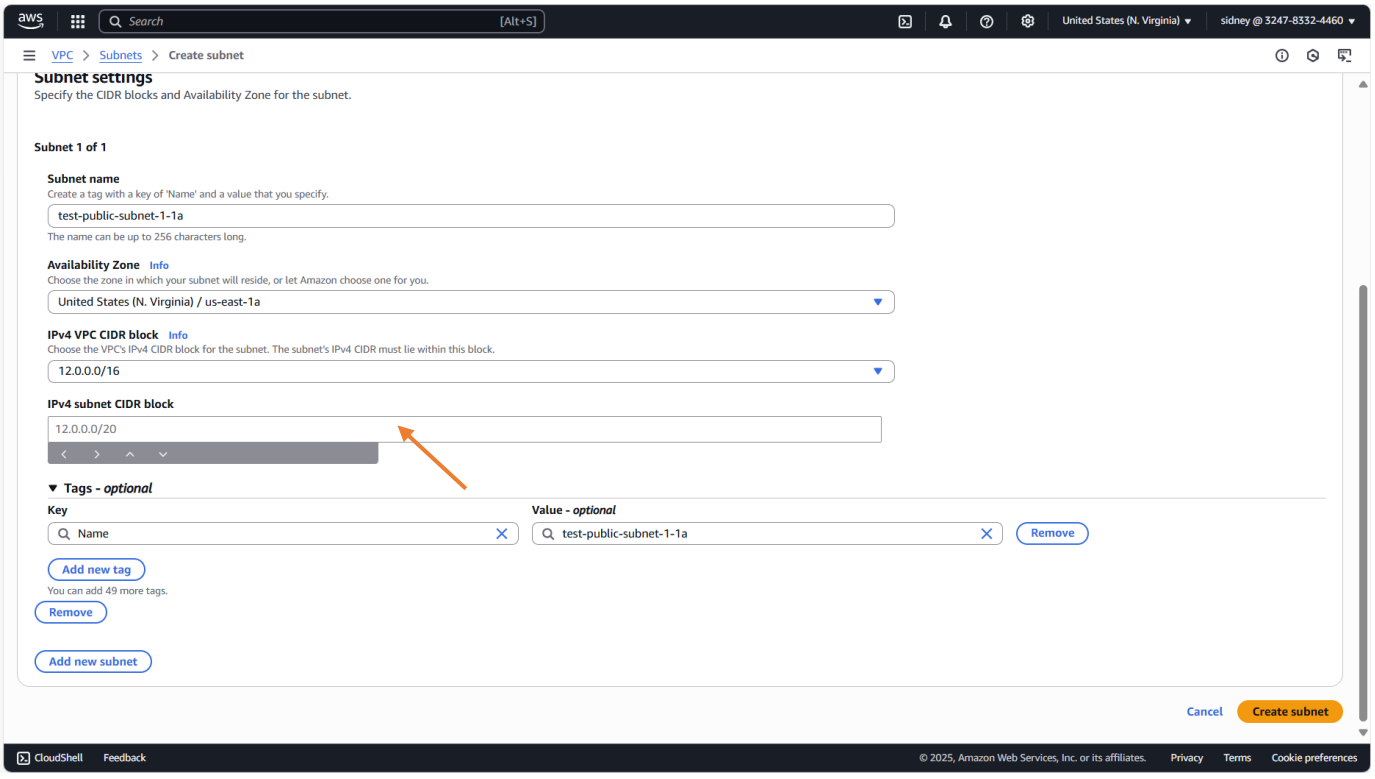
Click on the drop down on “**VPC ID**” and select our VPC



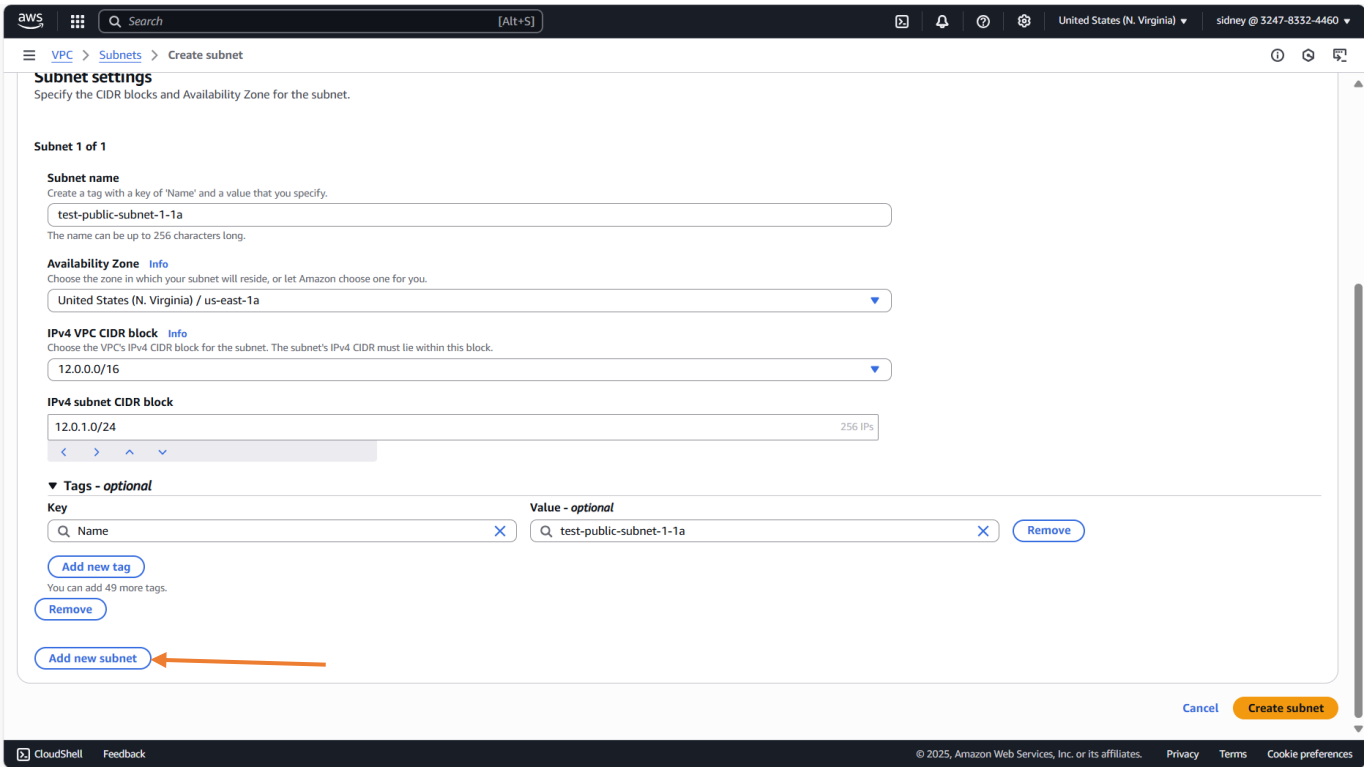
Then we will give the subnet the name “**test-public-subnet-1-1a**”



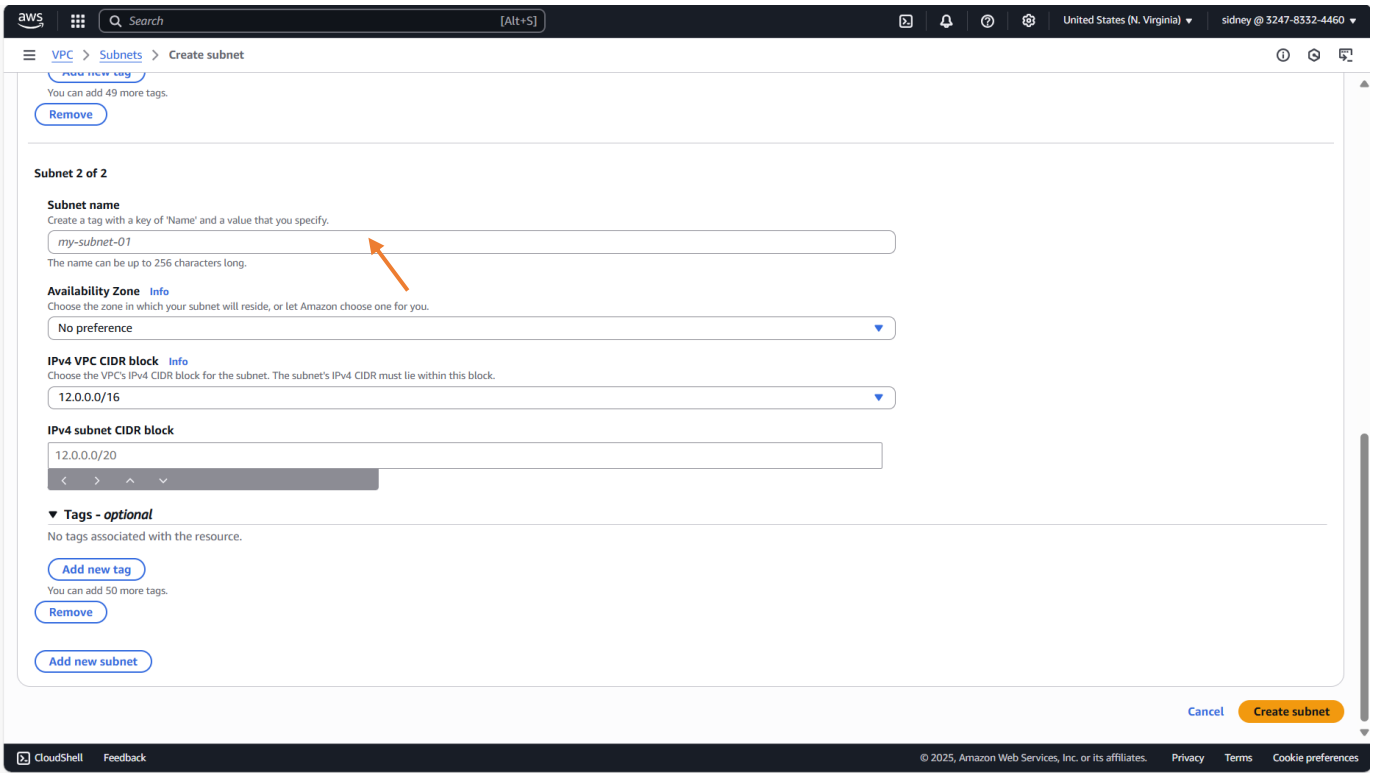
Click on the drop down on "**Availability Zone**" and select "**us-east-1a**"



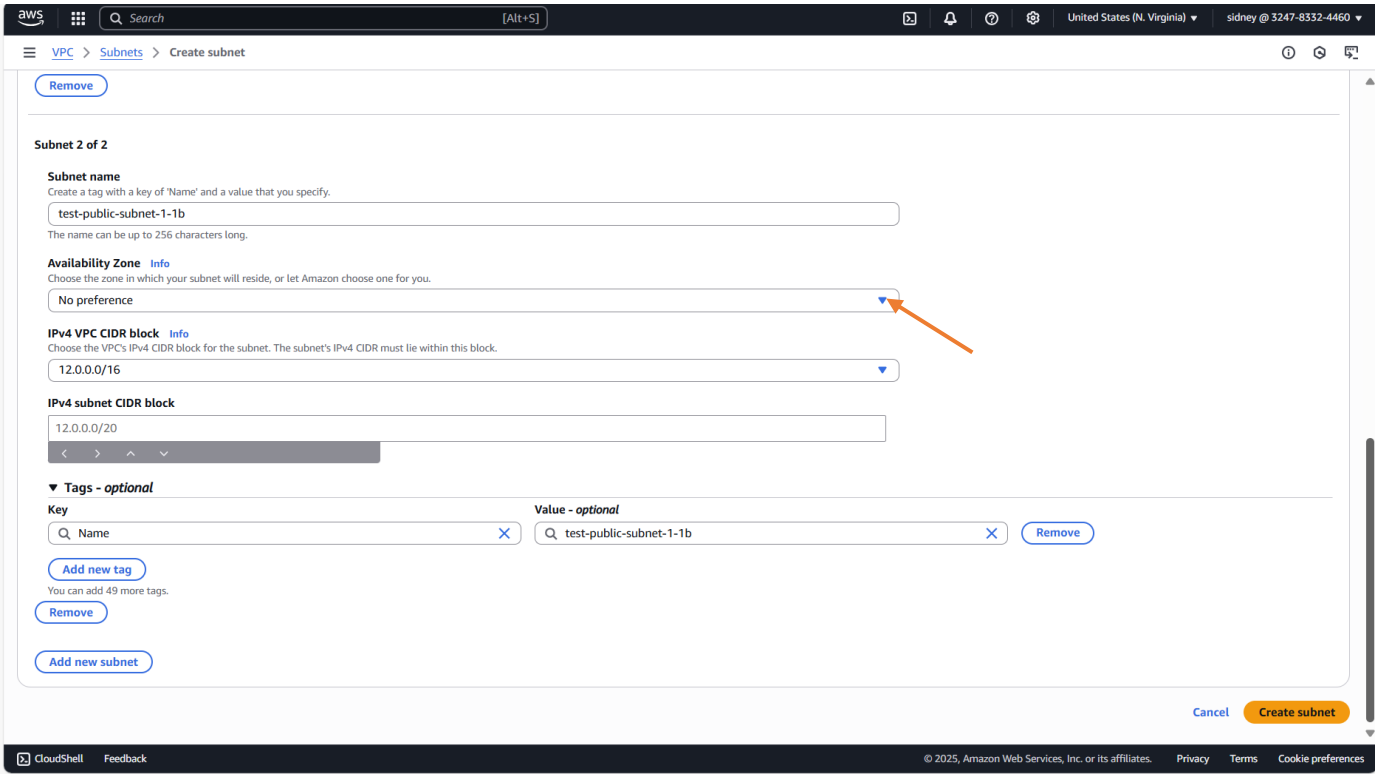
Next is to specify the IPv4 CICR range of the subnet. We will use "**12.0.1.0/24**"



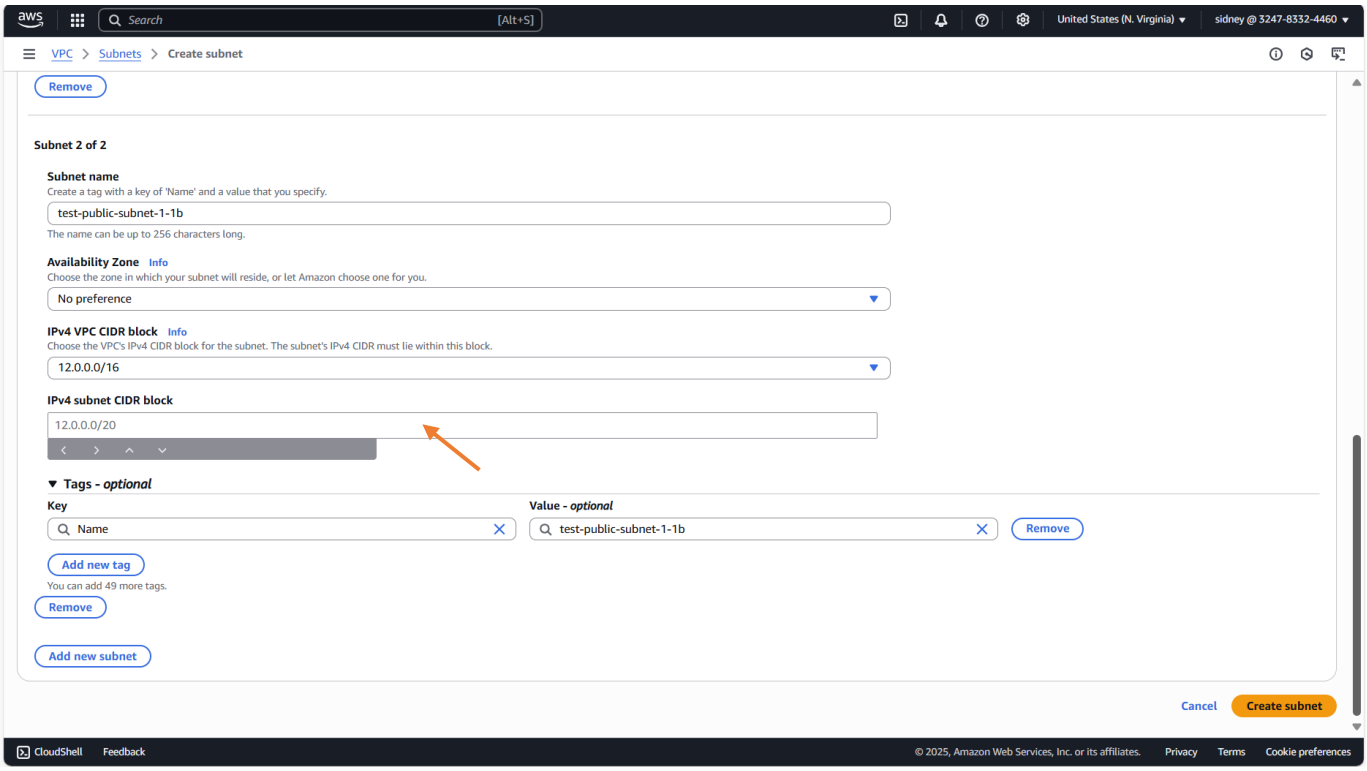
The we will add one more subnet so that we can have two subnets for more availability. Click on "**Add new subnet**"



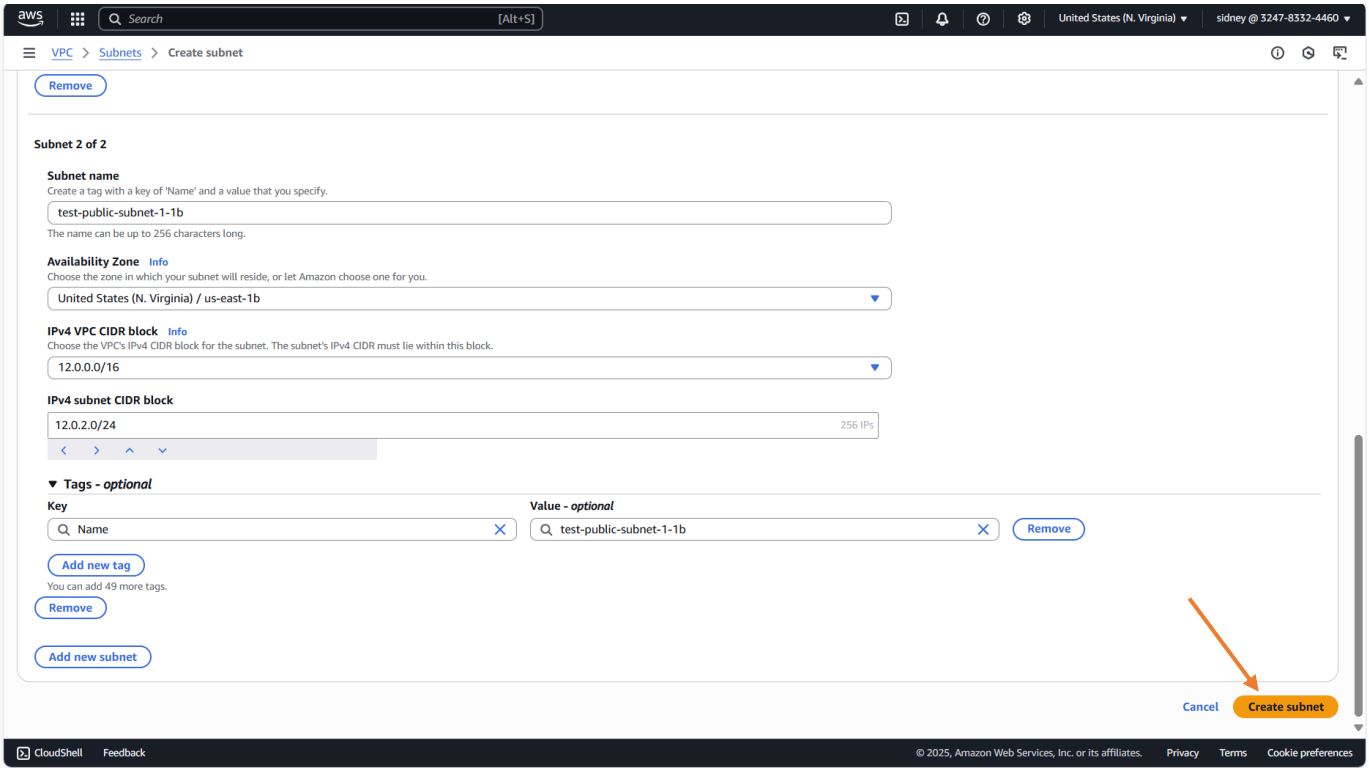
We will name the subnet as "**test-public-subnet-1-1b**"



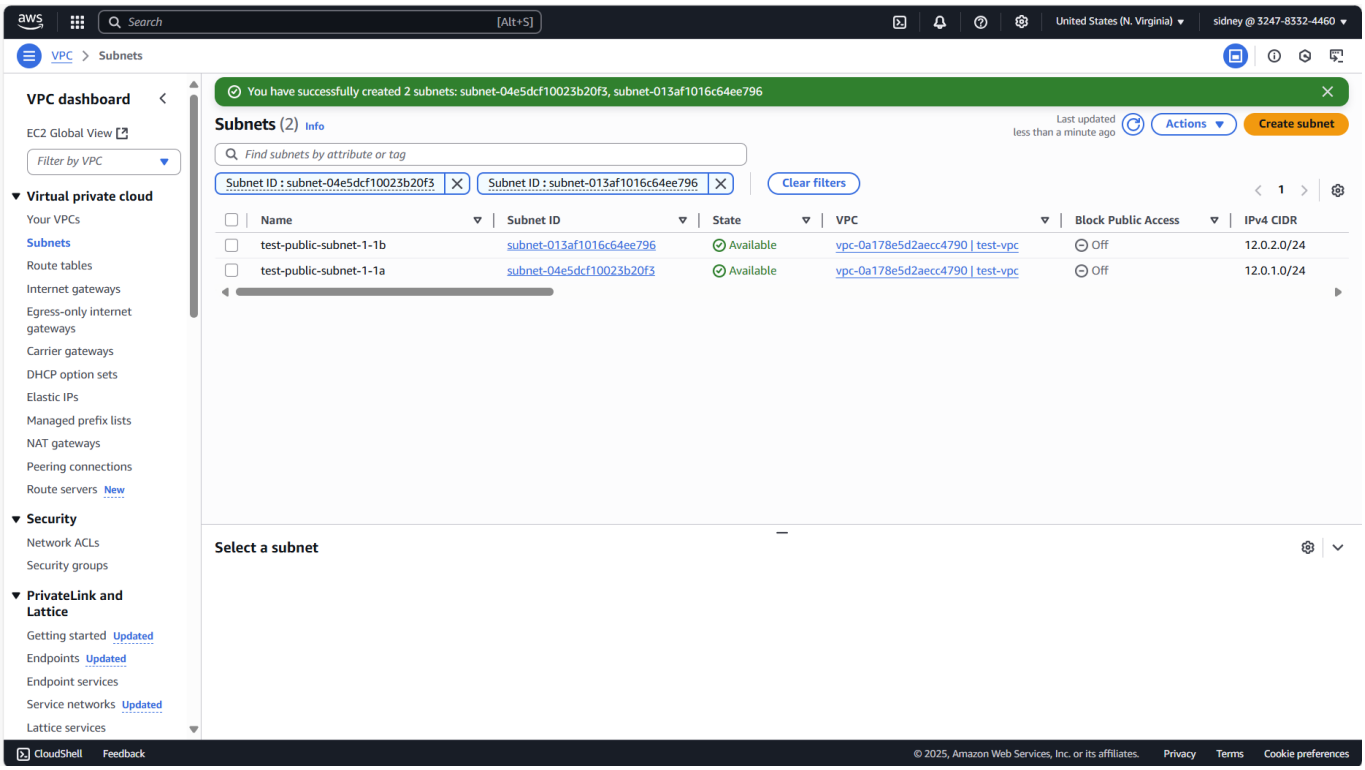
Click on the drop down on “Availability Zone” and select “**us-east-1b**”



Then enter our subnet IPv4 CIDR, we will use “**12.0.2.0/24**”



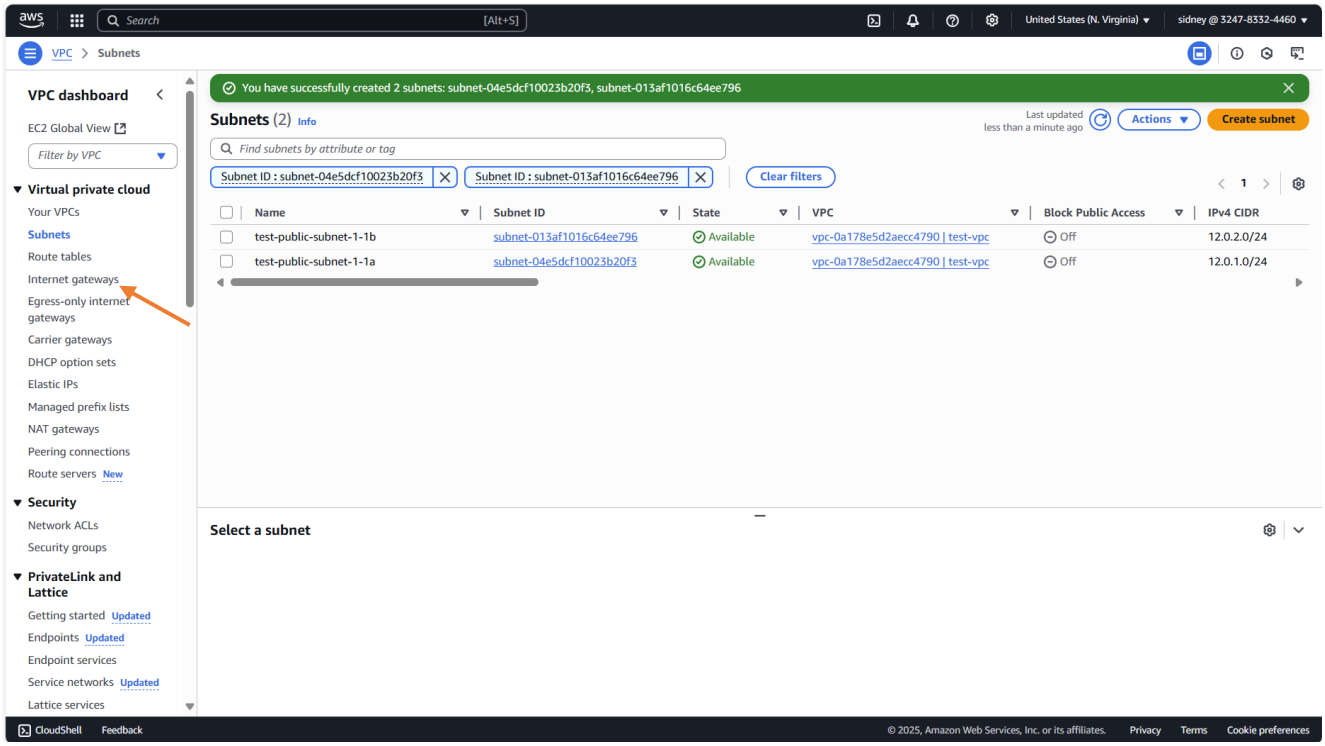
Click on "**create subnet**"



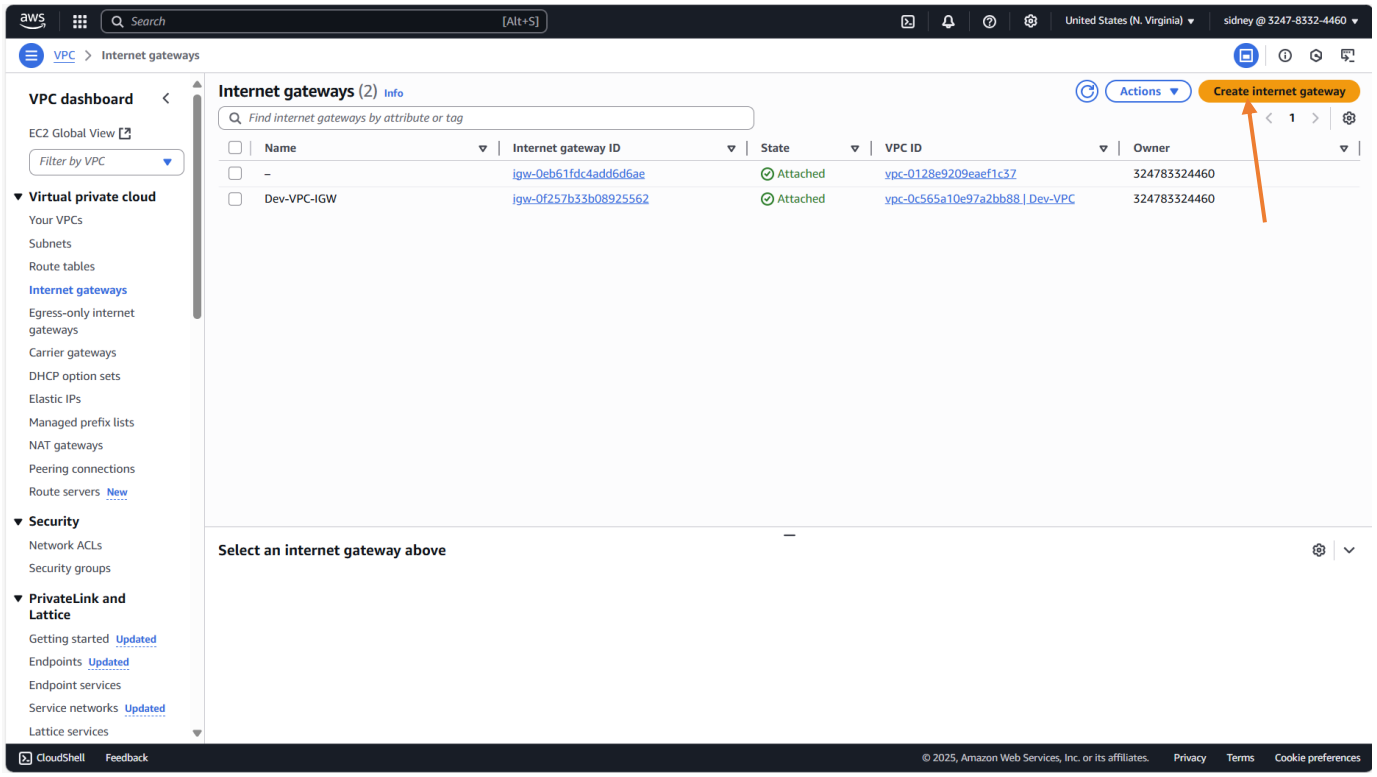
You can see that both subnets have been created.

STEP 3: Set up Internet Gateway

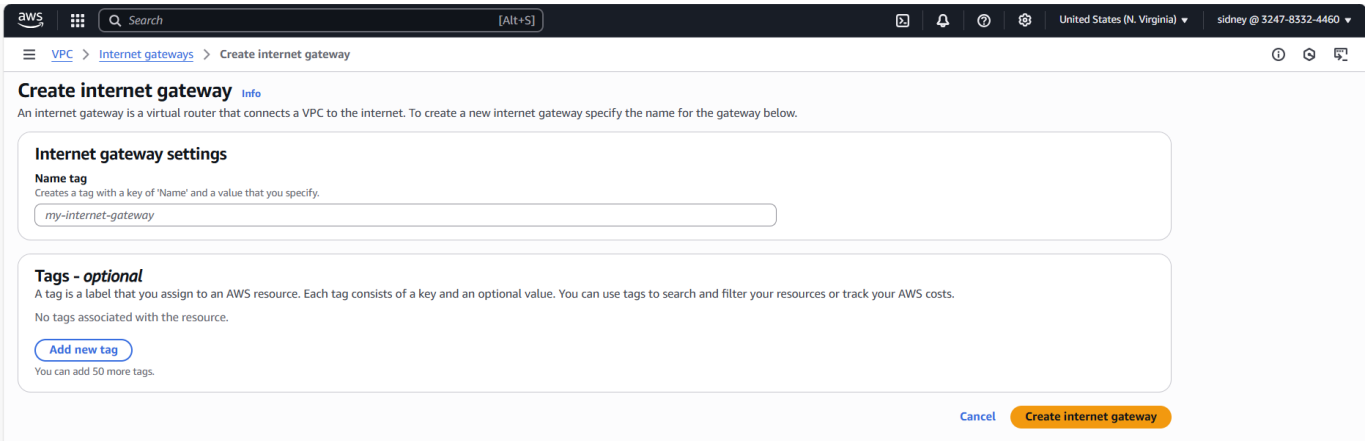
Now, we have to create our internet Gateway that will help us give our public subnet internet access.



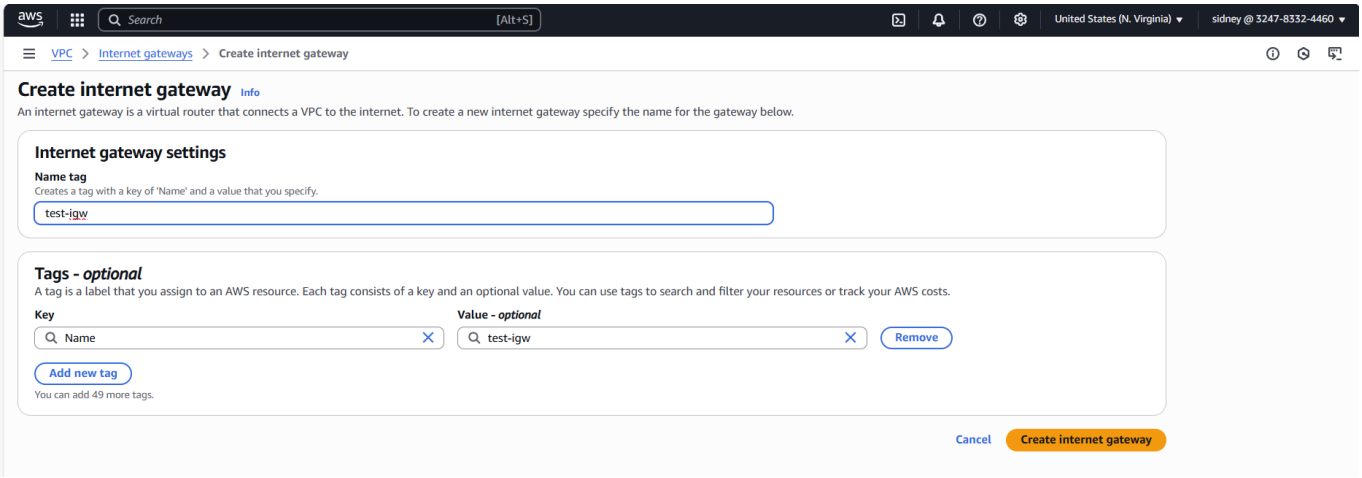
Click on "**Internet Gateways**"



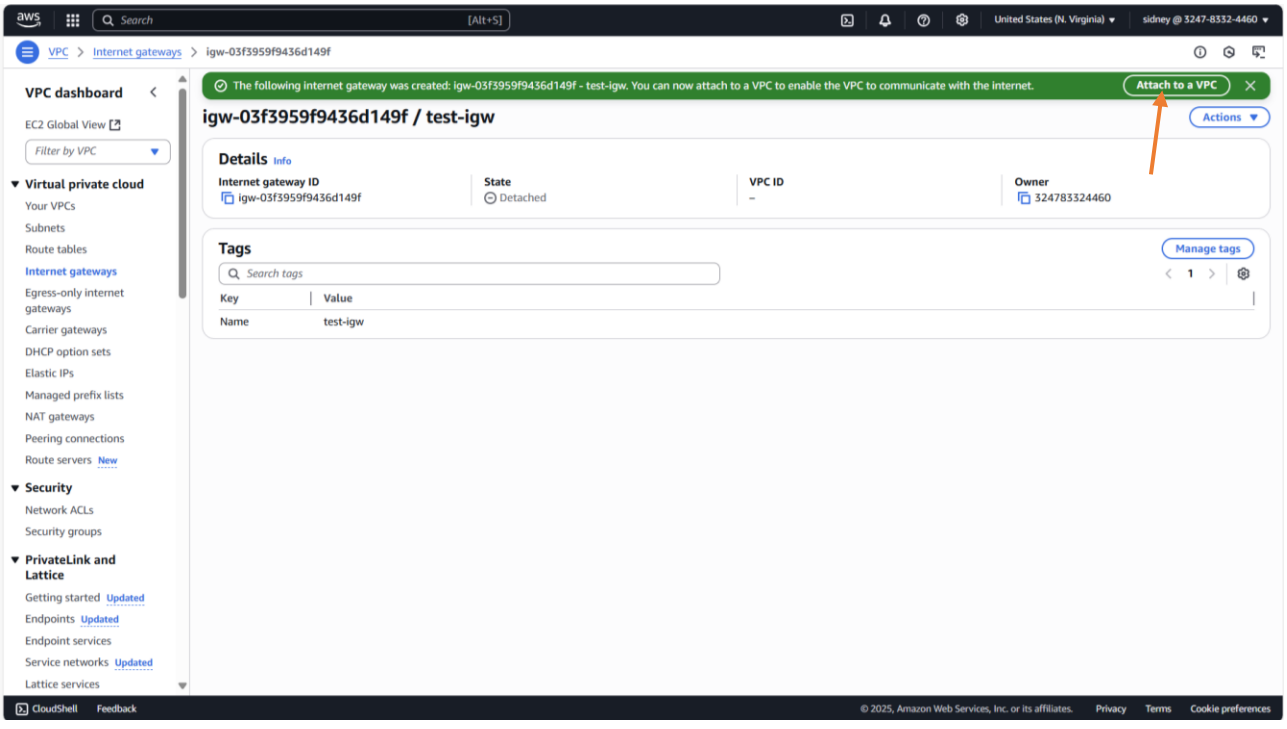
Click on "**Create Internet Gateway**"



Give the internet gateway a name. I will call it "**test-igw**"

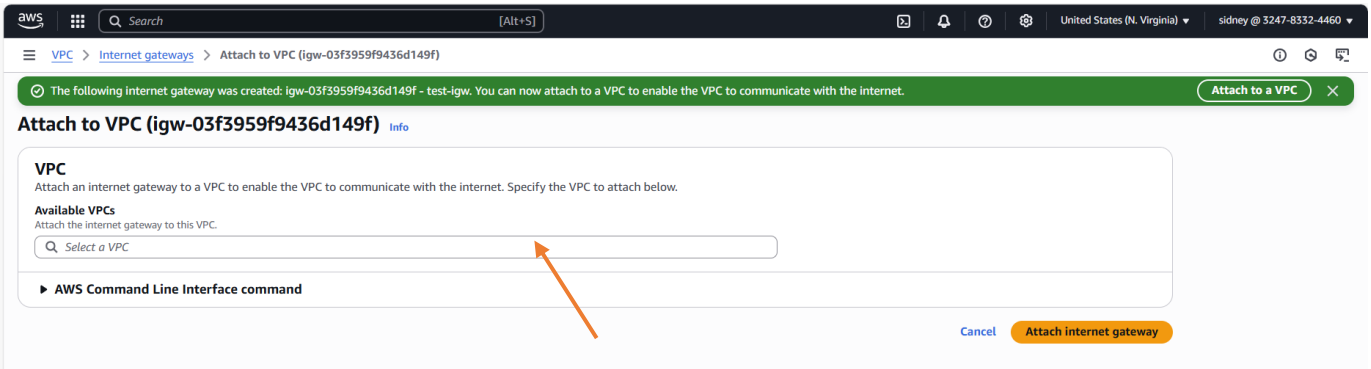


Then click on "**Create Internet Gateway**"

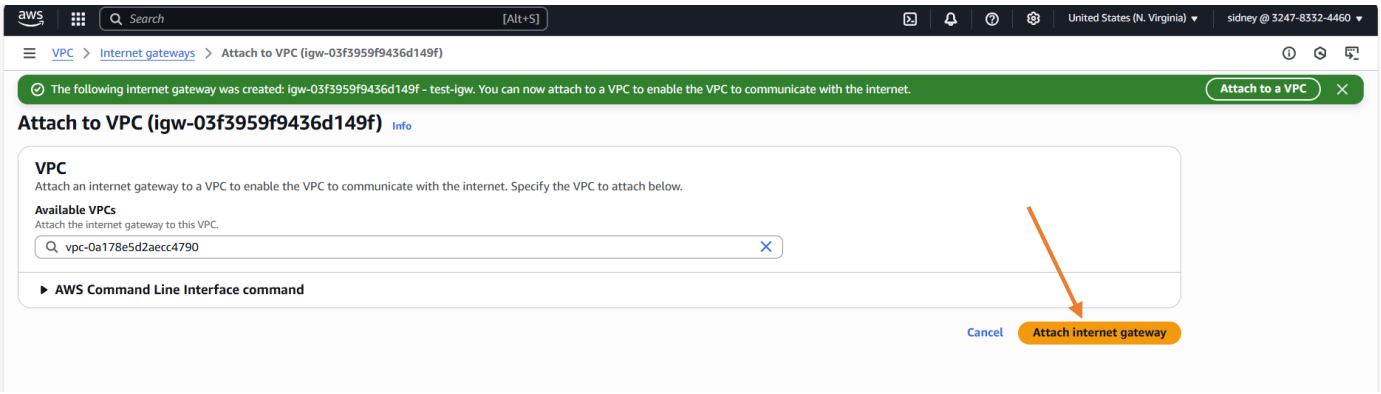


Our internet gateway has been created. The internet gateway is inside the VPC, so we have to associate it with our VPC.

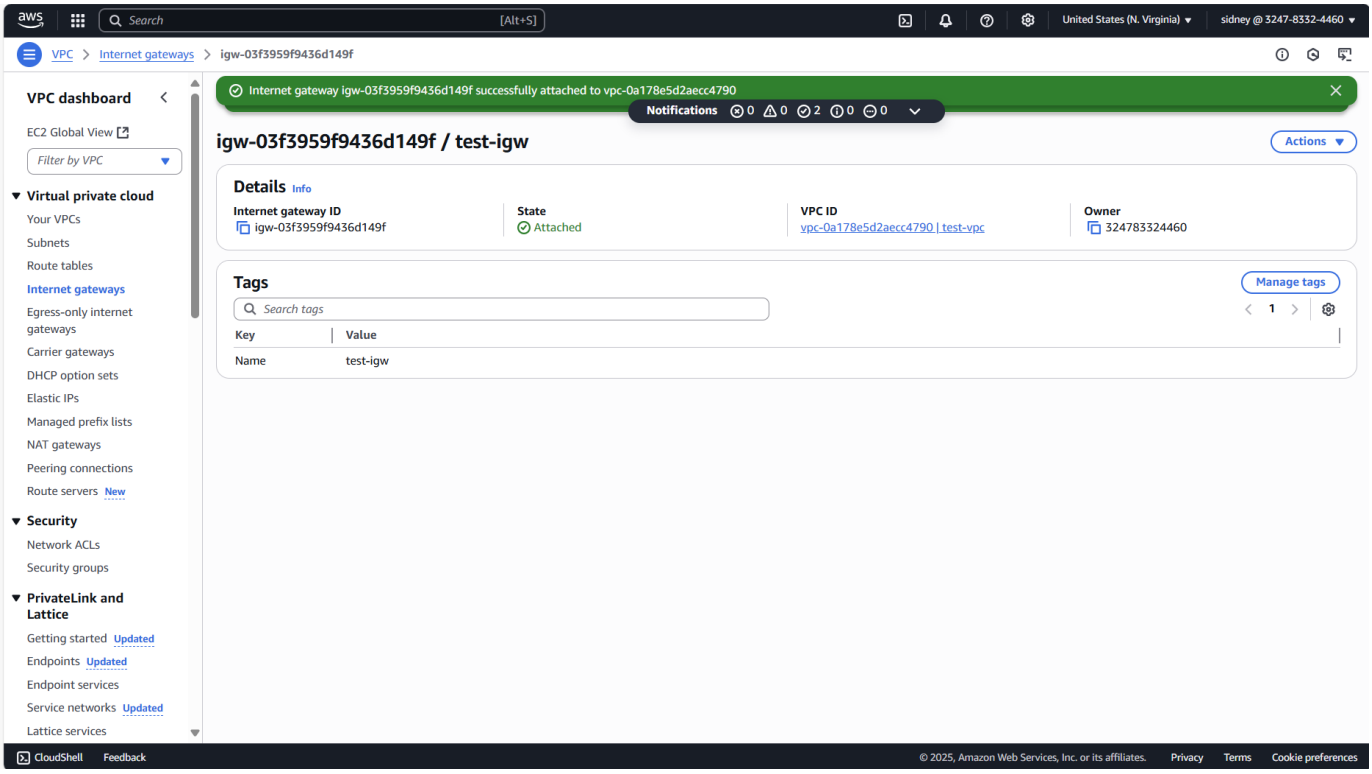
Click on "**Attach to a VPC**"



Click on "**Available VPCs**" and select our VPC



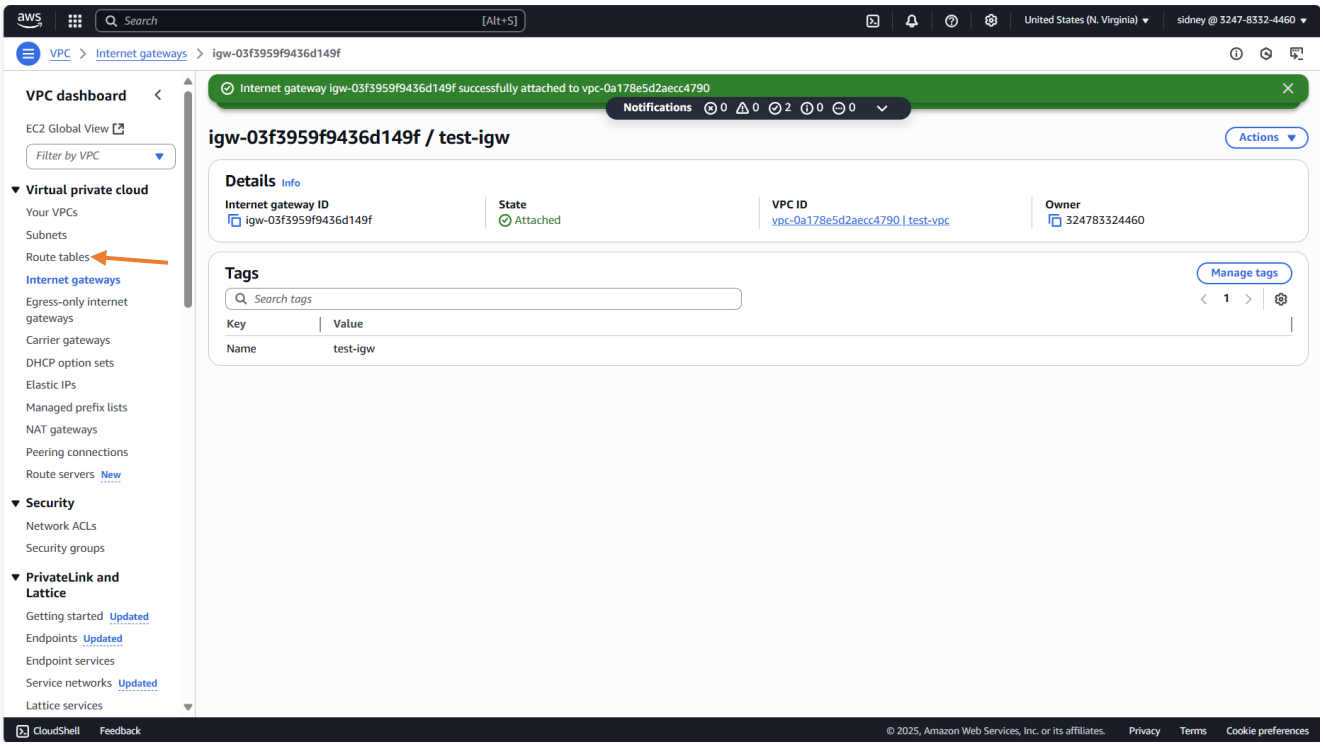
Click on "**Attach Internet Gateway**"



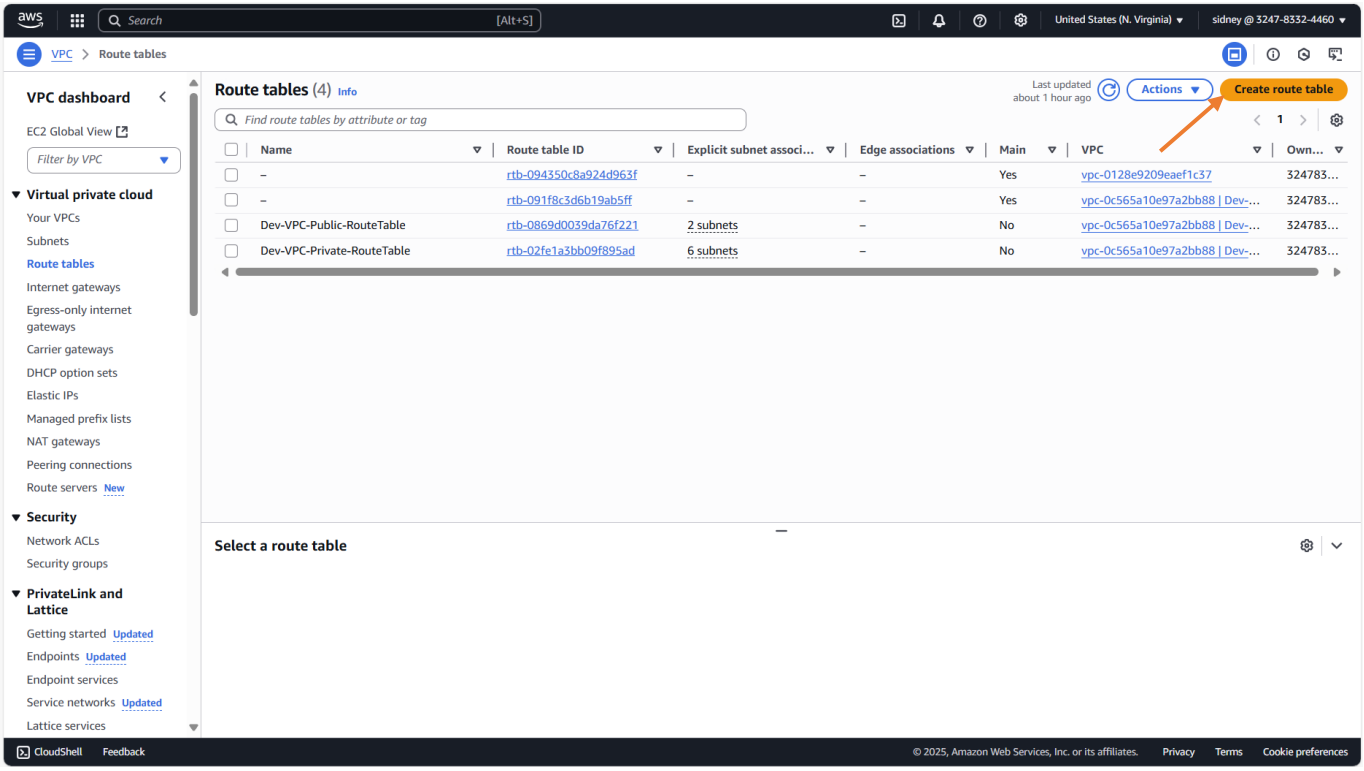
The internet Gateway has been attached to our VPC

STEP 4: Set up our Route Table

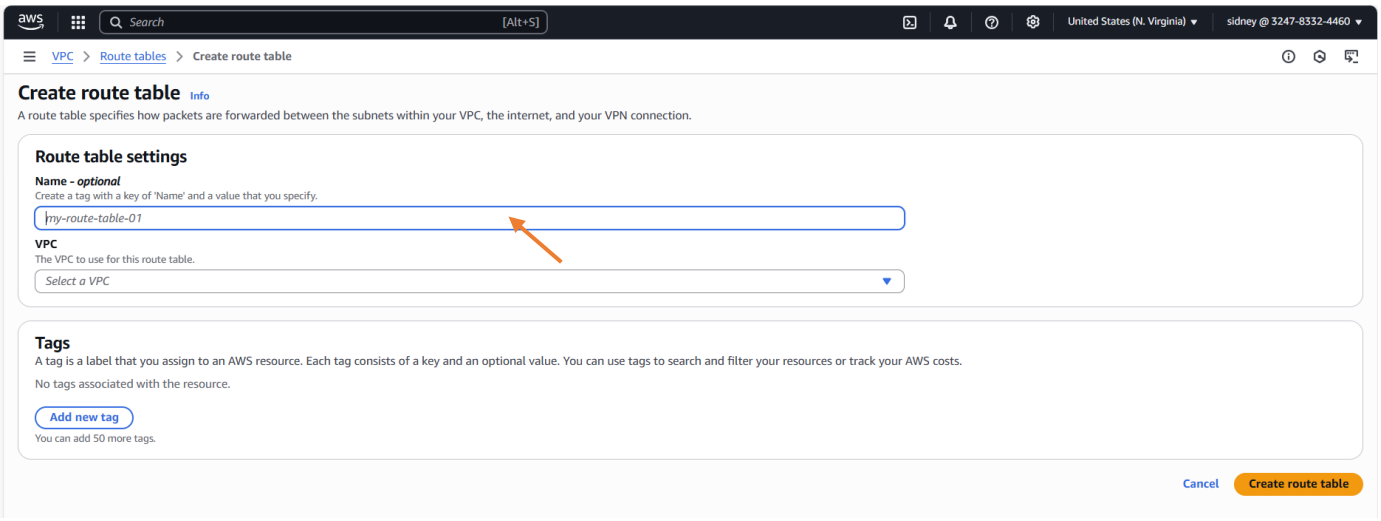
The next resource we have to create is our Route table.



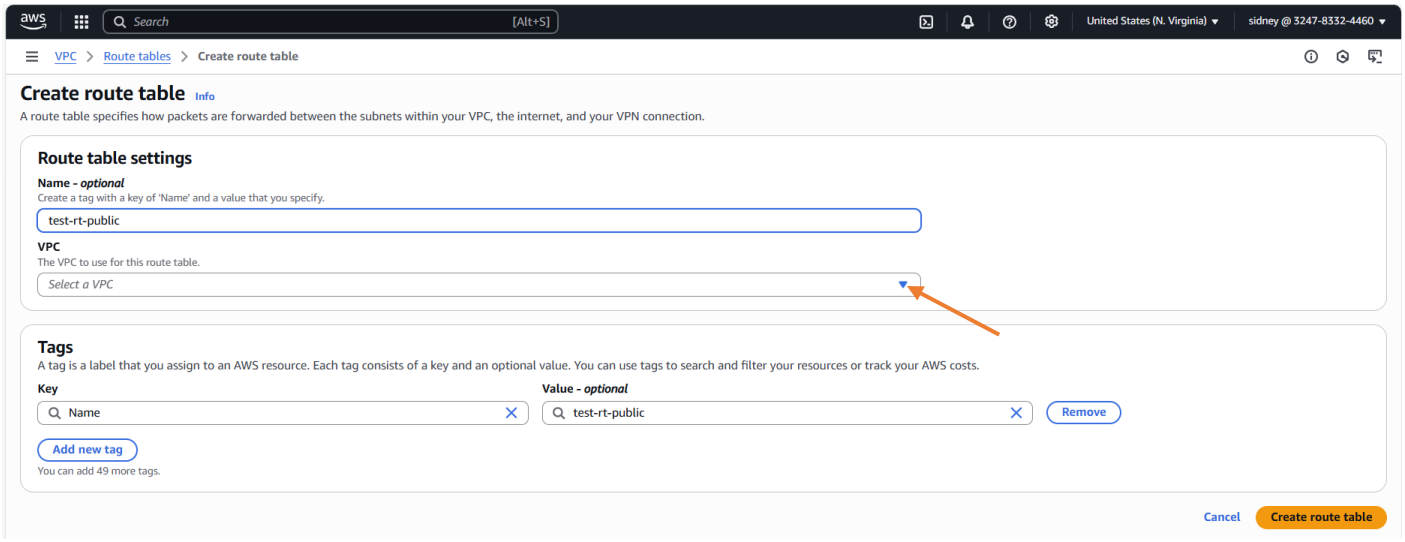
Click on "**Route Tables**"



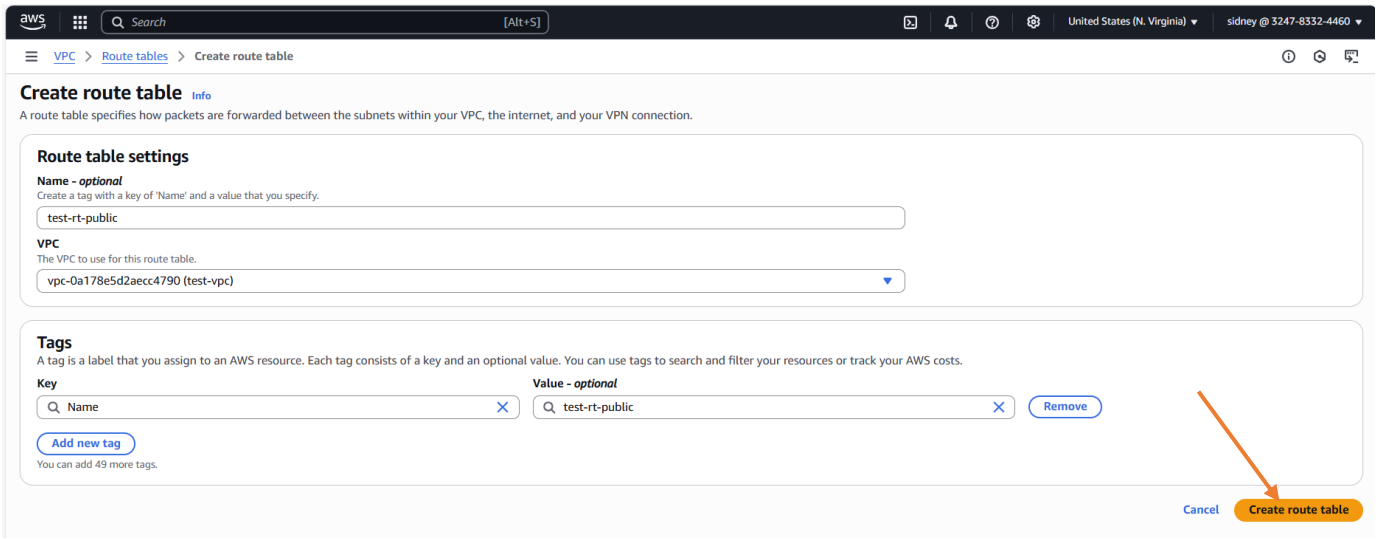
Click on “**Create Route Table**”



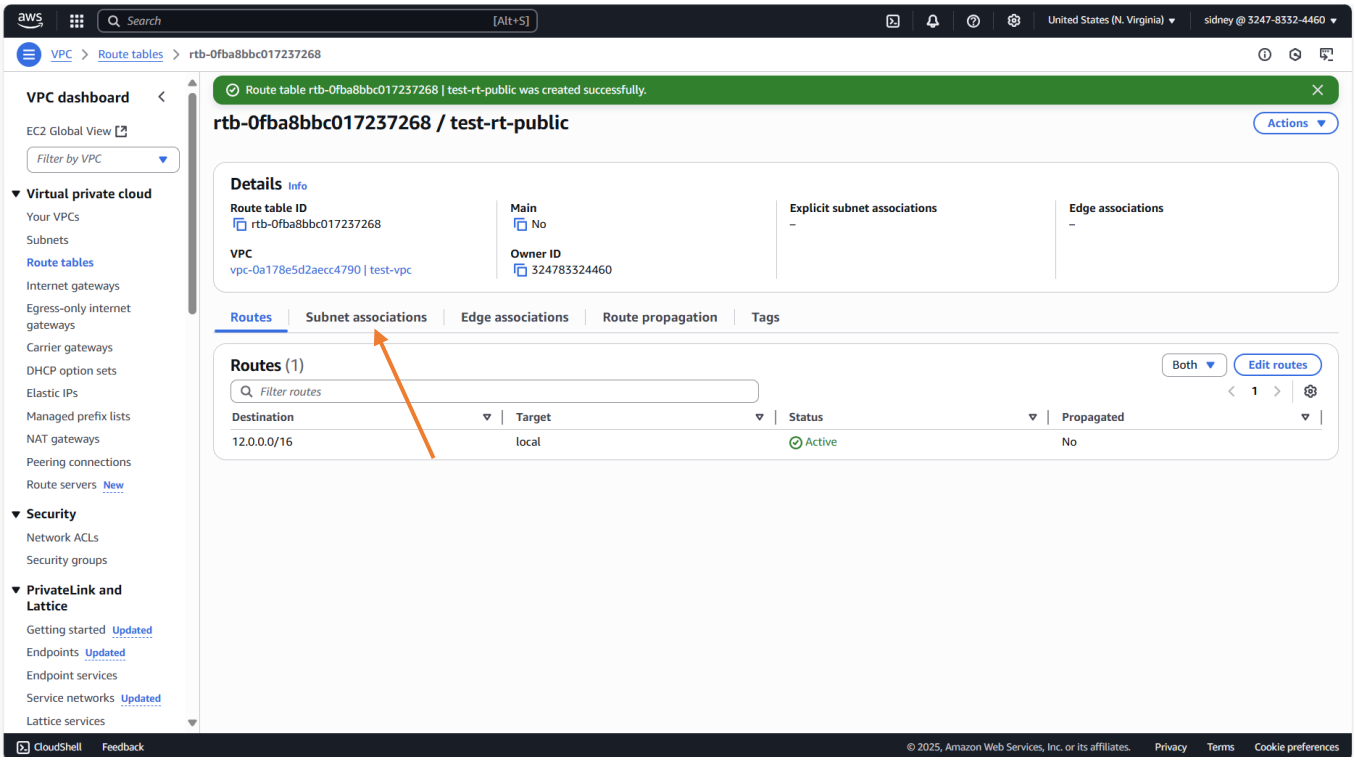
Give the Route Table a name, I will call it “**test-rt-public**”



Click on the drop down on "**VPC**" and select our VPC

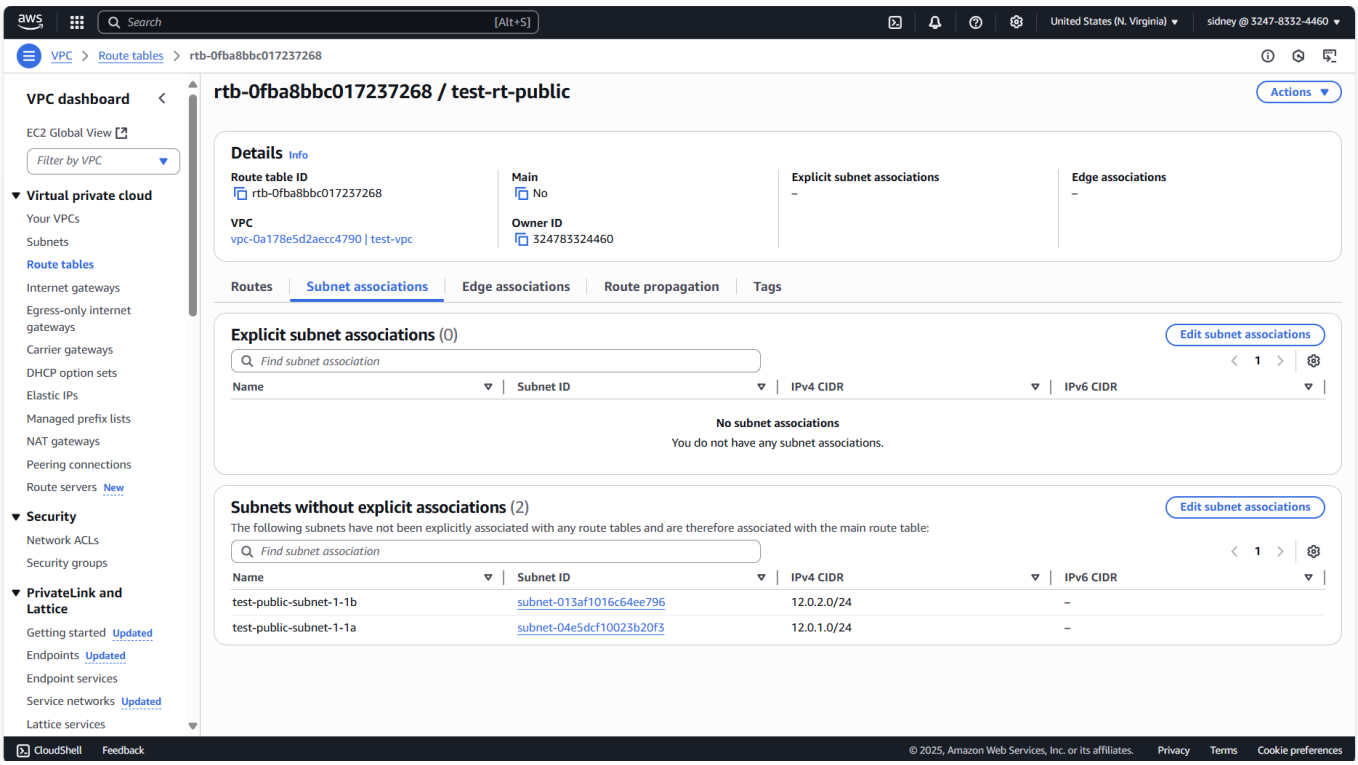


Click on "**Create Route Table**"

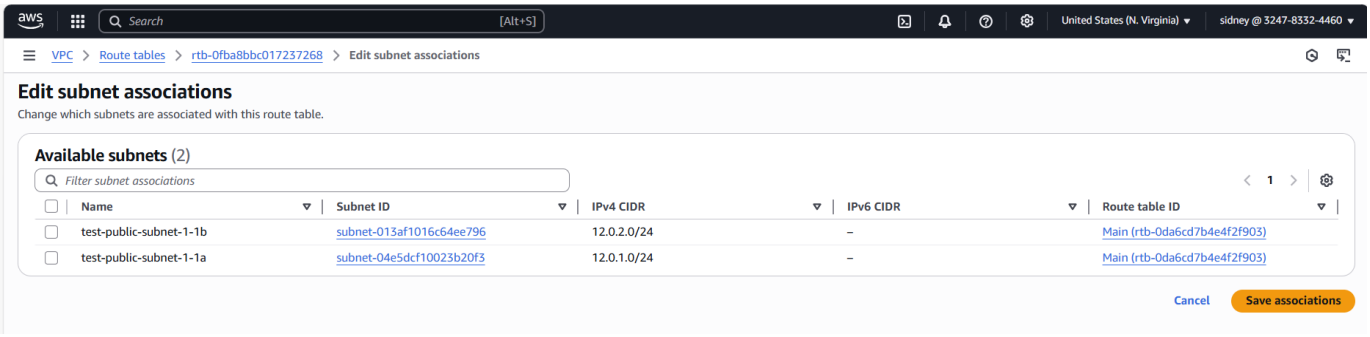


The VPC has been created. The Route Table is inside our VPC, so we have to associate the public subnet with our Route Table.

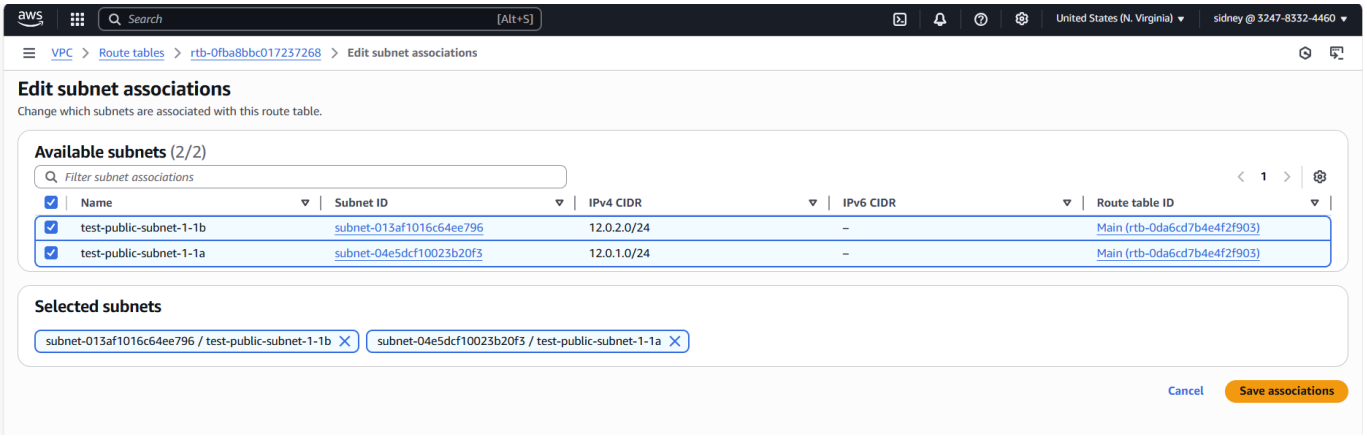
Click on "**Subnet Association**"



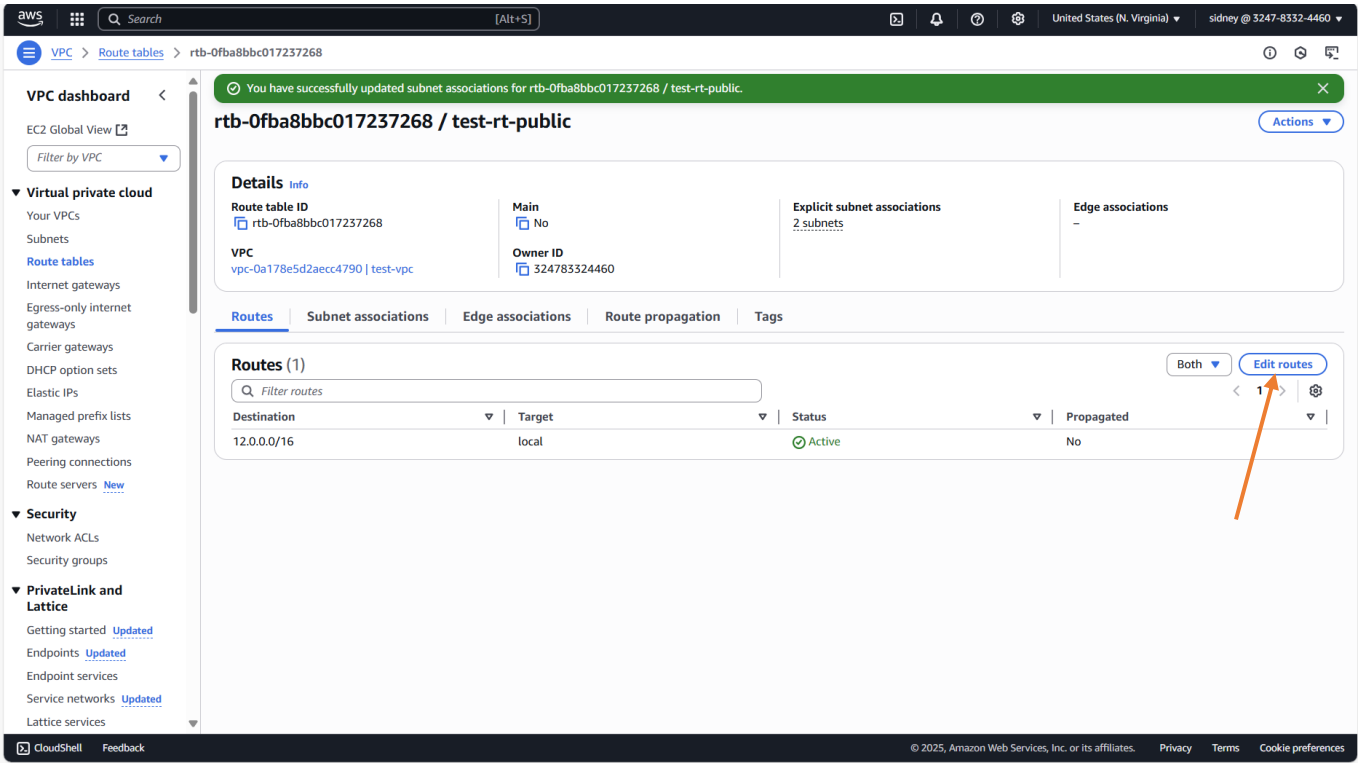
Click on "**Edit Subnet Associations**"



Select both public subnets

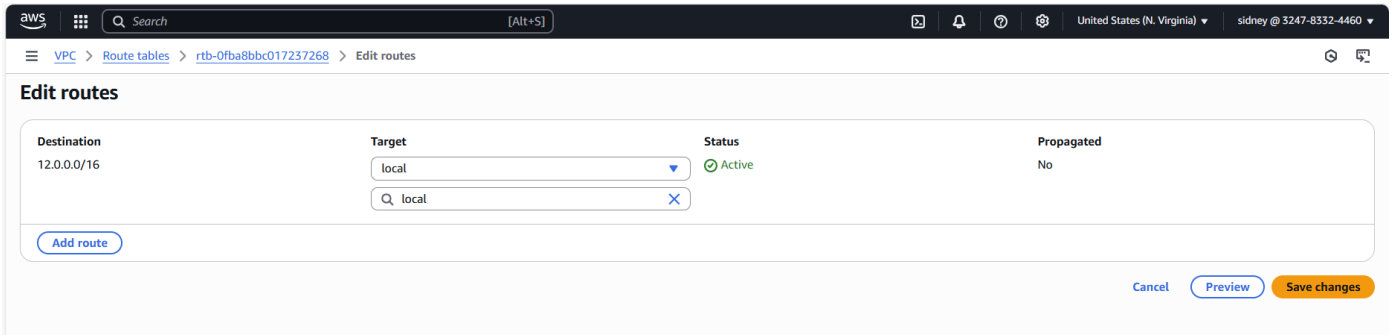


Click on “**Save Associations**”

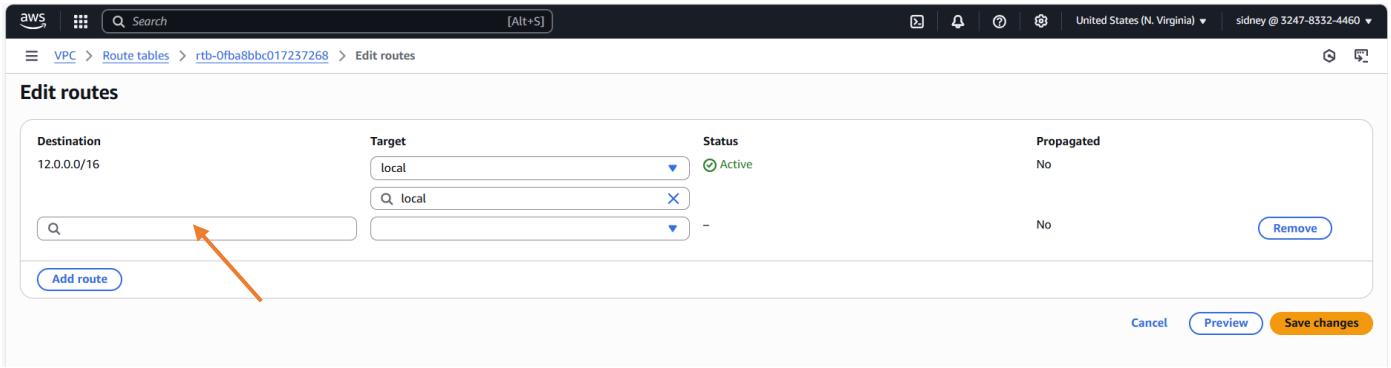


We have now associated our subnet with our Route Table. But we also have to attached our internet gateway to our subnet, so that the public subnet can access the internet.

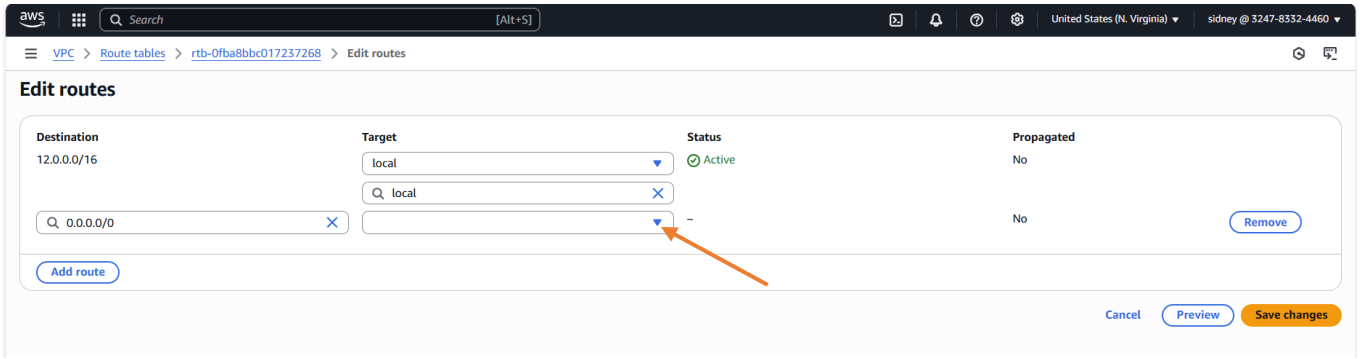
Click on “**Edit Route**”



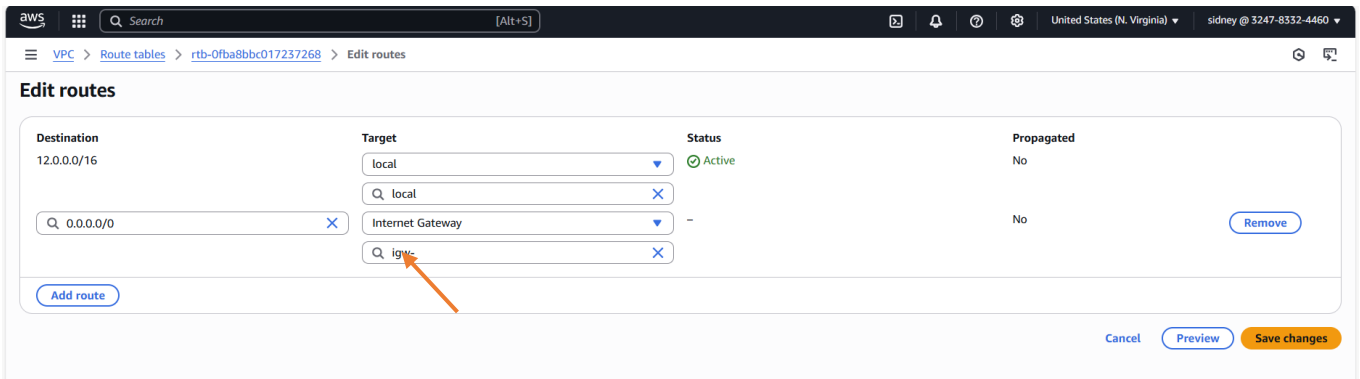
Click on “**Add Route**”



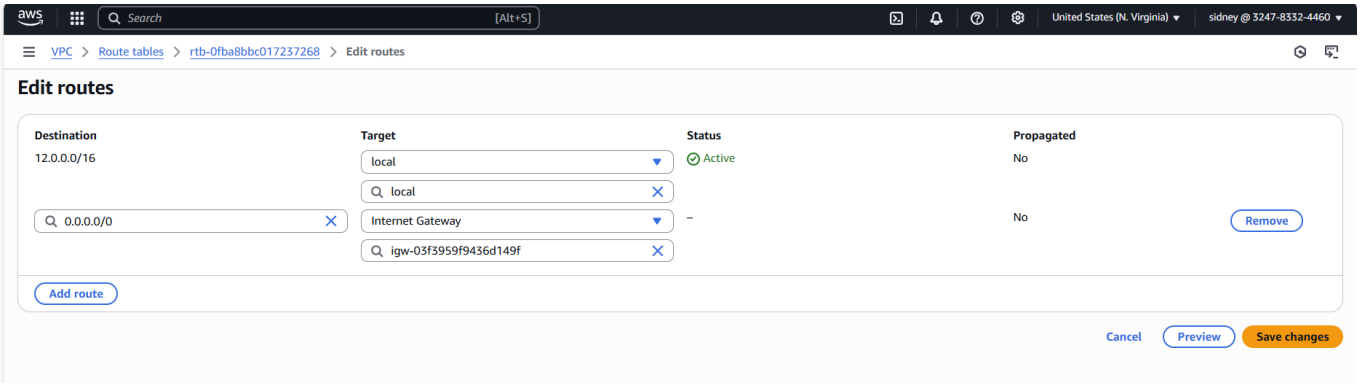
Click and select “**0.0.0.0/0**”



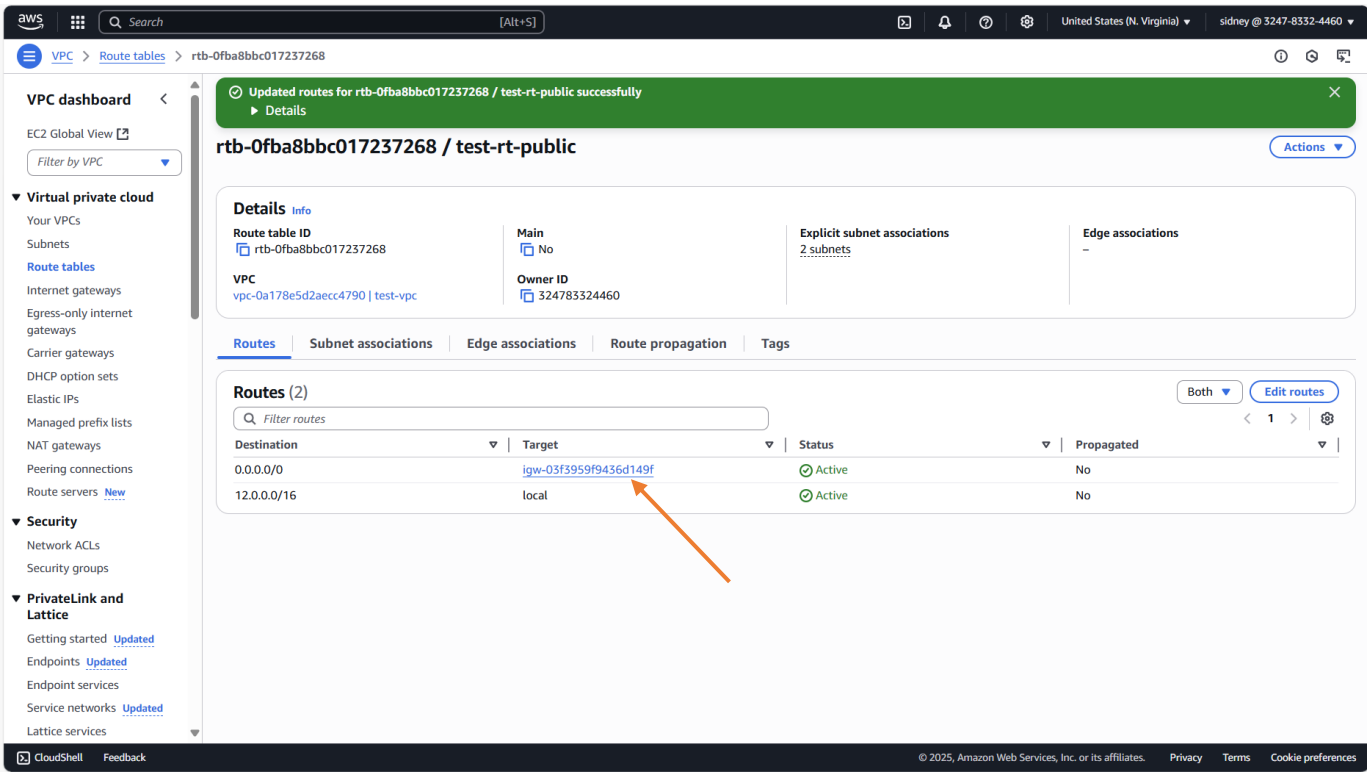
Click on the drop down and select “**Internet Gateway**”



Click on “**igw**” and select our Internet Gateway



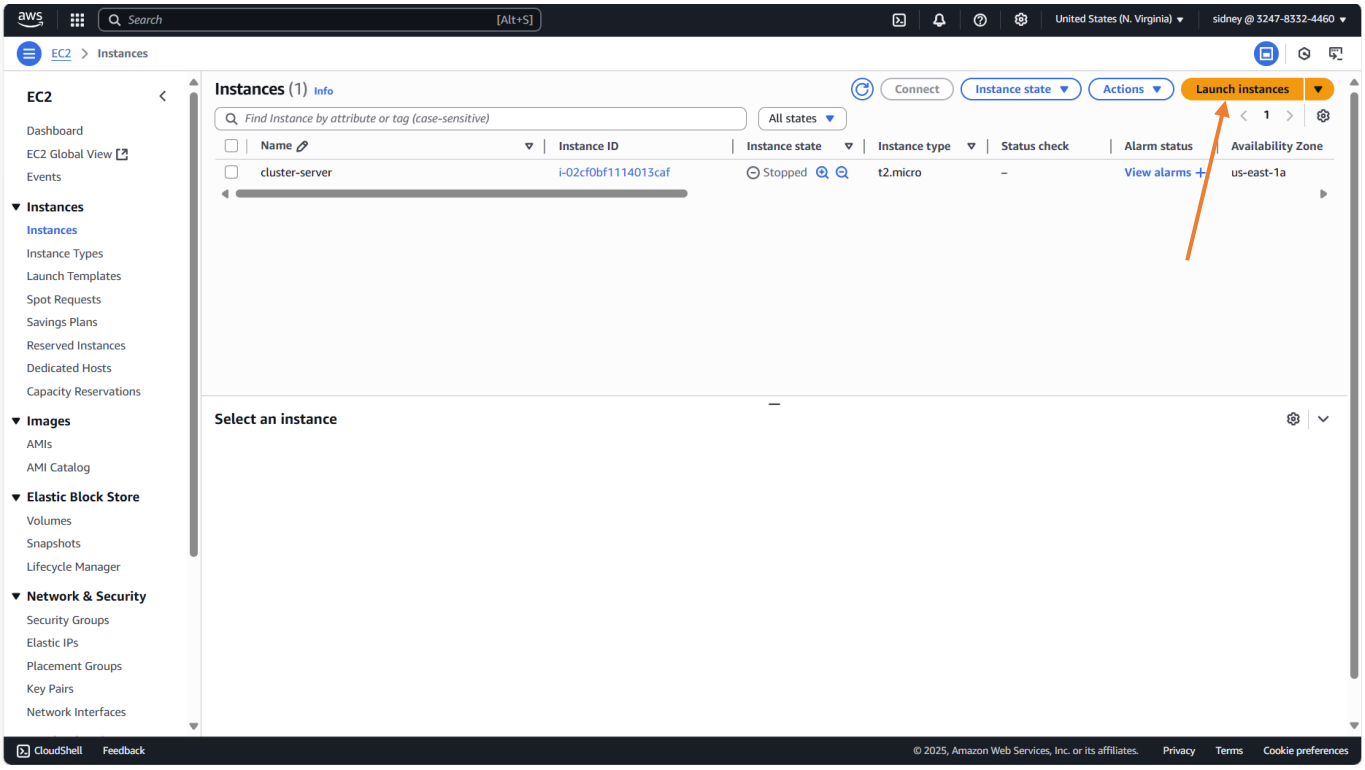
Click on “**Save Changes**”



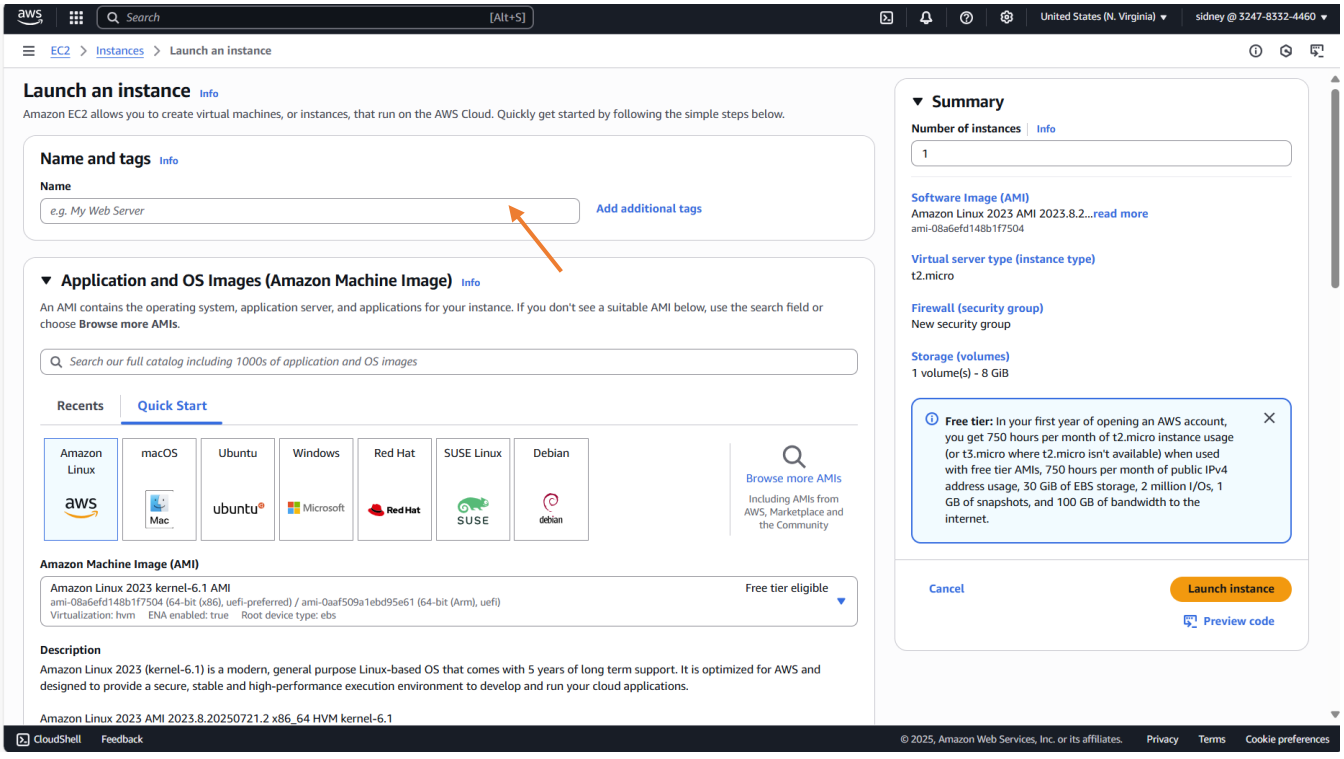
You can see that our internet gateway has been attached to our route table.

STEP 5: Launch the EC2 Instance

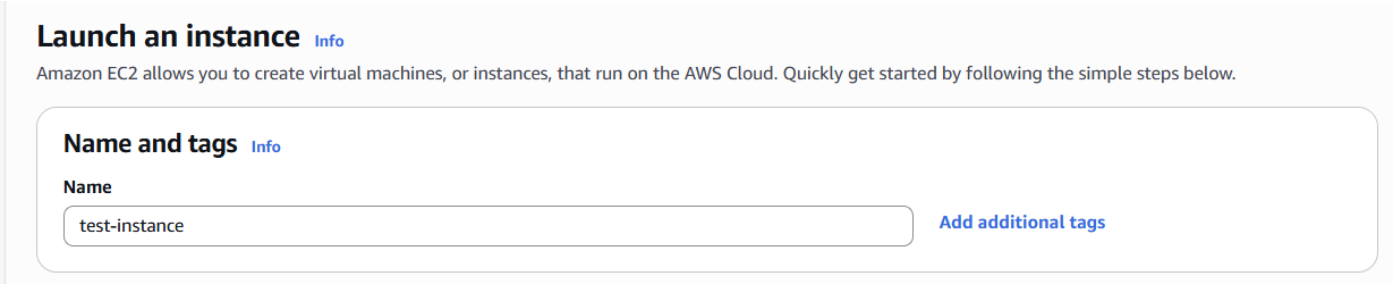
The next resource we have to create is the EC2 instance. Go to EC2 dashboard on AWS Management Console.



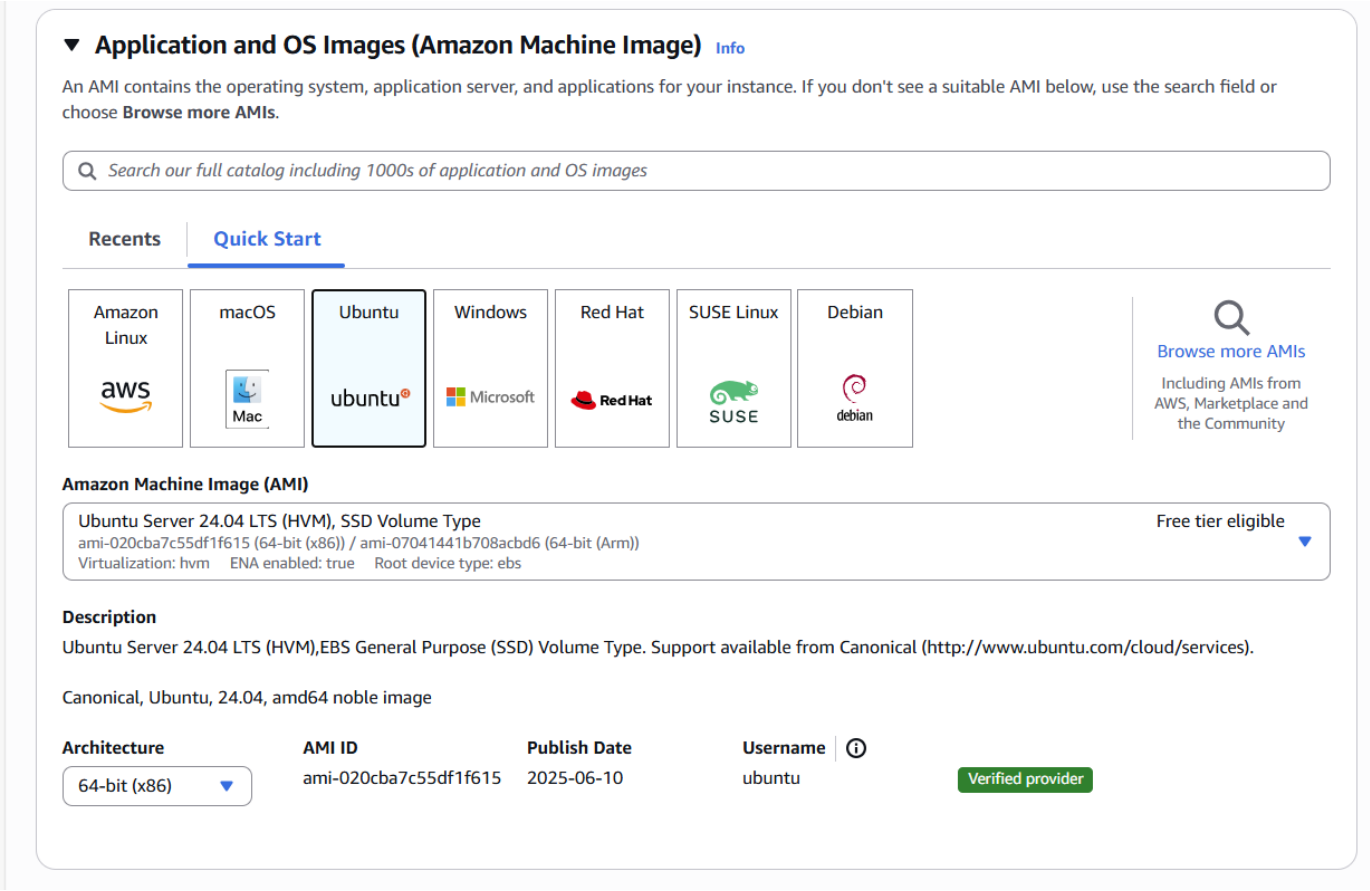
Click on "**Launch Instance**"



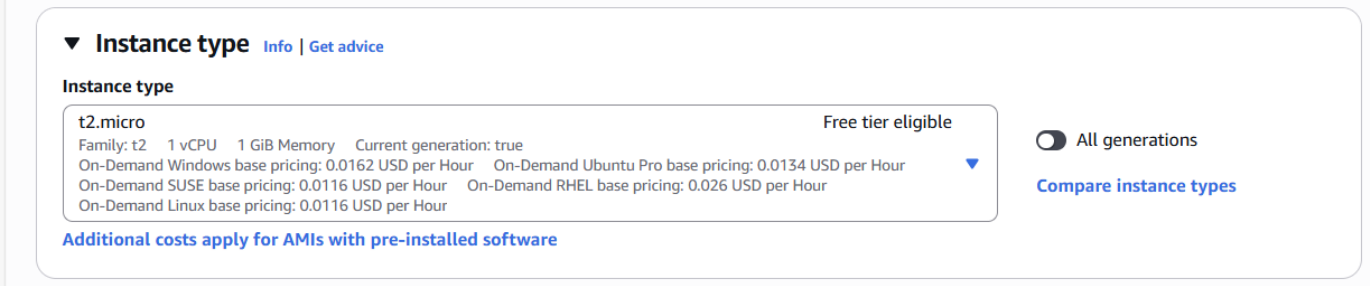
We will call the Instance “**test-instance**”



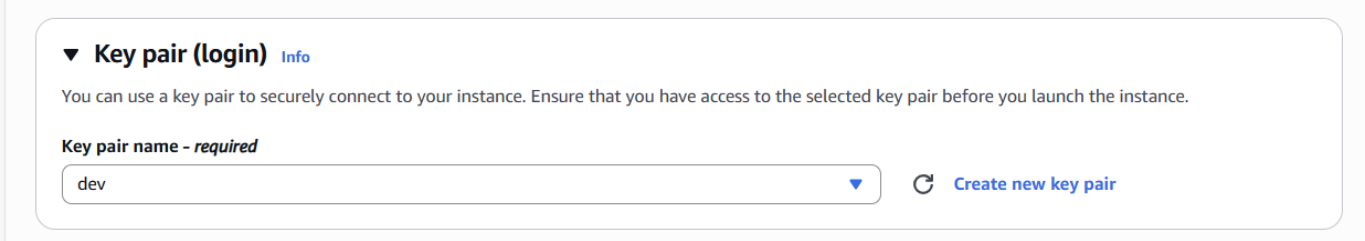
On “**AMI**”, we will use “**Ubuntu**”



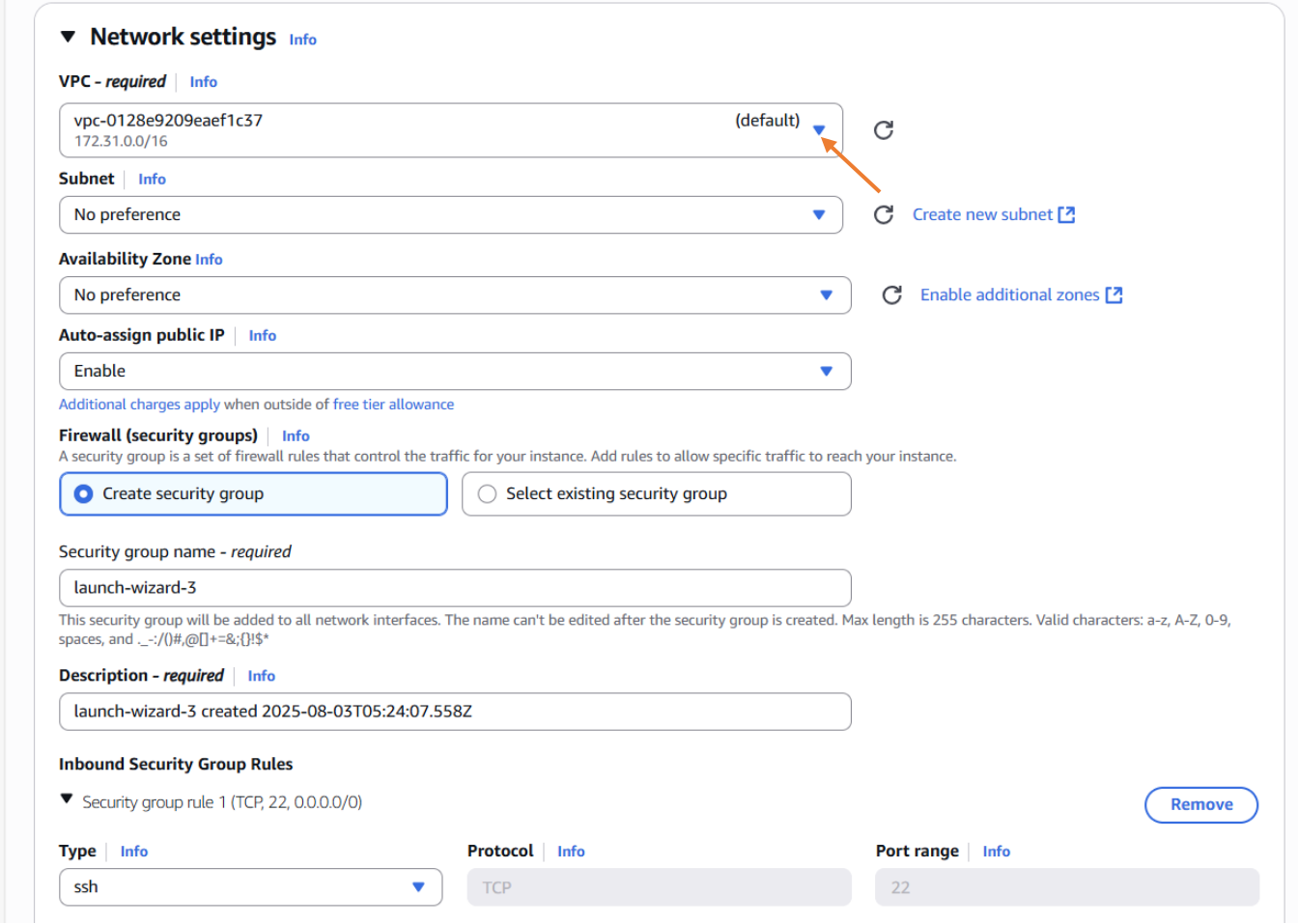
On “**Instance Type**”, select “**t2.micro**”



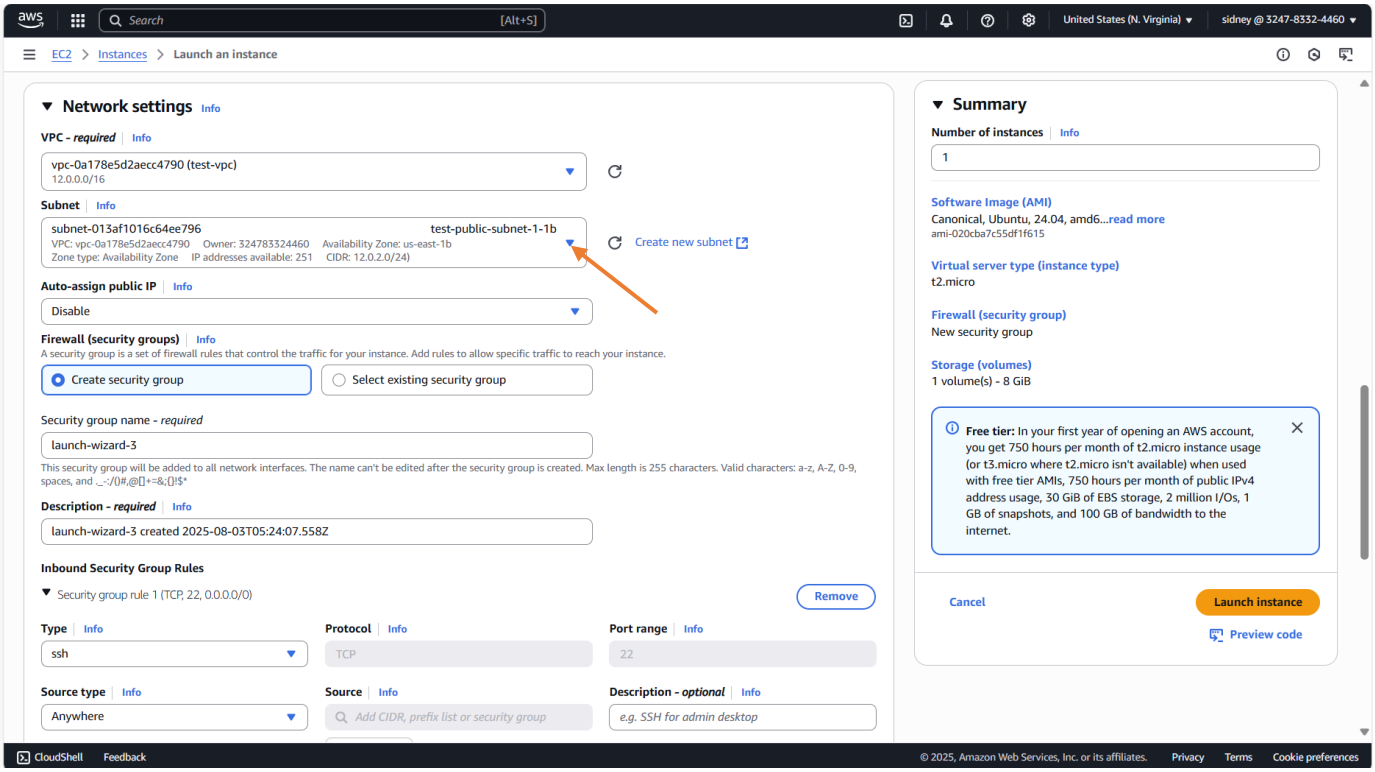
On “**Key Pair**”, I will select a key pair I had created previously. Click on the drop down and select the key.



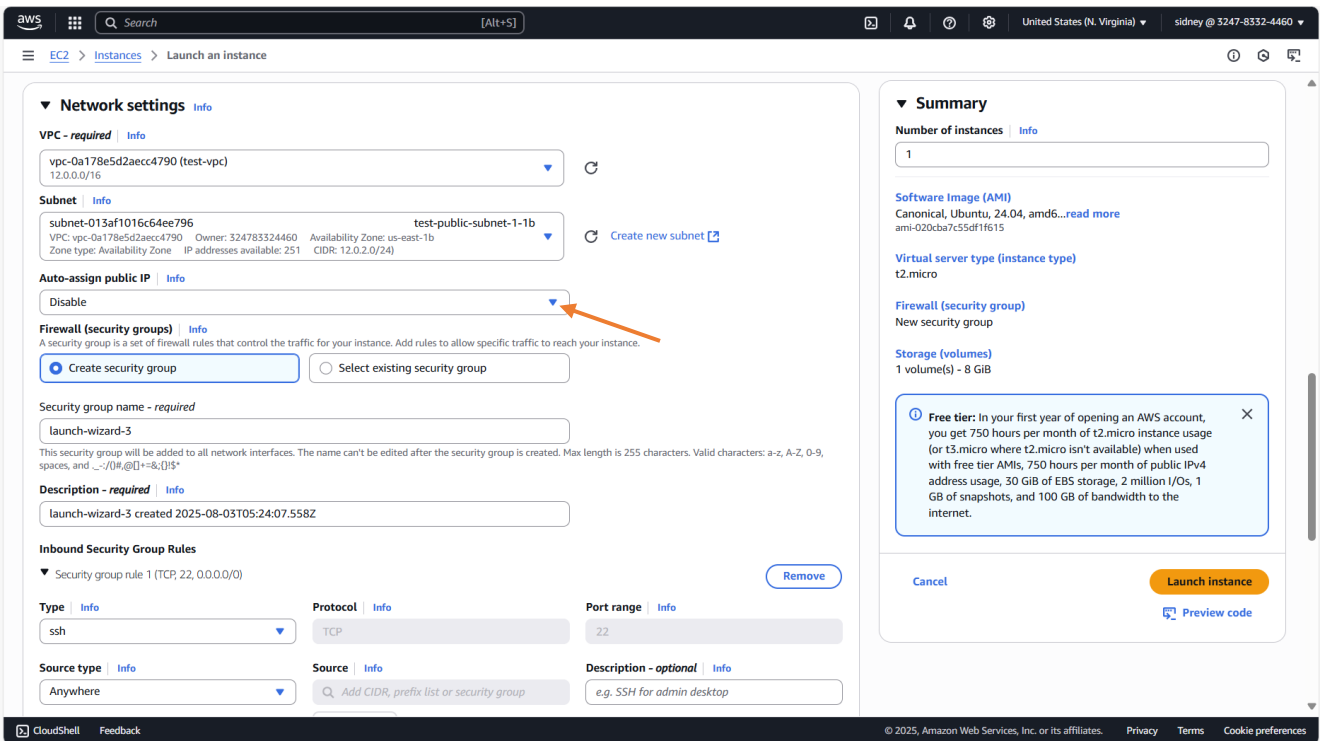
On “**Network Settings**”, click on “**Edit**”



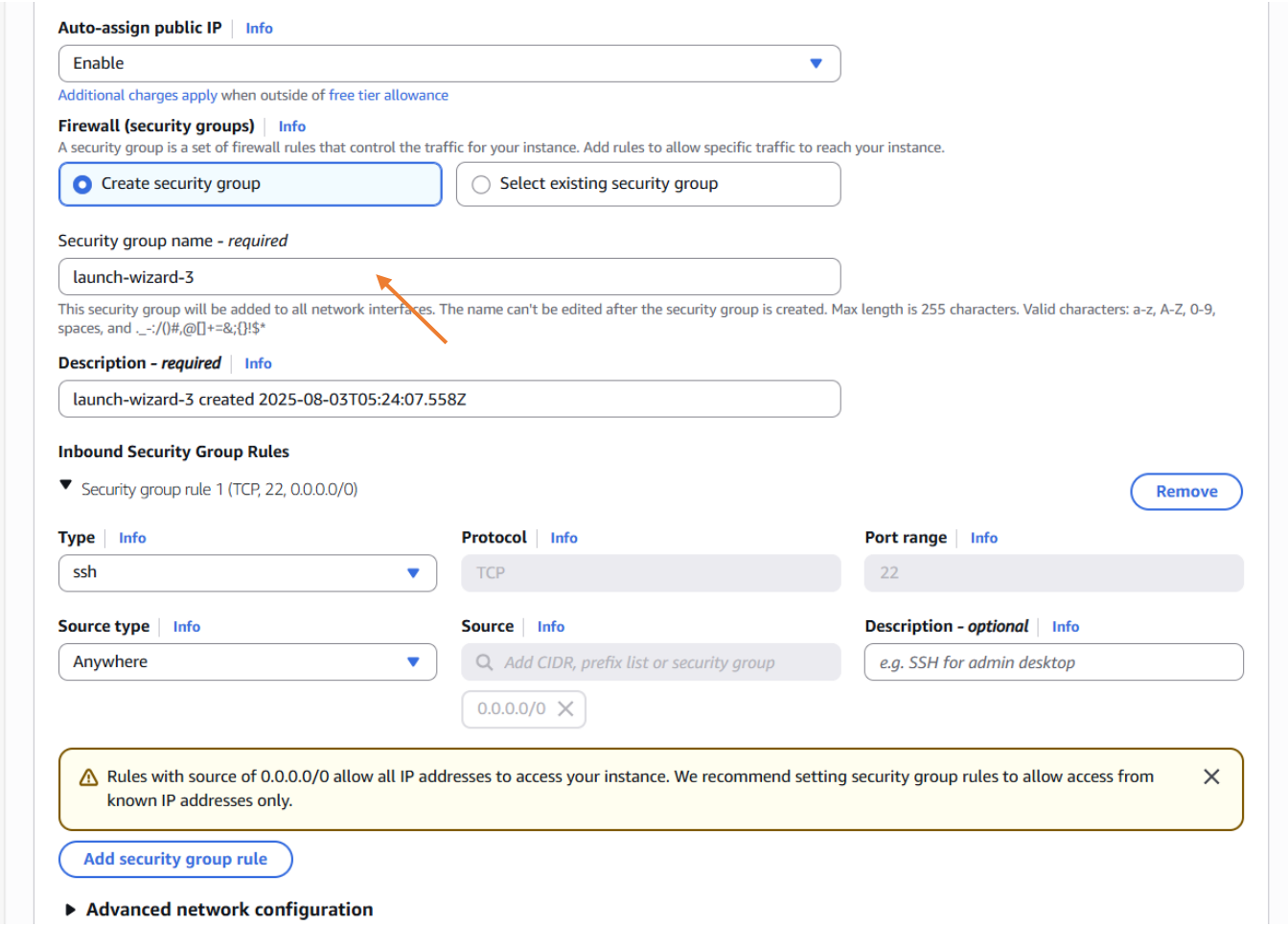
Click on the drop down and select our VPC



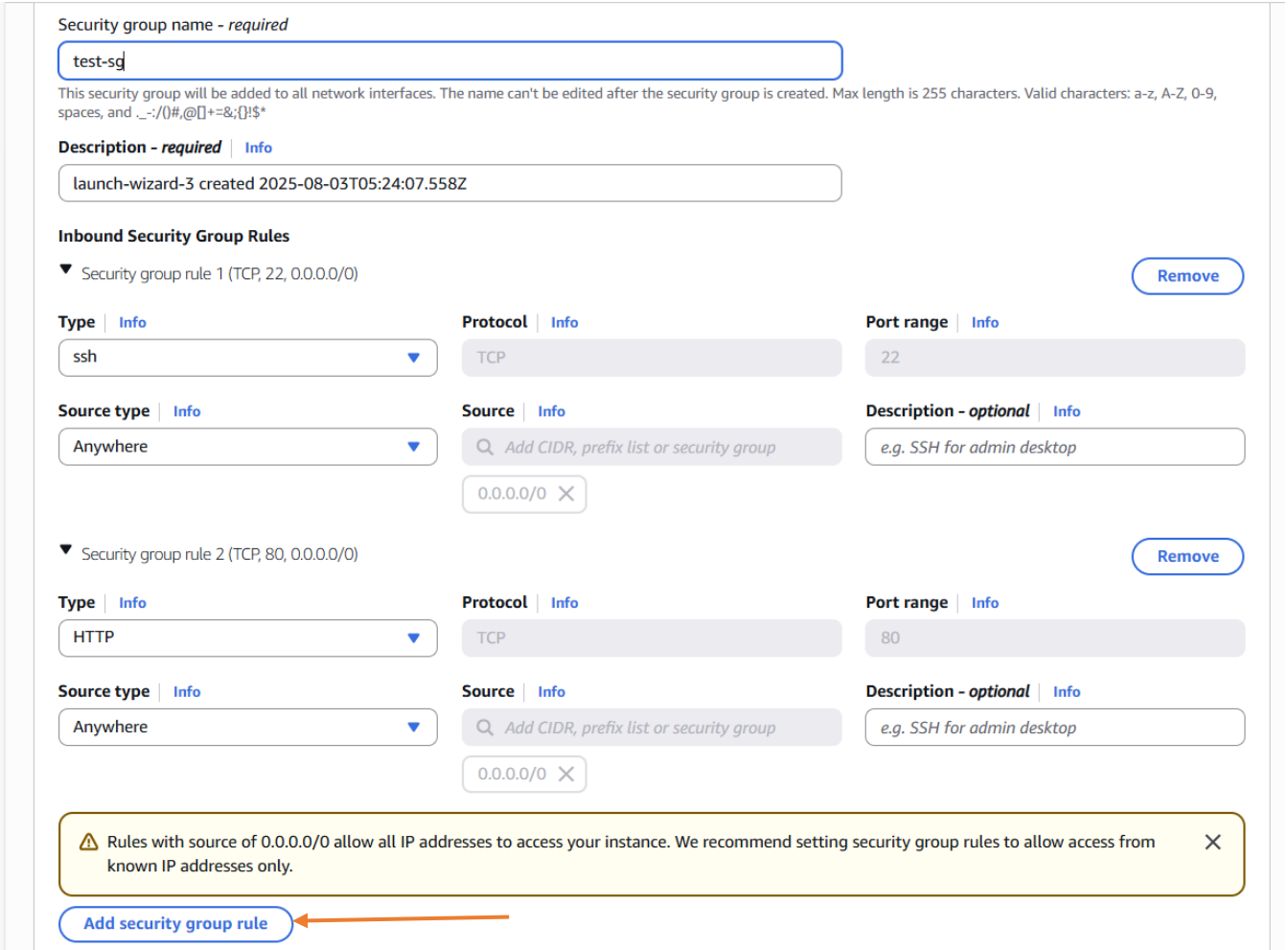
On "**Subnet**", select one of the public subnets we created



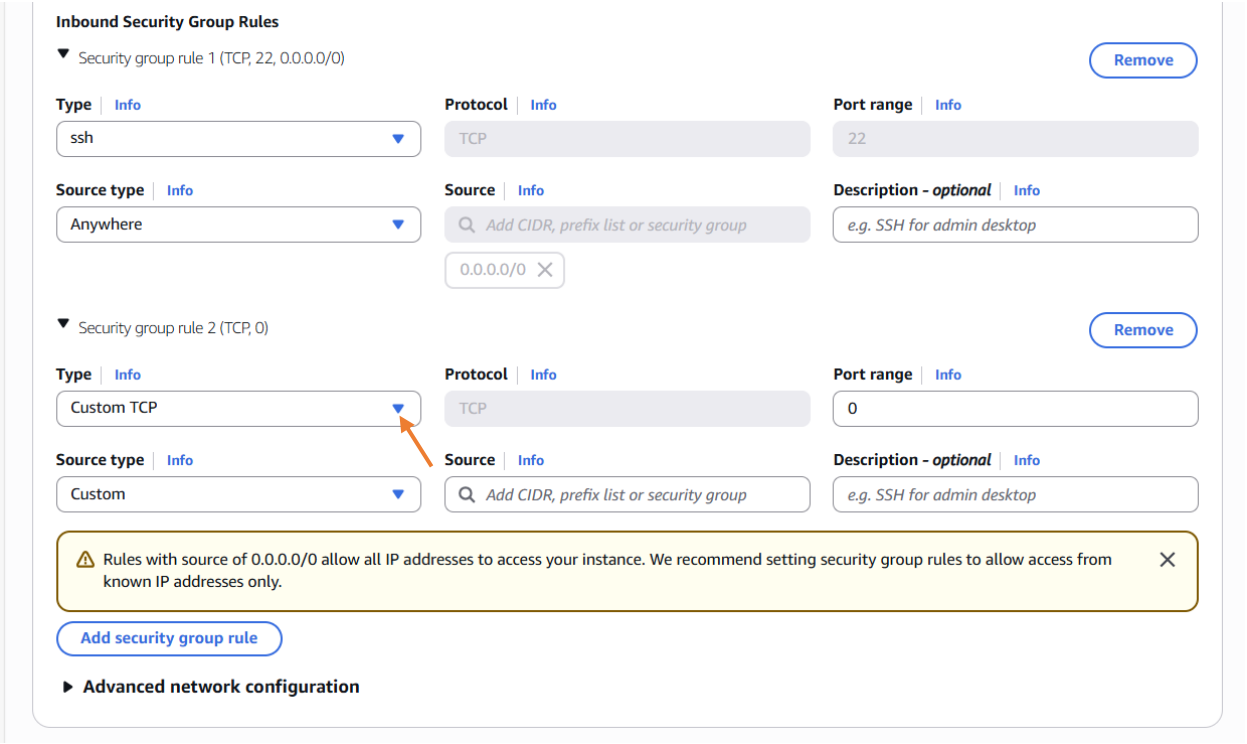
Click on the drop down on "**Auto-assign public IP**" and select "**Enable**"



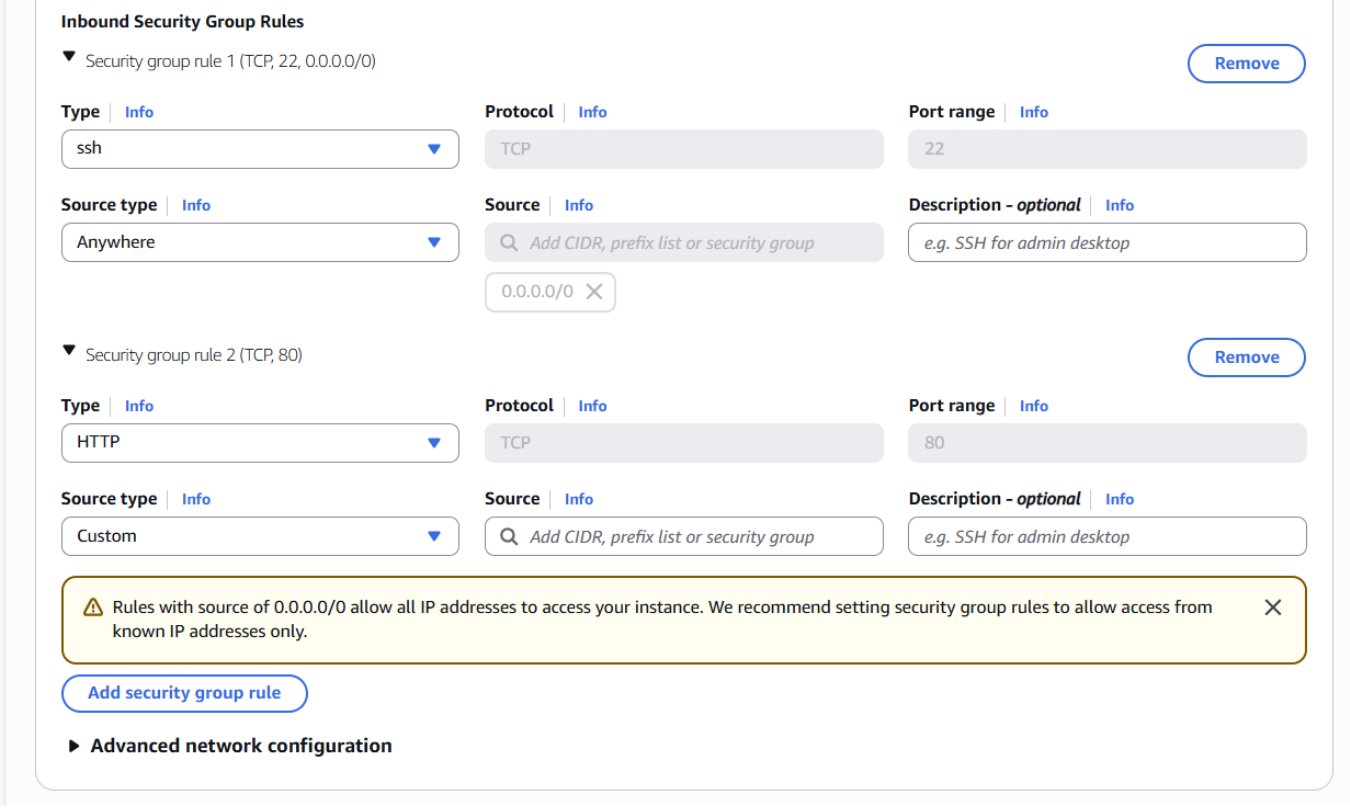
On "**Security Group Name**", we will call it "**test-sg**"



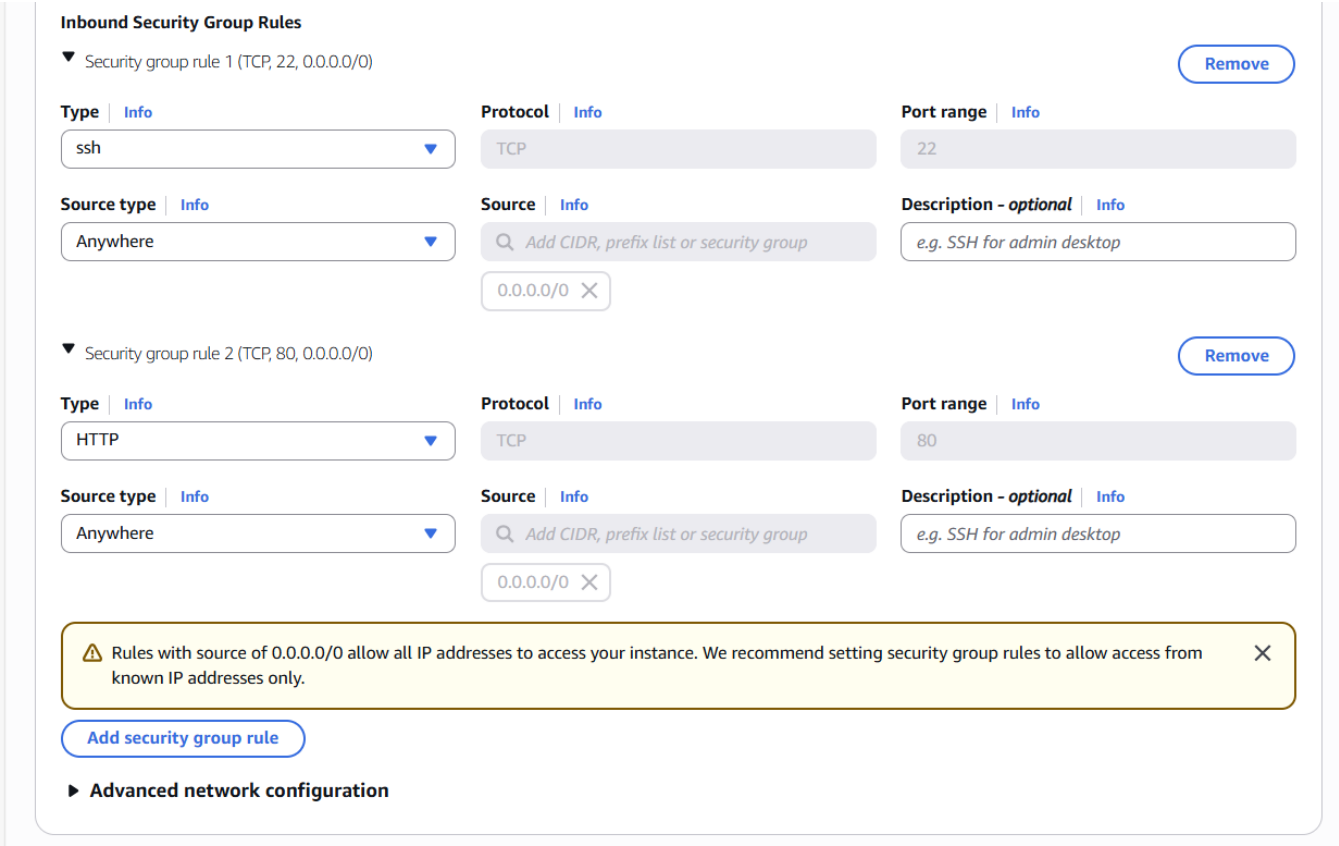
Then we have to create a security group with some rules. Click on "**Add security group rule**"



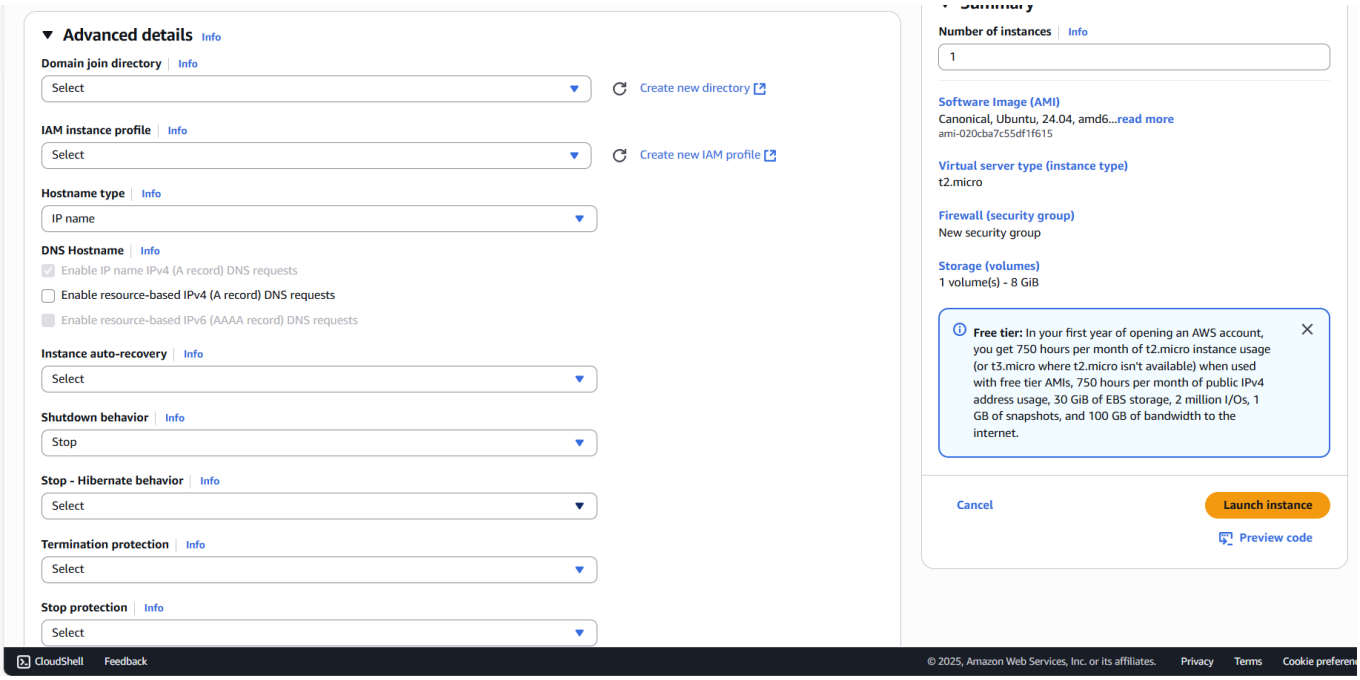
Click on the drop down on "**Type**" and select "**HTTP**"



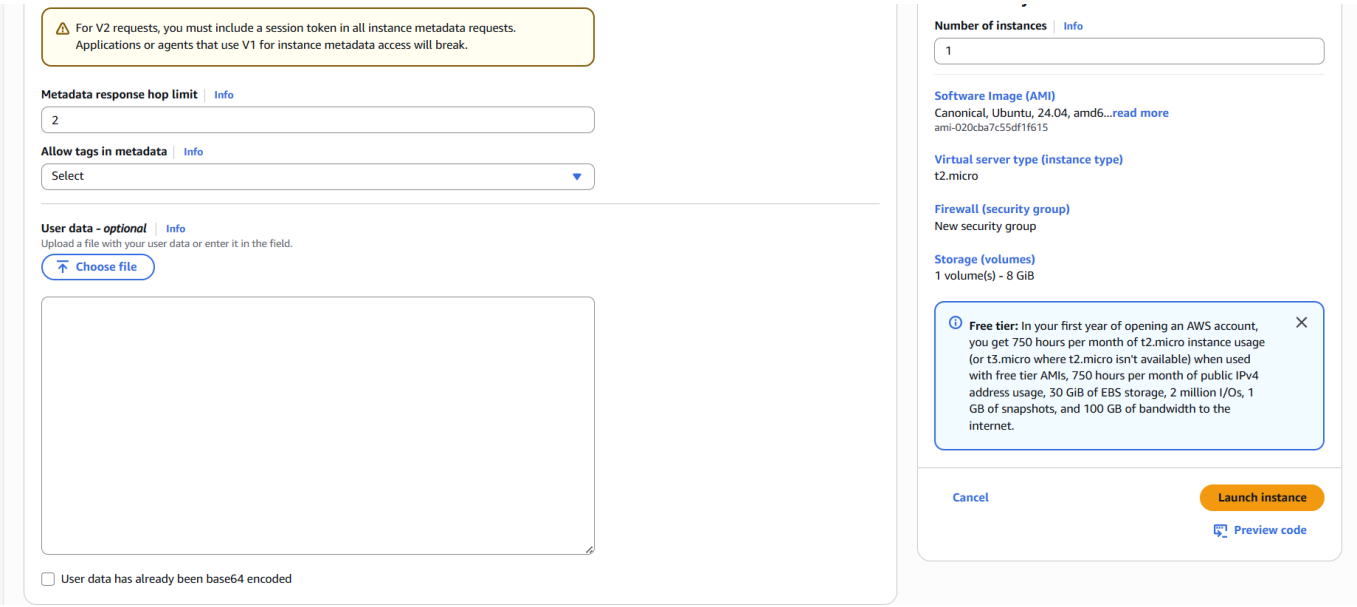
On "**Source Type**", select "**Anywhere**"



Scroll down to “**Advanced Details**” and click on it

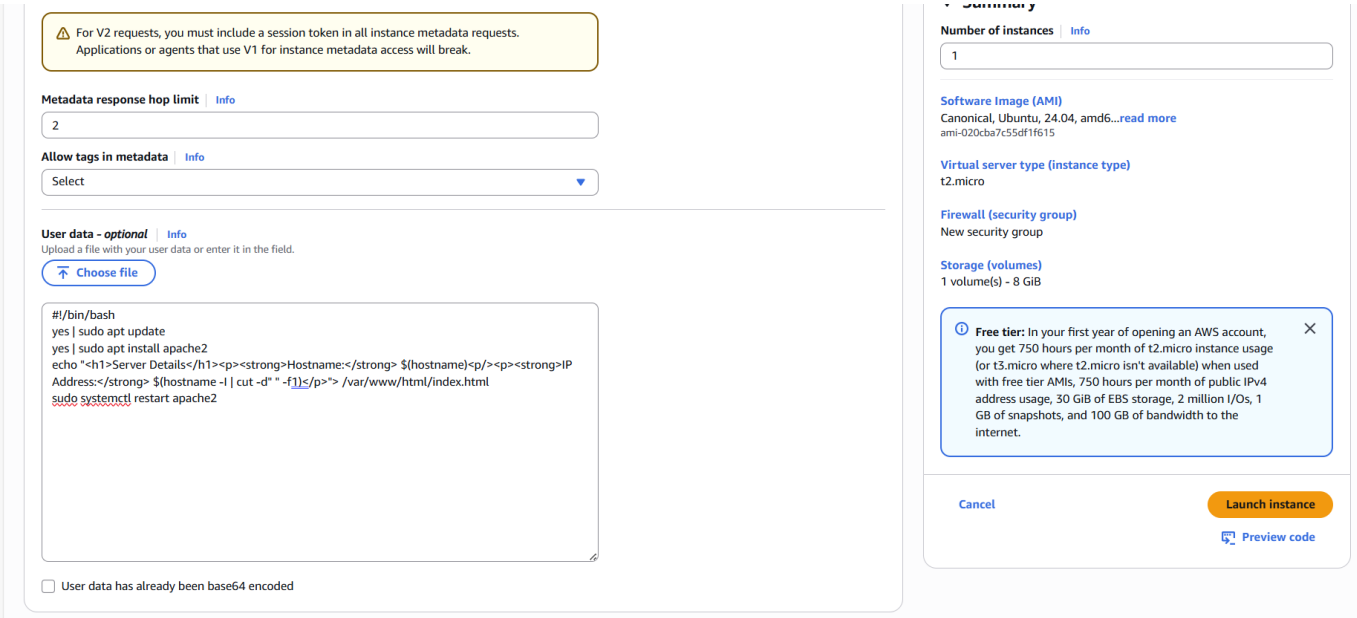


Scroll down to the end

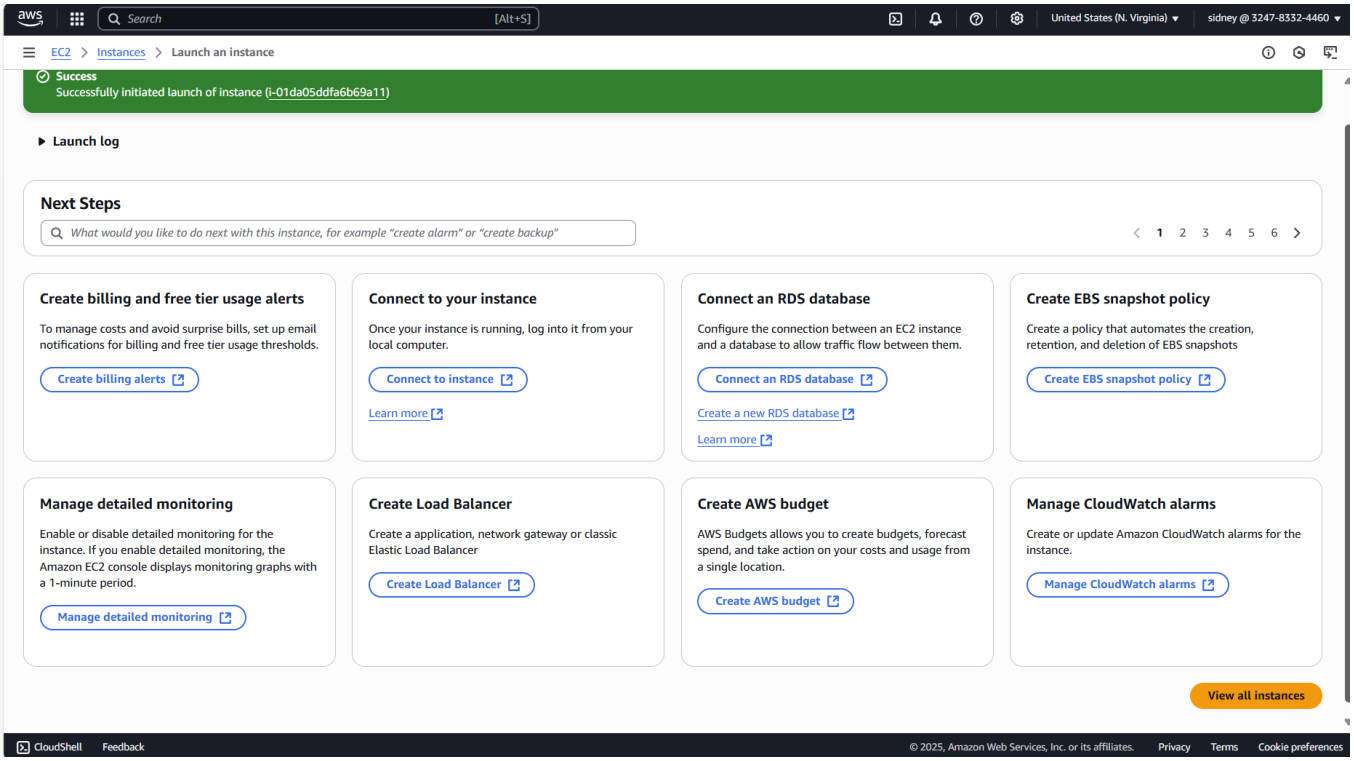


On “**user data**” paste the code below to install Apache

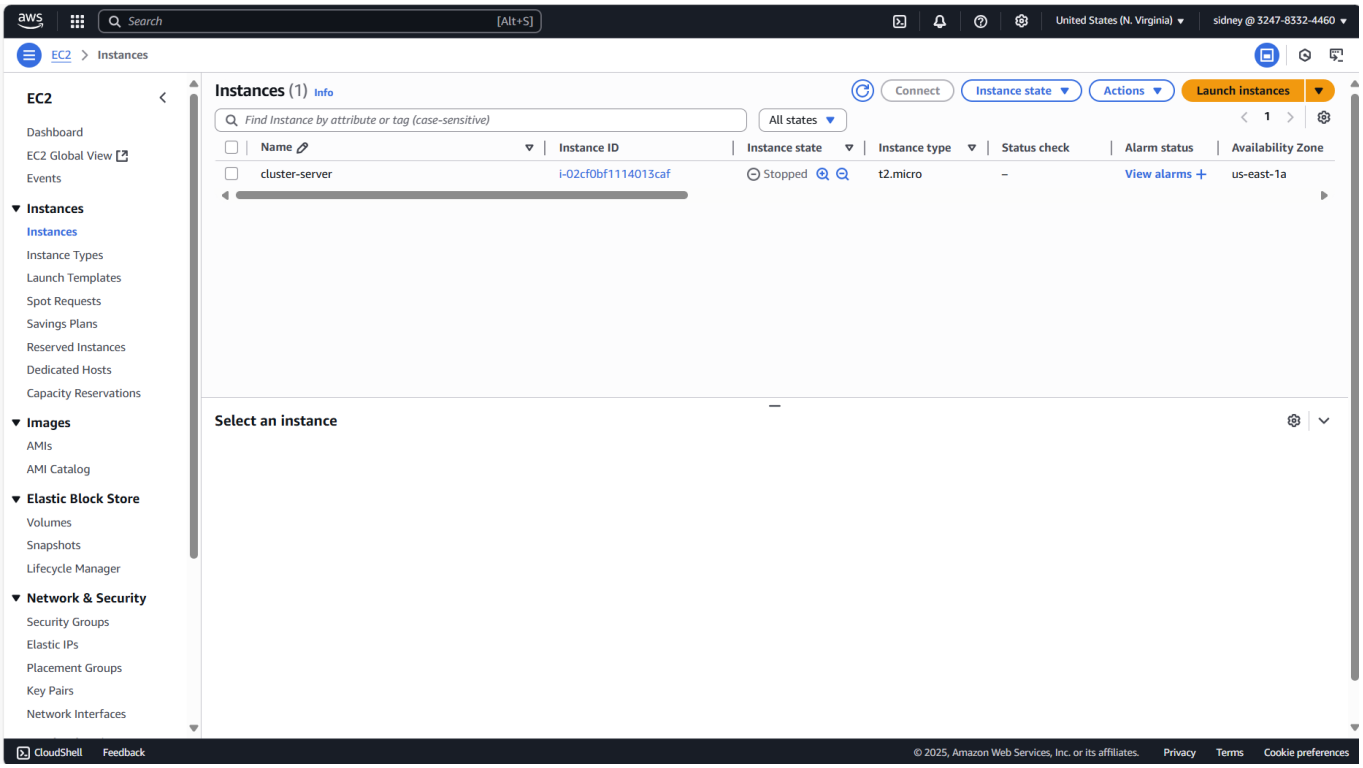
```
#!/bin/bash
yes | sudo apt update
yes | sudo apt install apache2
echo "<h1>Server Details</h1><p><strong>Hostname:</strong>
$(hostname)<p/><p><strong>IP Address:</strong> $(hostname -I | cut -d" " -
f1)</p>"> /var/www/html/index.html
sudo systemctl restart apache2
```



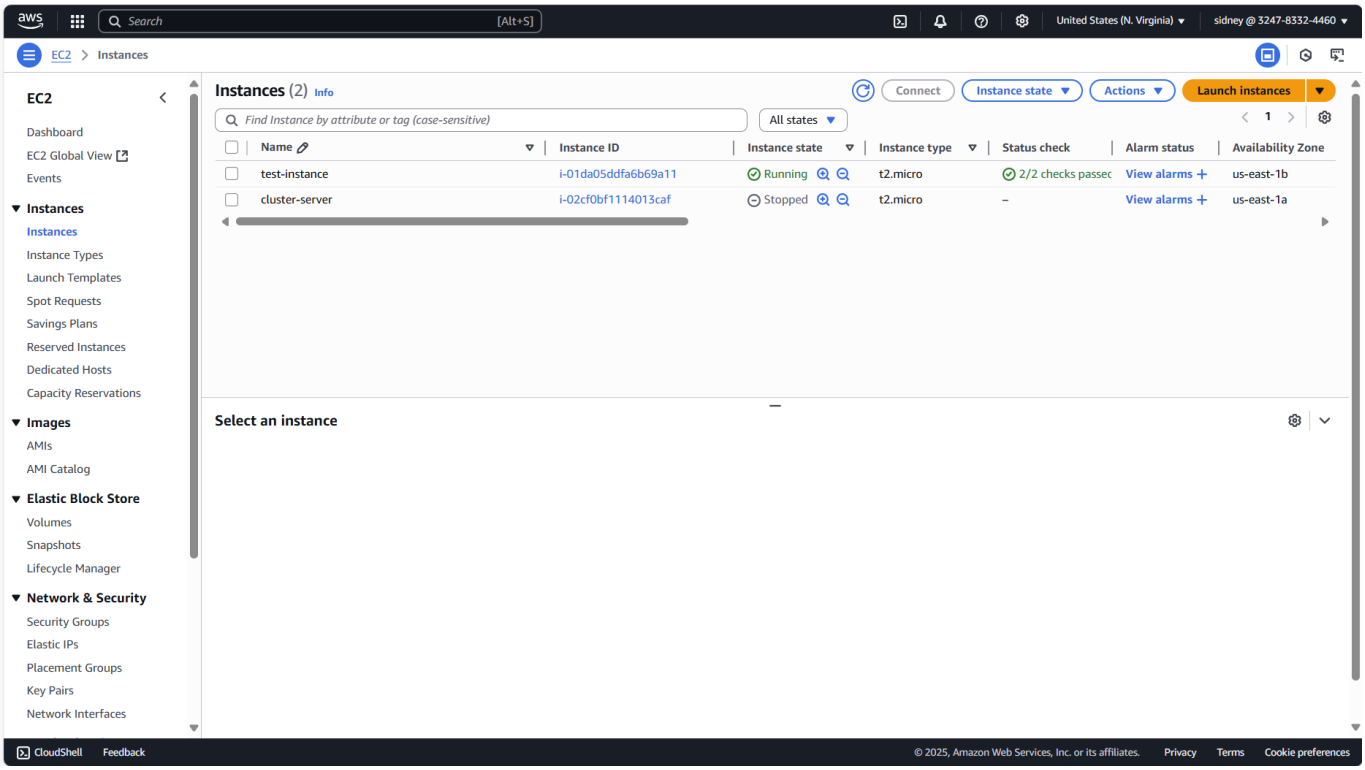
Click on “**Launch Instance**”



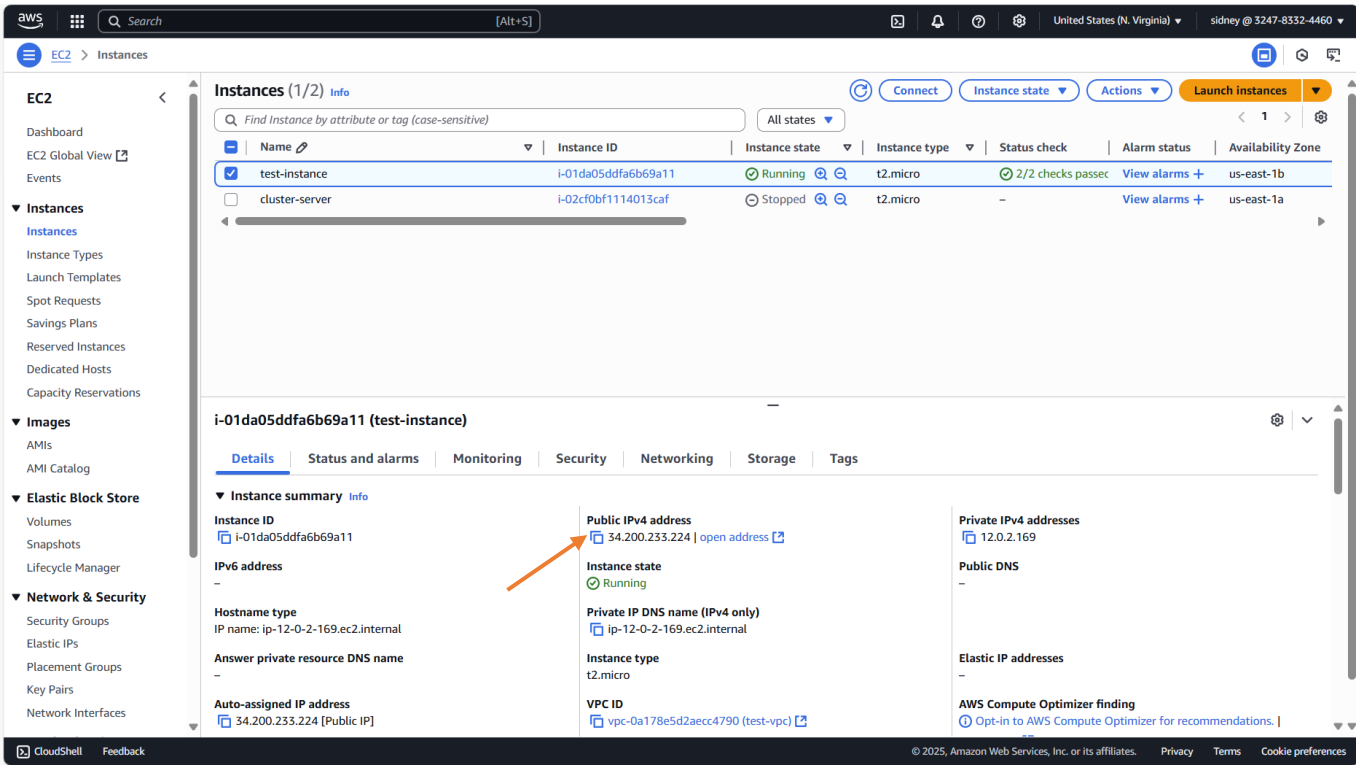
The instance is being created. Click on "**View All Instances**"



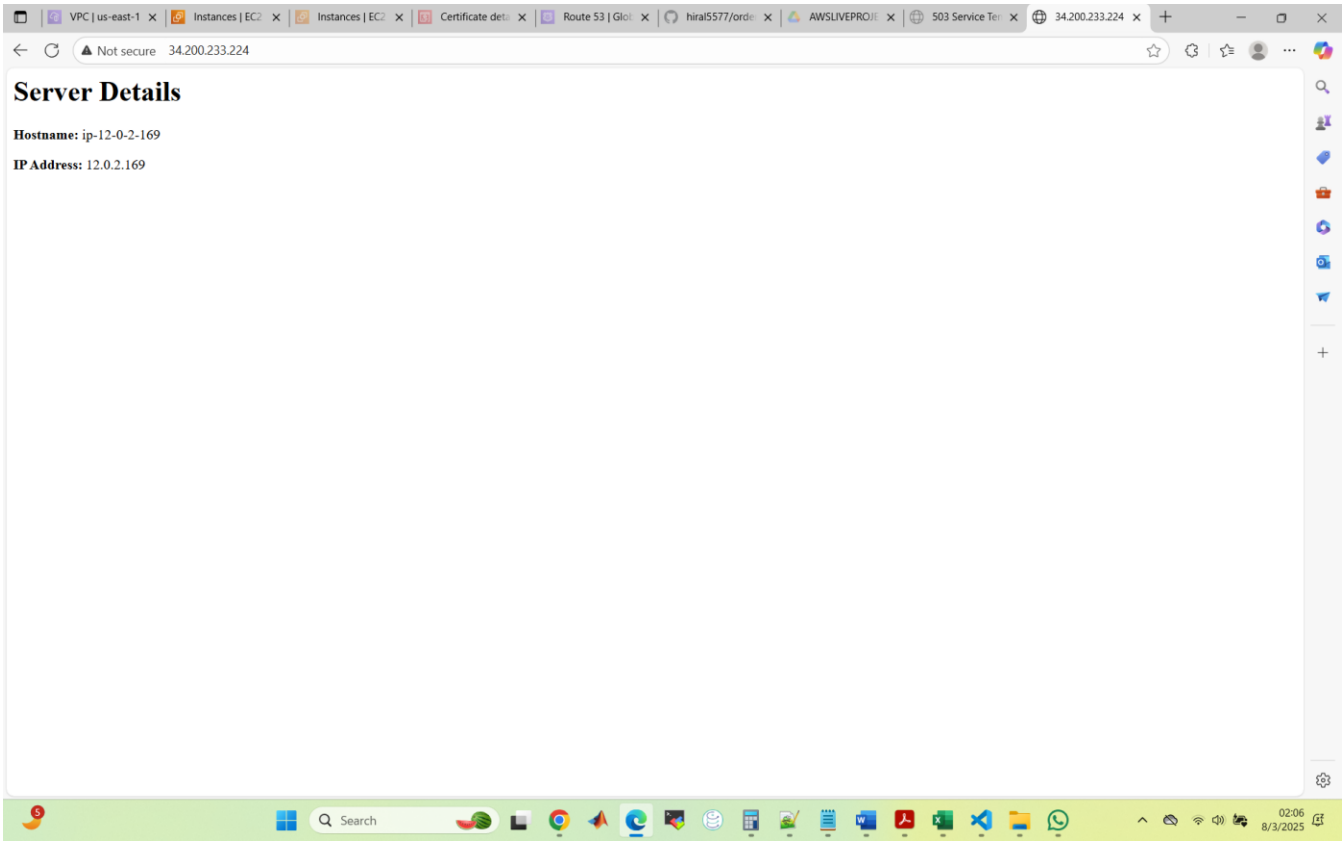
Wait for the instance to pass the "**2/2 check**"



The instance has passed the 2/2 check. Select the instance



Copy the "**Public IPv4 address**" and test on the browser.



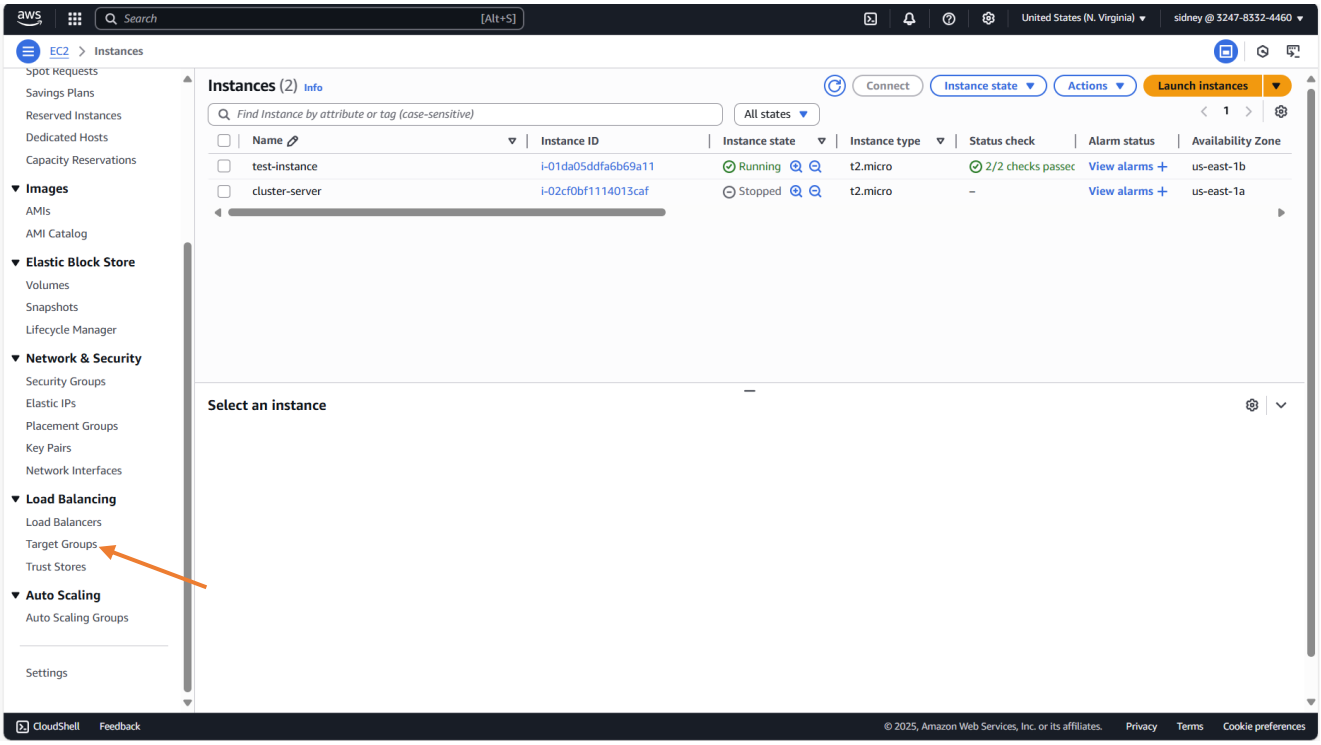
You can see that the supposed homepage of the EC2 instance is displayed. So far everything is working.

Remember we are not going to use the IP address. Our aim is to create a load balancer then point that domain to this particular IP address.

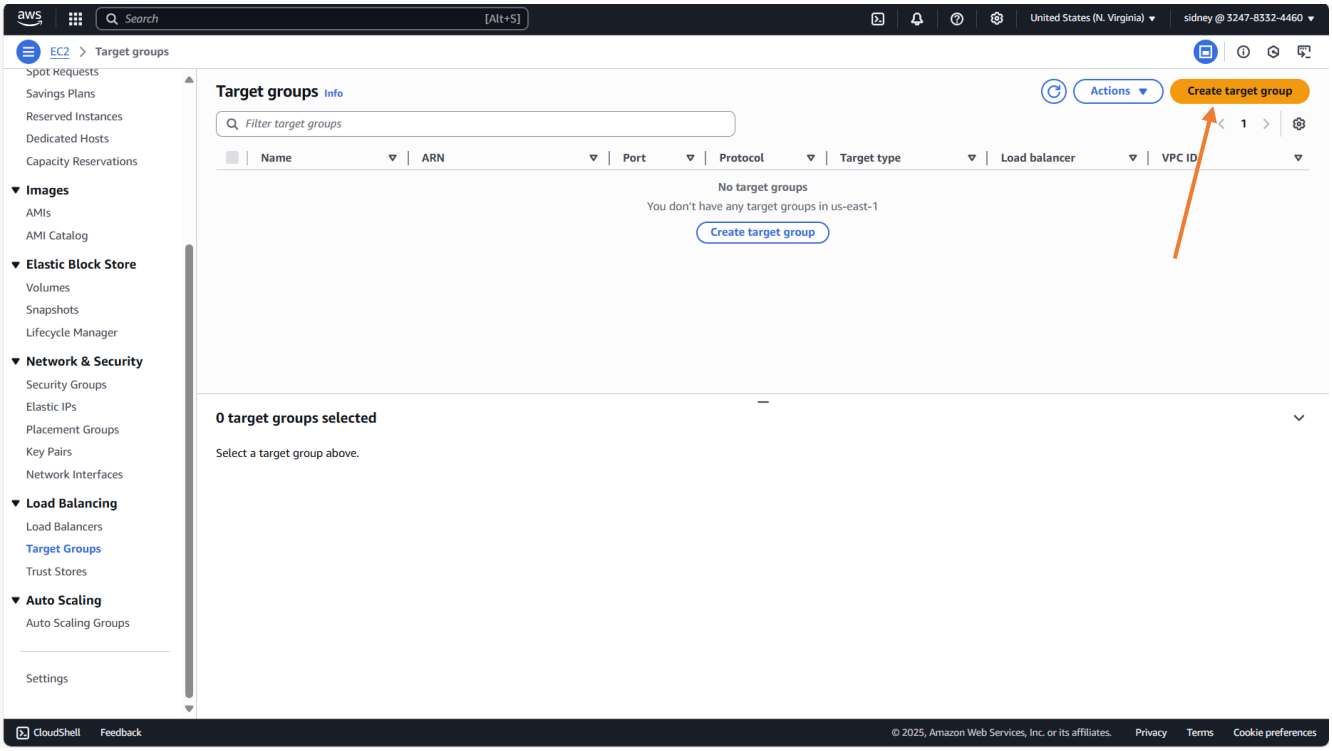
STEP 6: Set up Load Balancer

The next resource to create is the load balancer but before we create a load balancer, we need to create a "**Target Group**" for the load balancer.

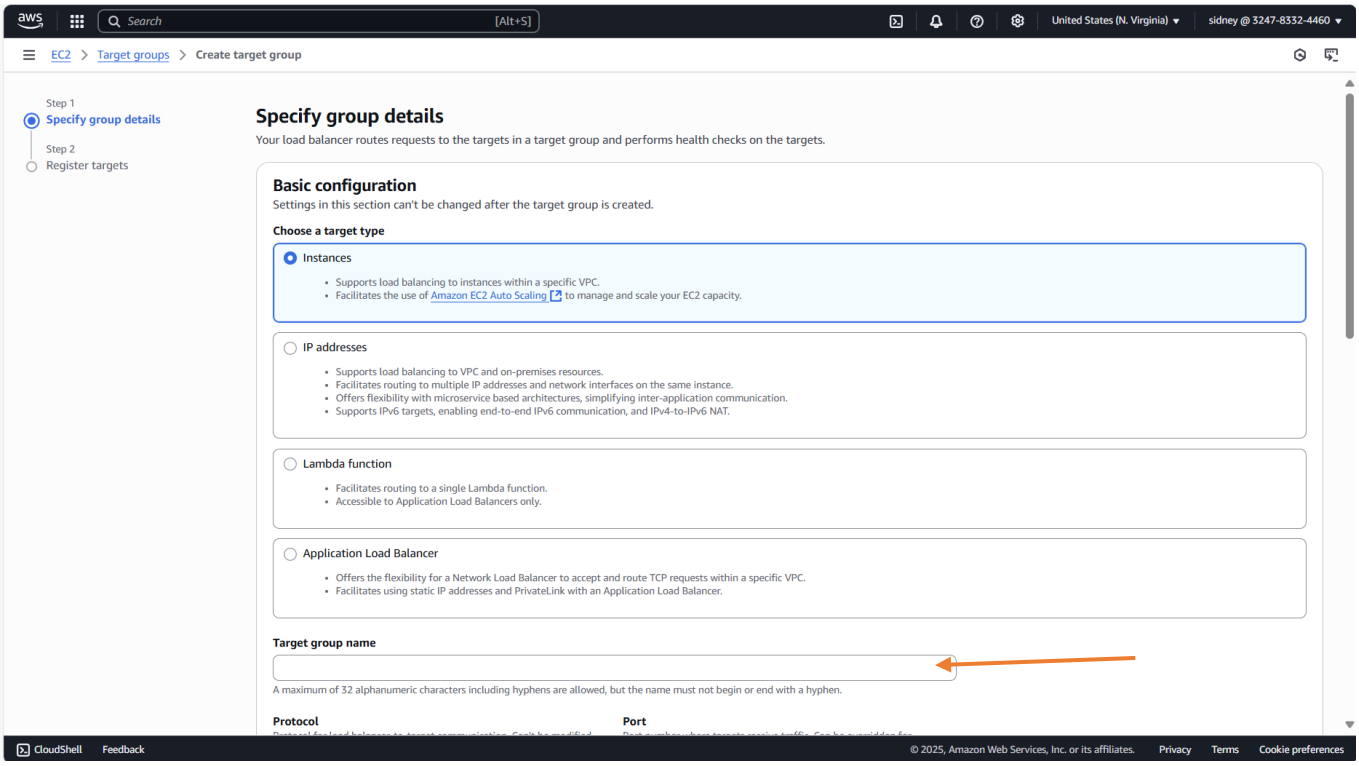
Go to the EC2 dashboard



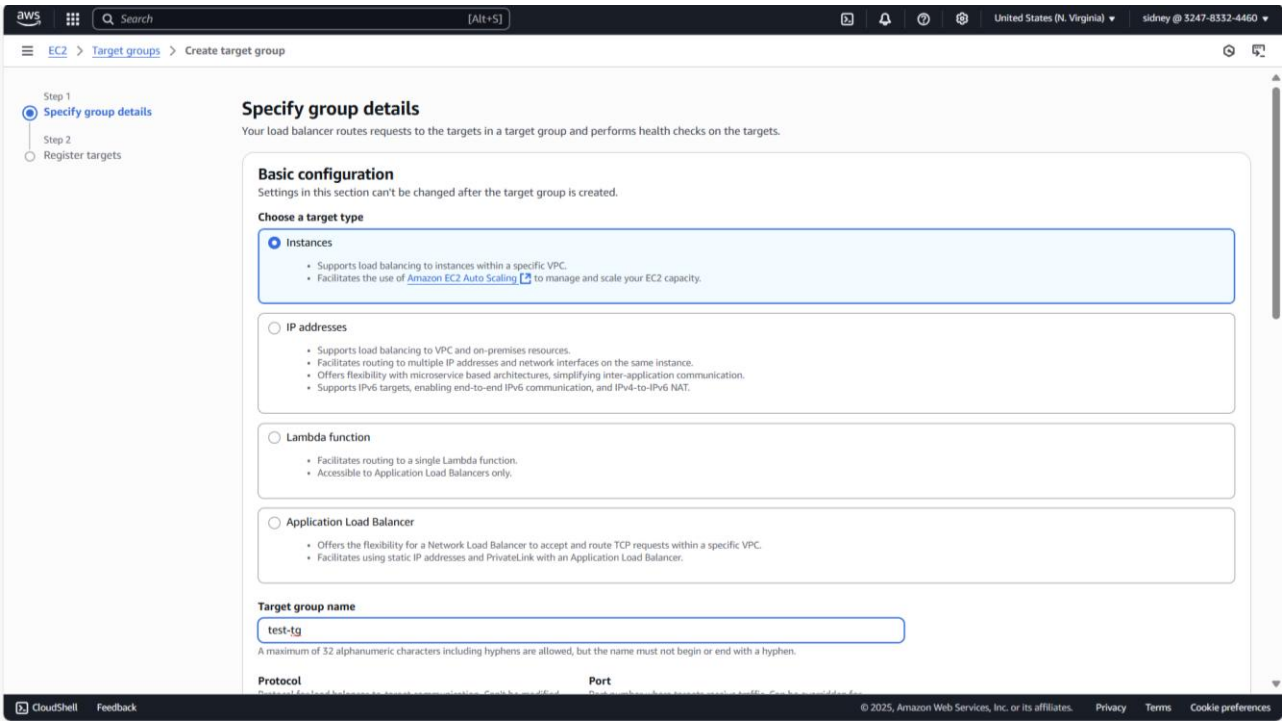
Click on "**Target Groups**".



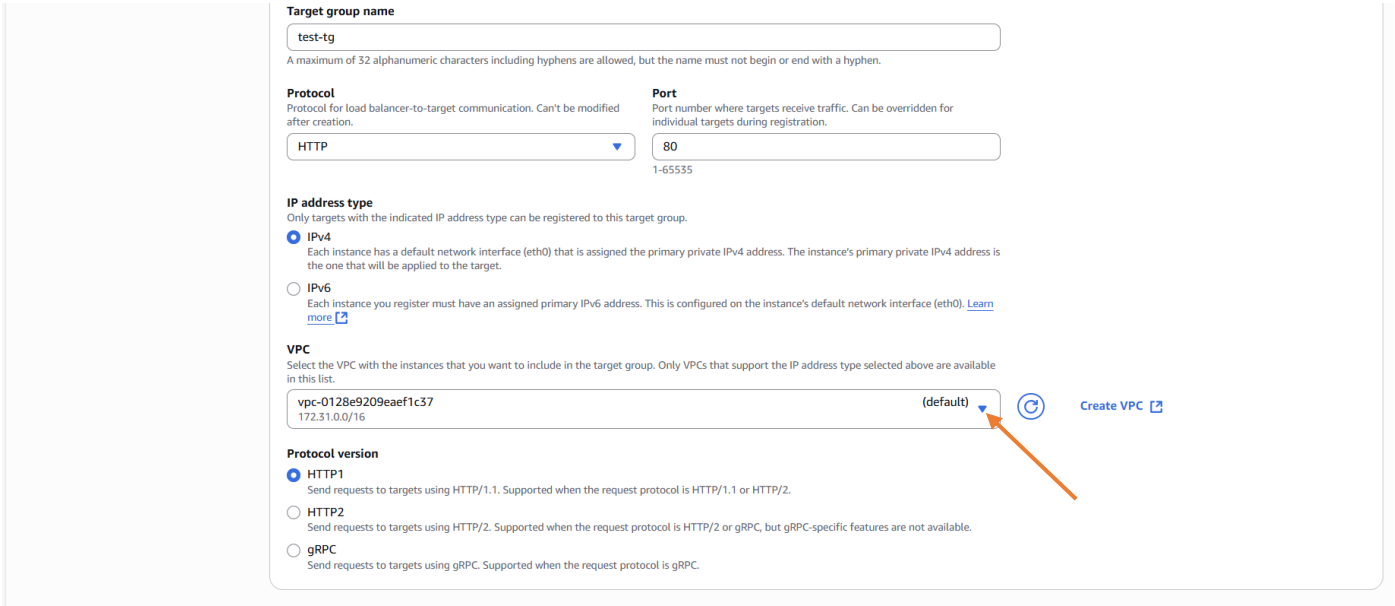
Click on “**Create Target Group**”. Target group is just a logical group where we will put our EC2 instance. For example, we have created one EC2 instance, so we can add that EC2 instance to one target group. One target group can contain multiple EC2 instances, so that our Load Balancer can point to the target group and the targe group can have a bunch of EC2 instances running. Load balancer can direct the traffic between those EC2 instances.



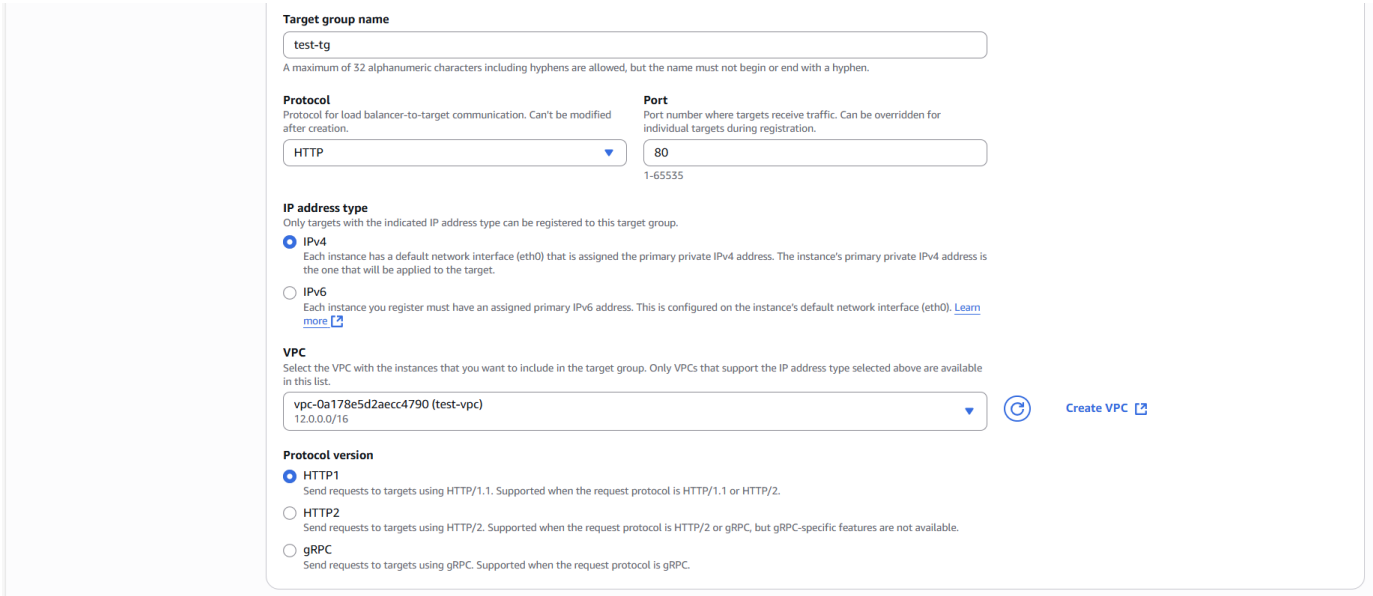
For the “**Target Type**”, we will use “**Instances**”. Then for “**Target Group Name**”, we will use “**test-tg**”



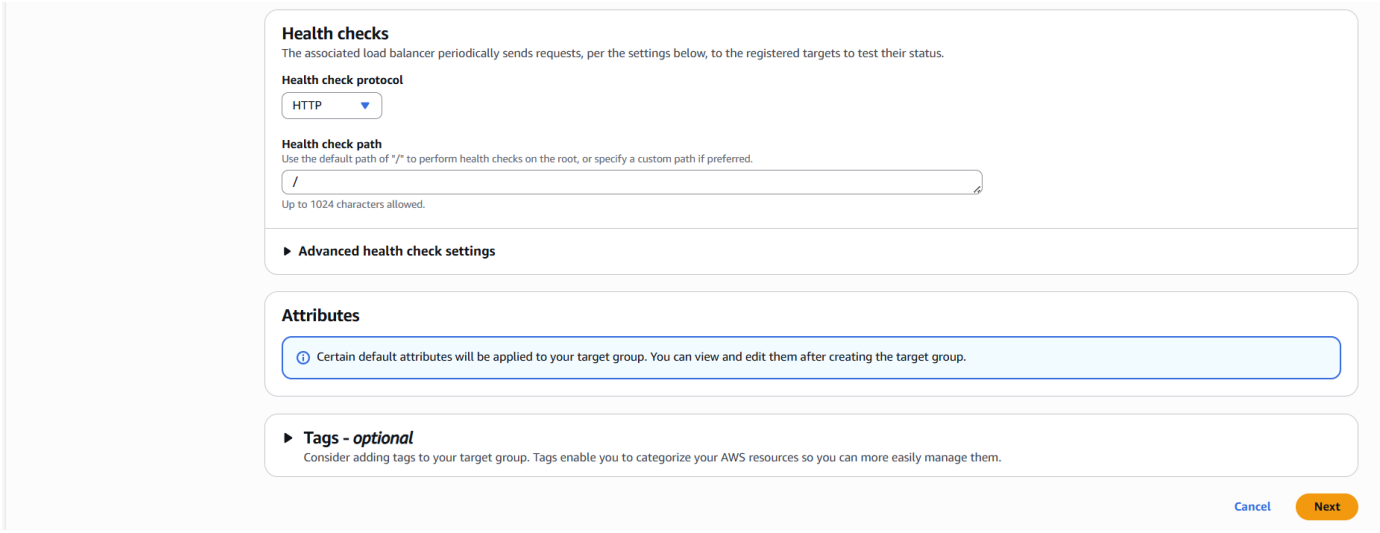
Scroll down to “**VPC**”



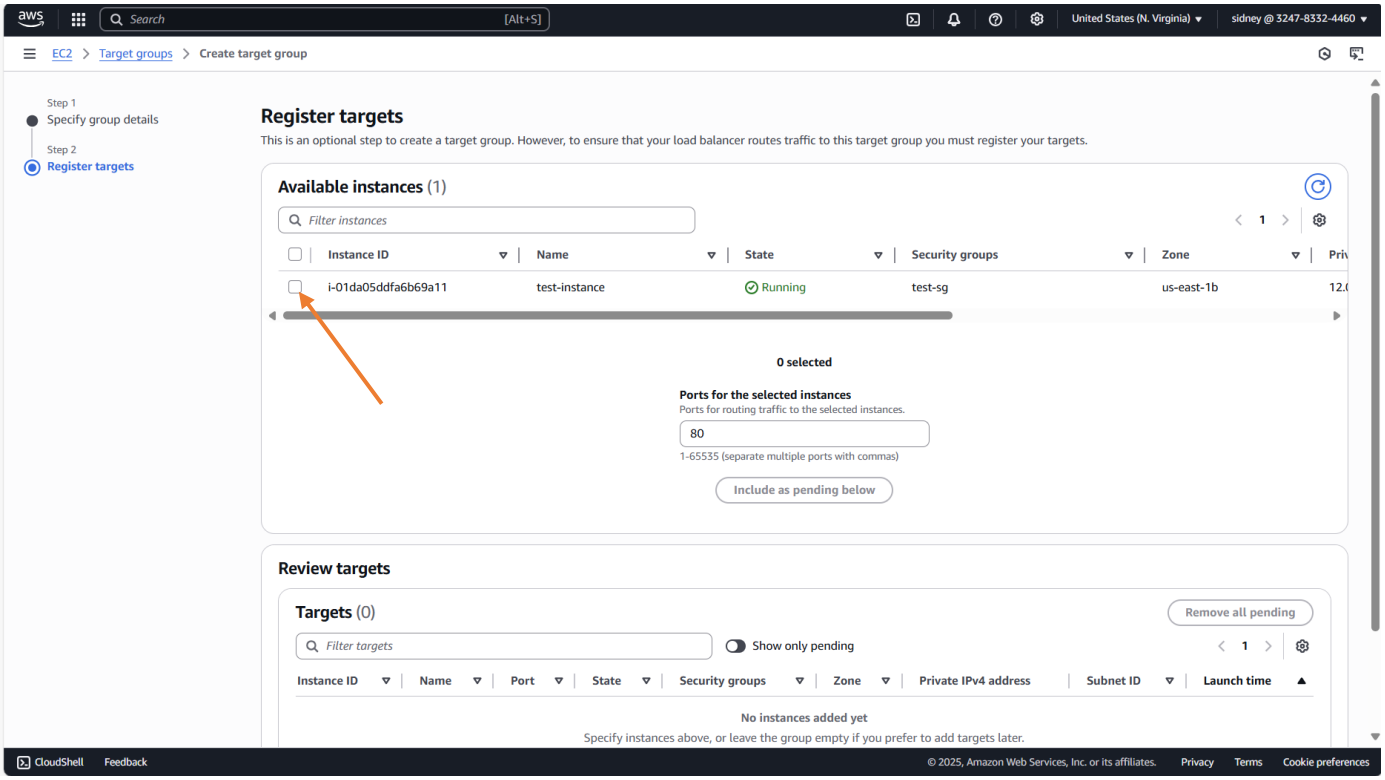
Click on the drop down on VPC and select our VPC that is “**test-vpc**”



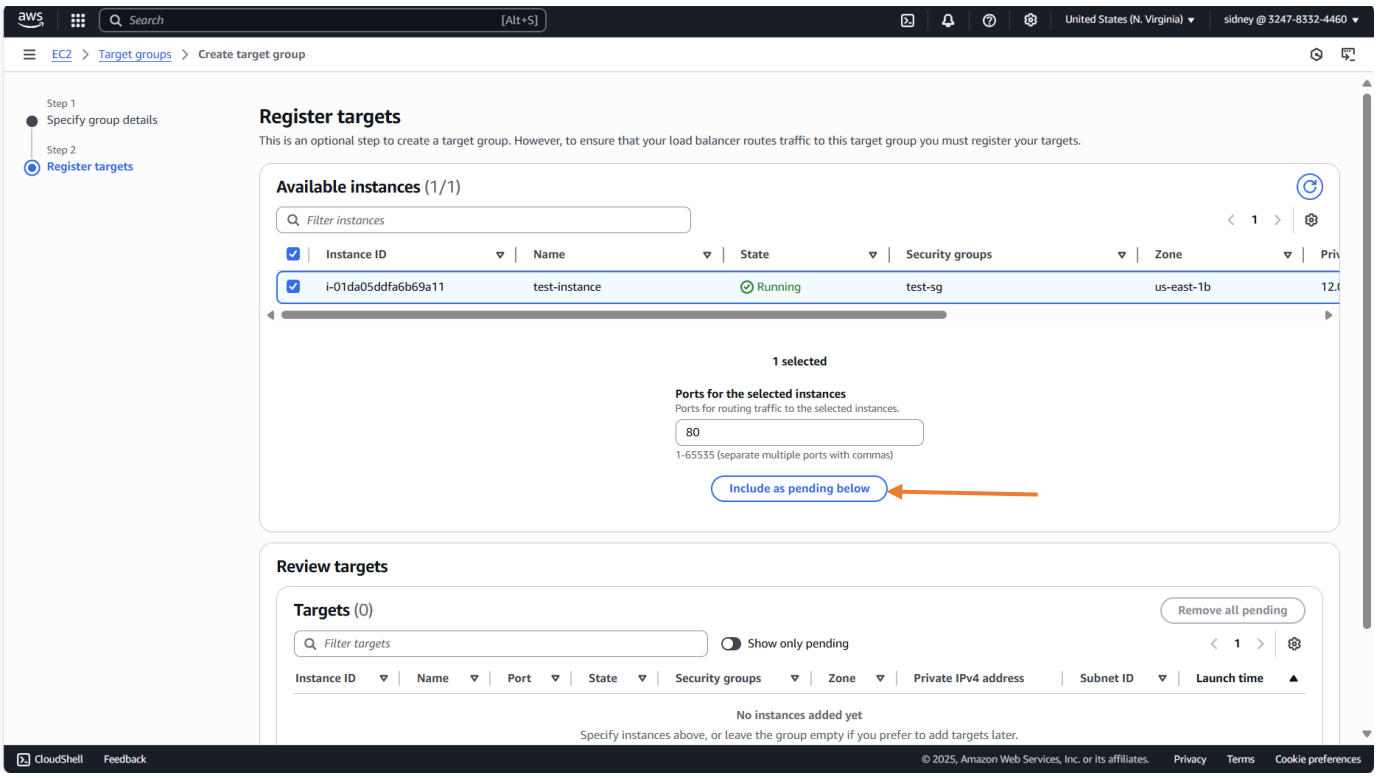
Then scroll down to the end



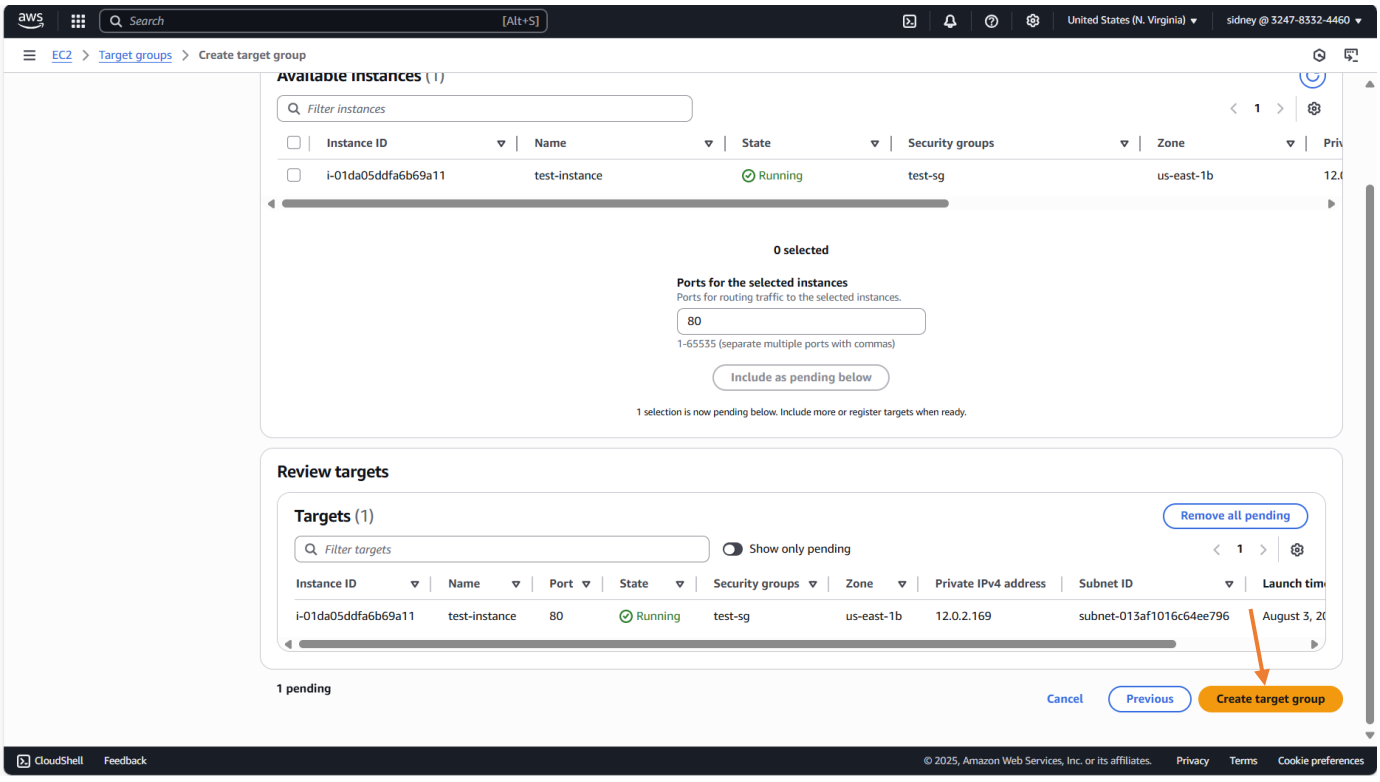
Click on "**Next**"



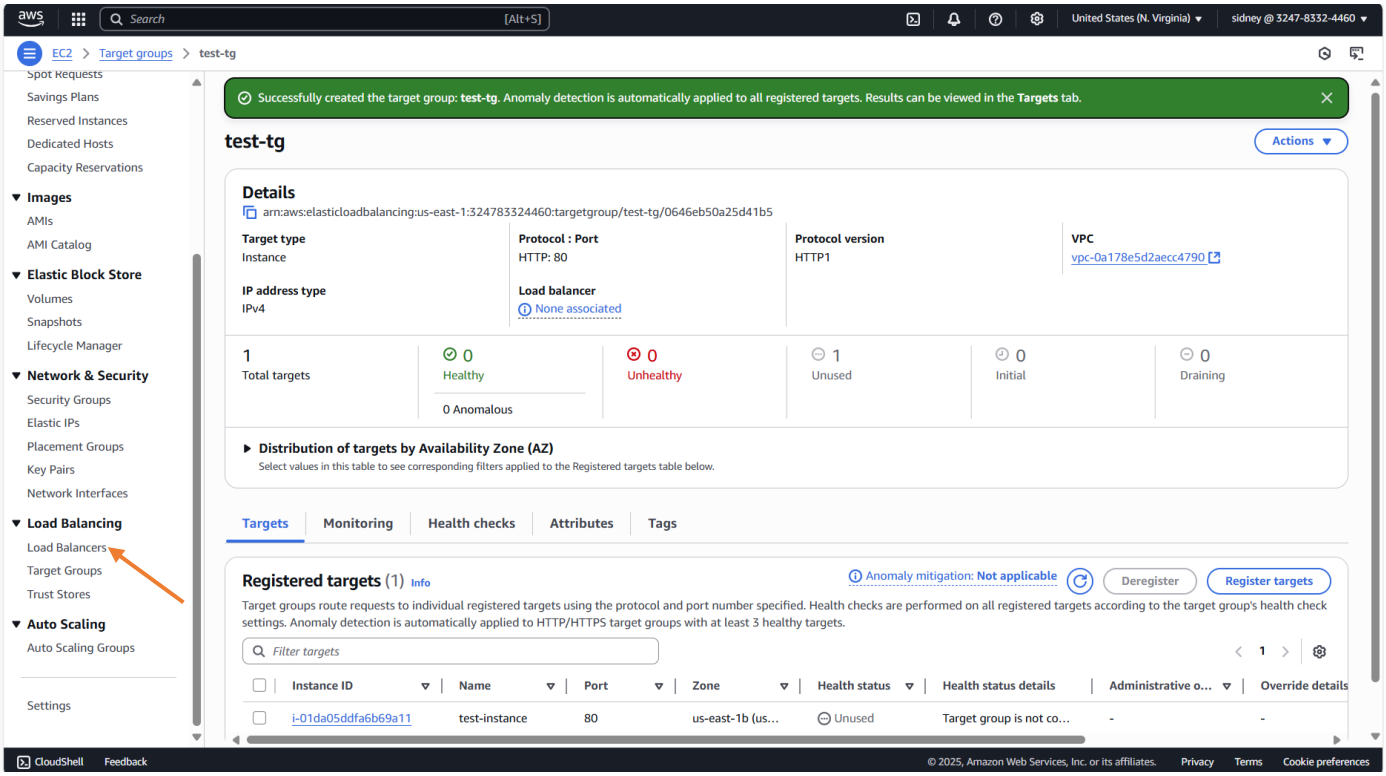
Now, we have to add our instance to the target group. Select the instance



Click on “**Include as pending below**”

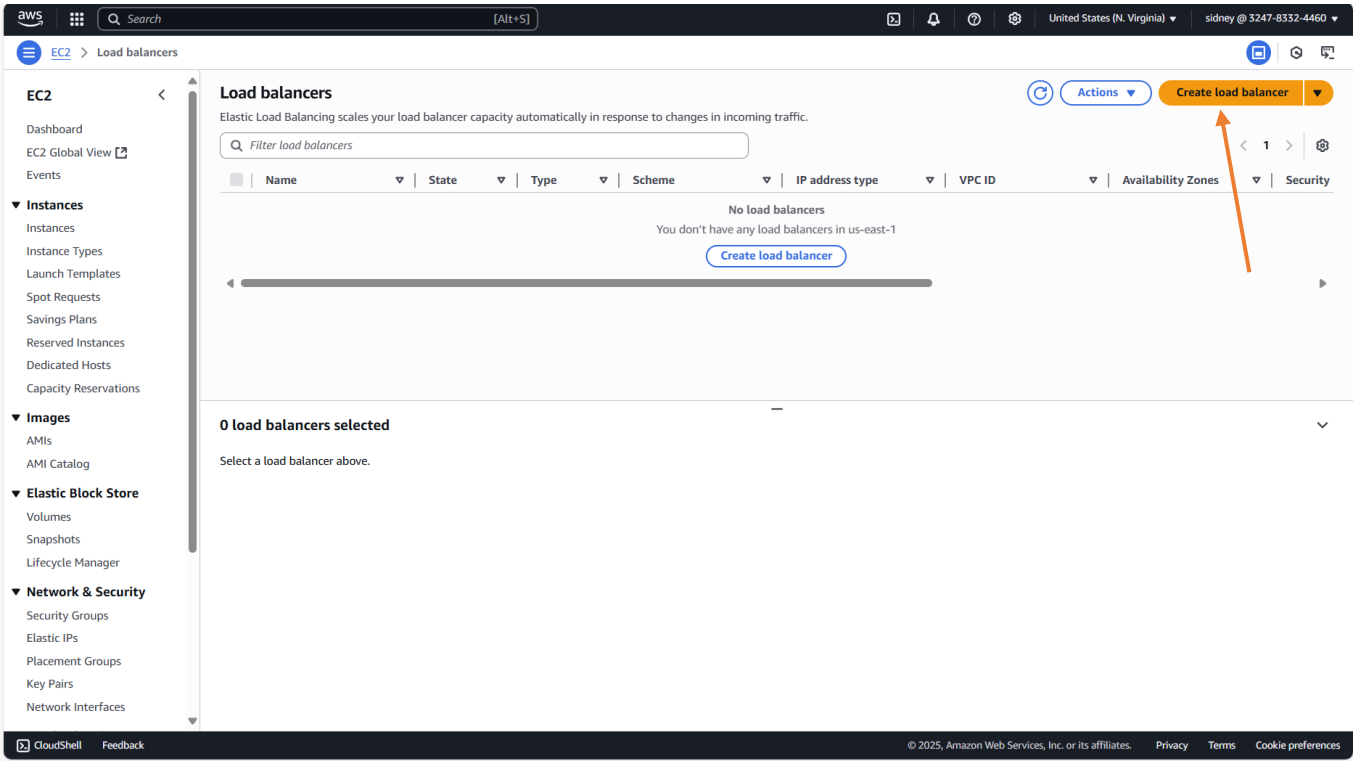


Then click on “**create target group**”

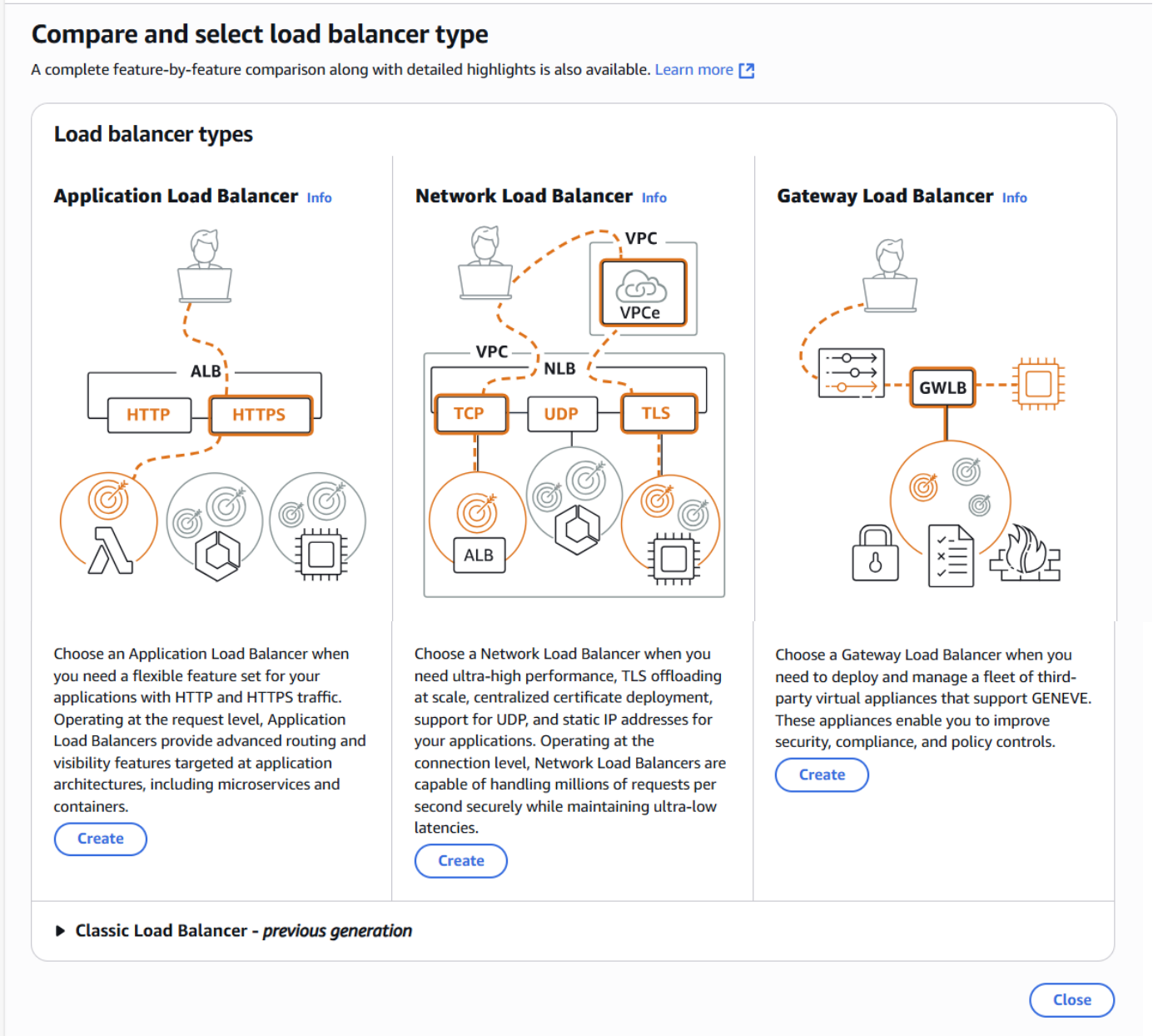


The target group has been created and our instance has been added to this target group.

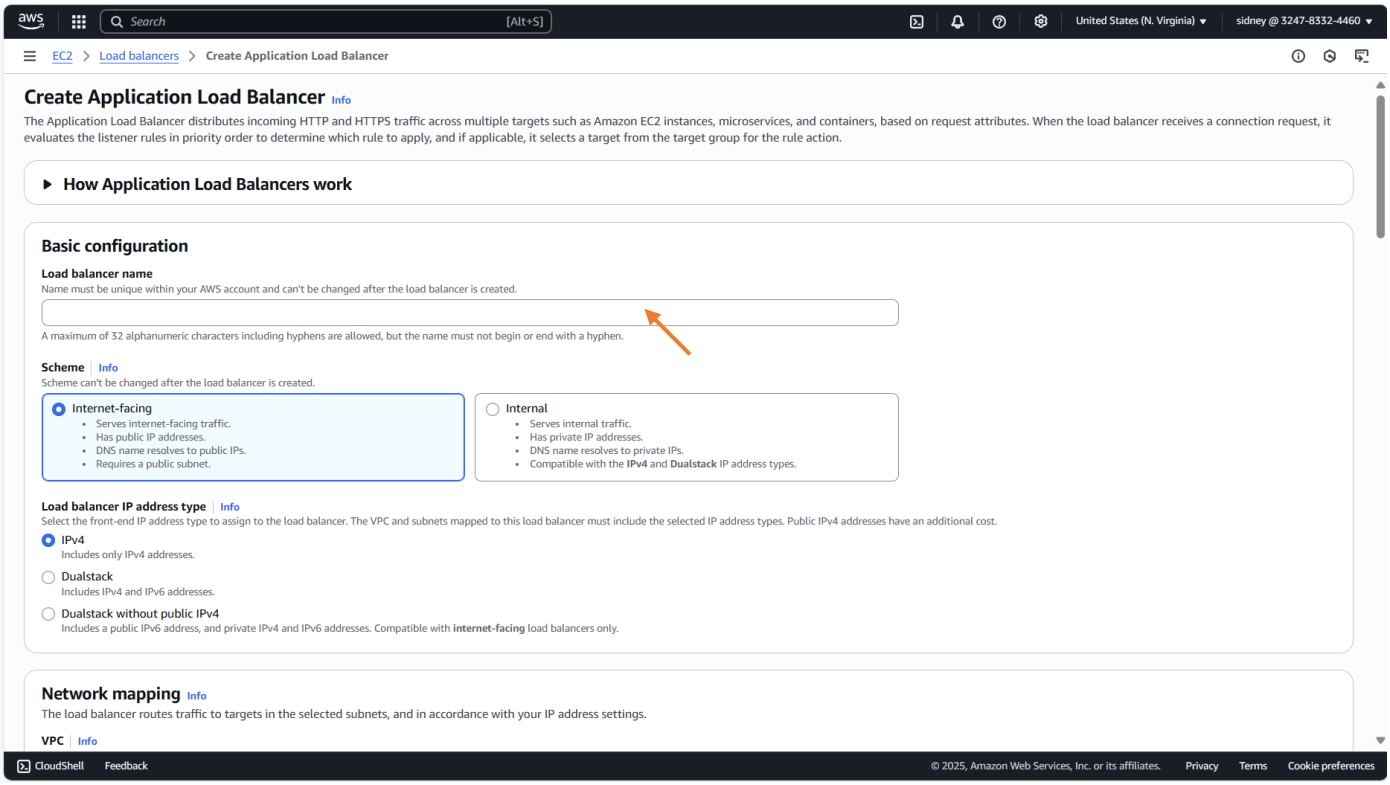
Now, we will create our load balancer. Click on "**Load Balancers**"



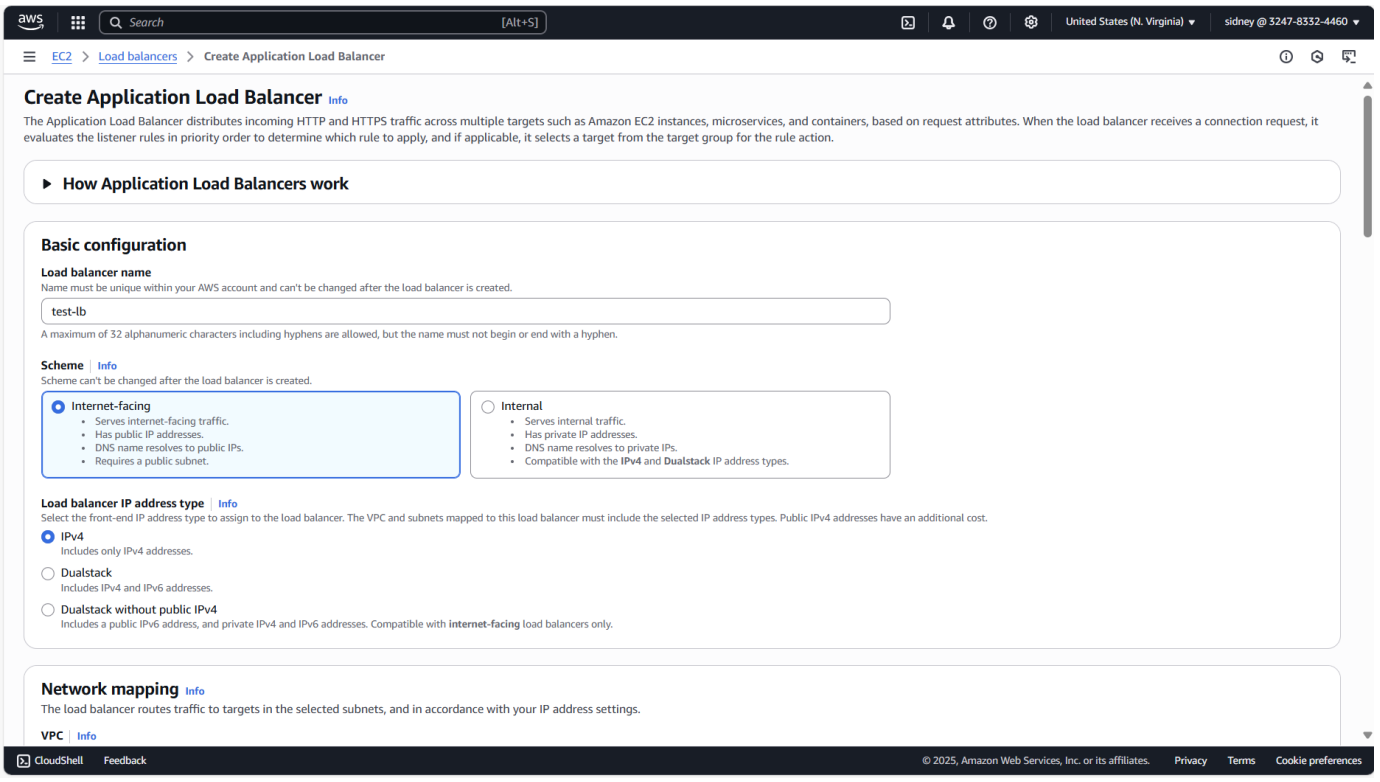
Click on "**create load balancer**"



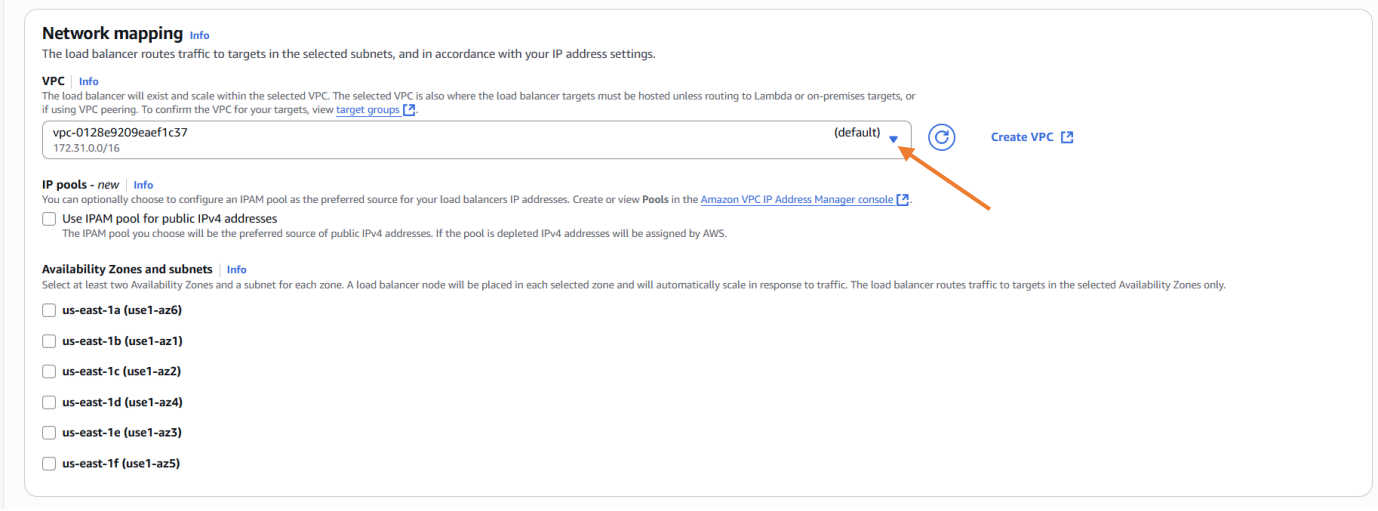
We are going to use application load balancer, so click on "**create**" under "**Application Load Balancer**"



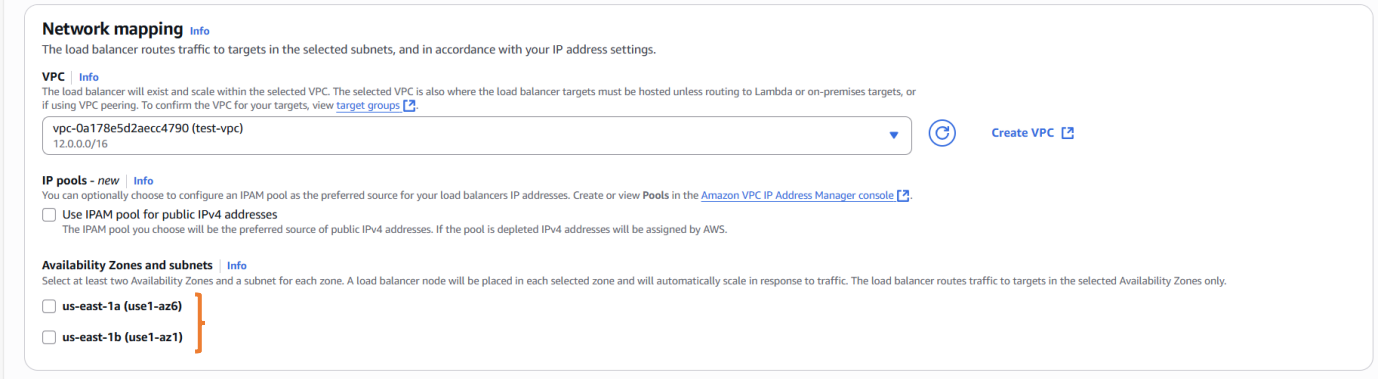
We have to give the Load Balancer a name, I will call it "**test-lb**"



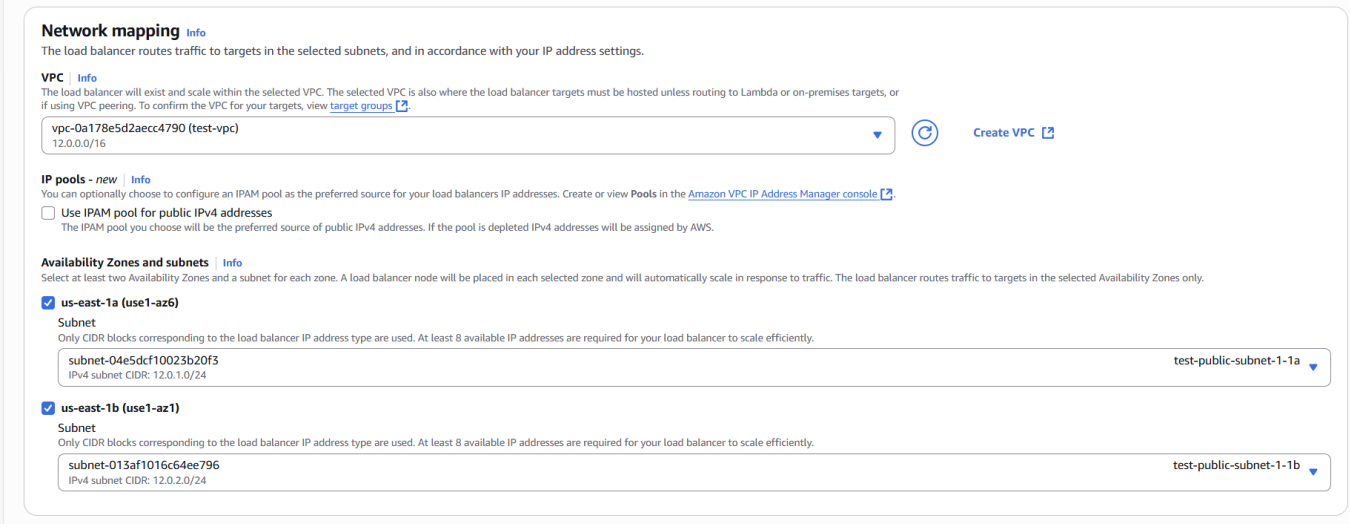
The scroll down to "**Network Mapping**"



Click on the drop down on "**VPC**" and select our VPC that is "**test-vpc**"

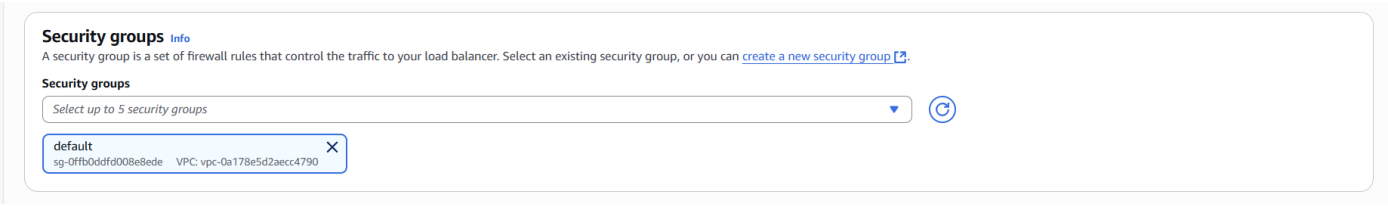


On "**Availability Zones and subnets**", select the two availability zones and their subnets

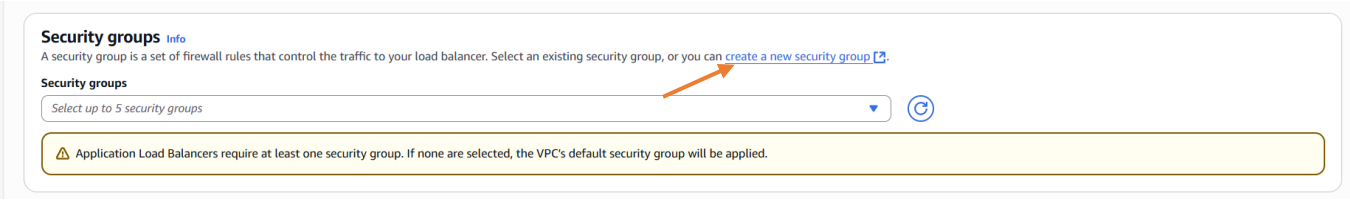


In the load balancer we also need to enable the same port 22 for SSH so that we will be able to connect to our EC2 instance. We need to also enable port 80 for http and also enable **port 443**. For now, we will enable the **security group for Port 22 and 80**.

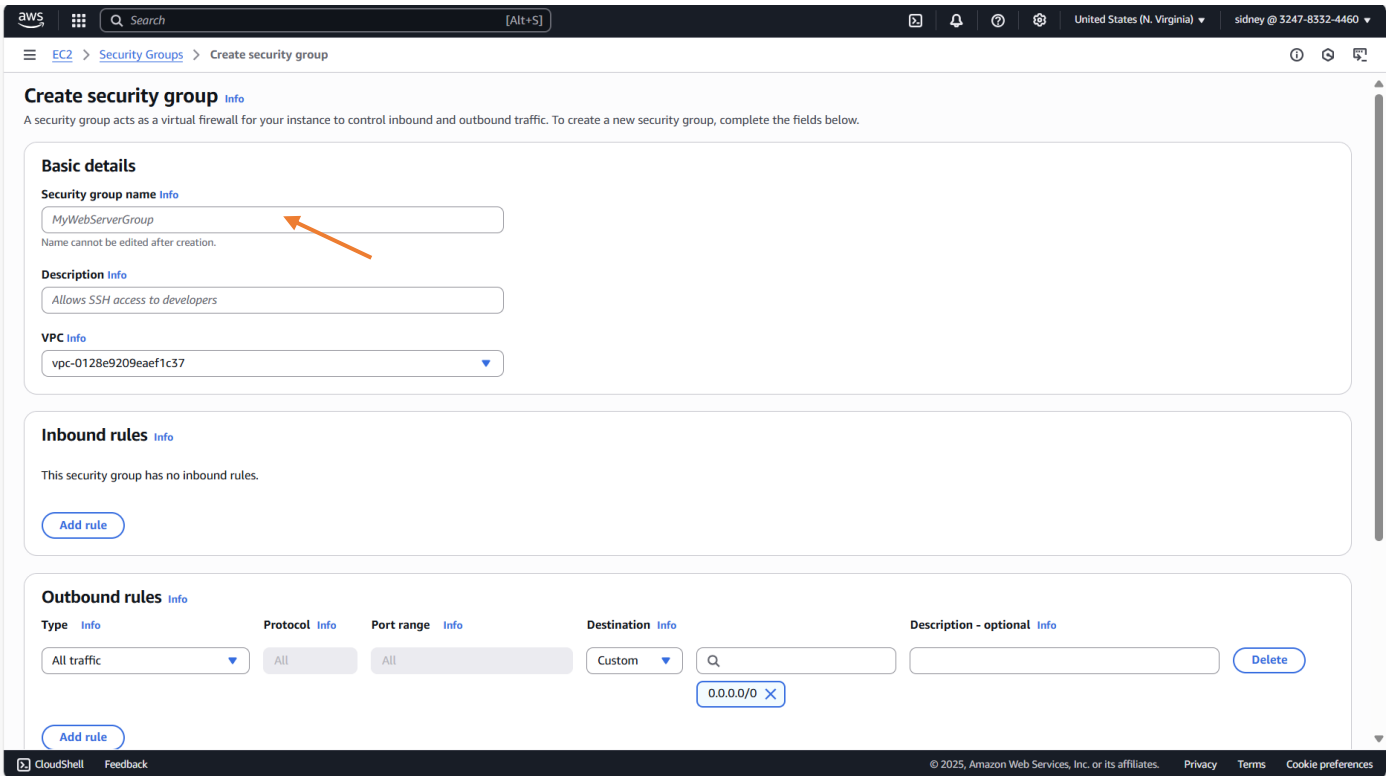
Scroll down to "**Security Group**"



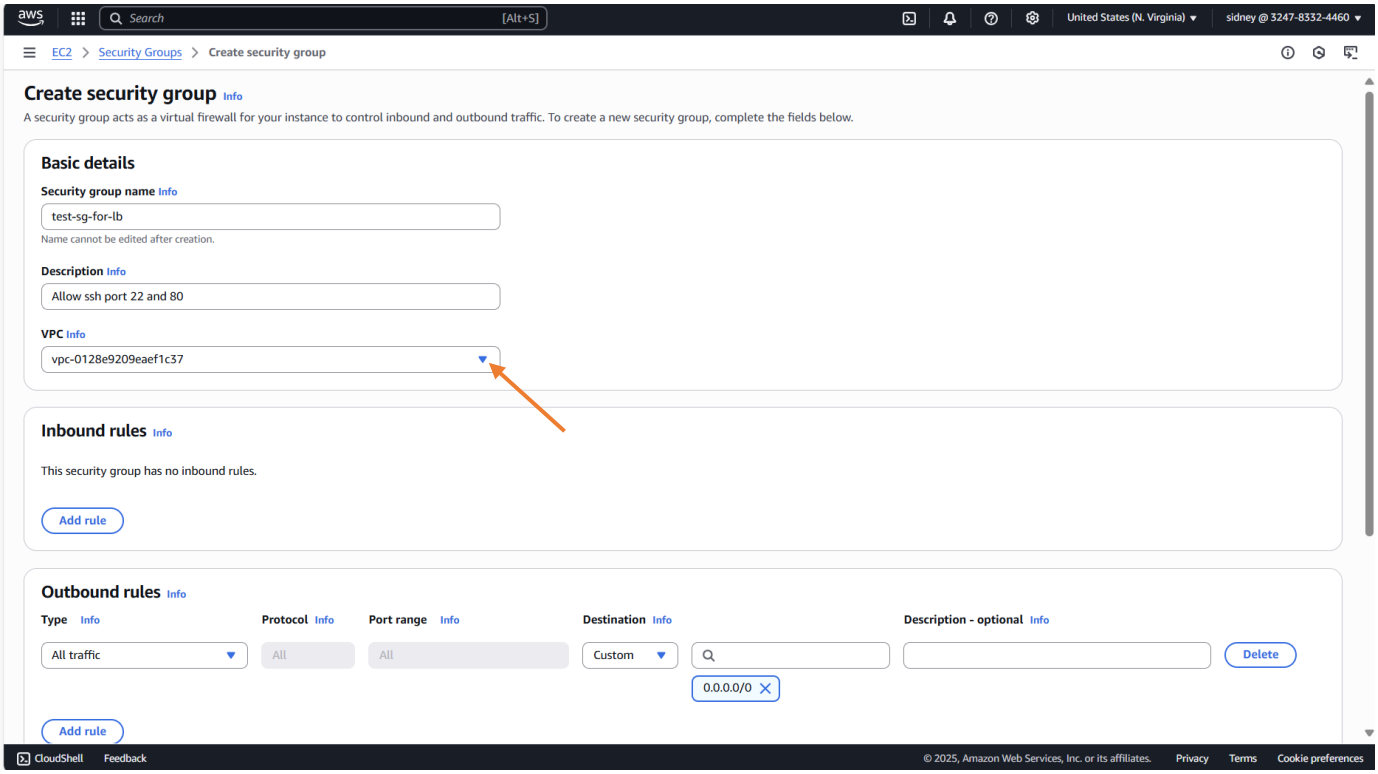
Remove the “**default**” security group



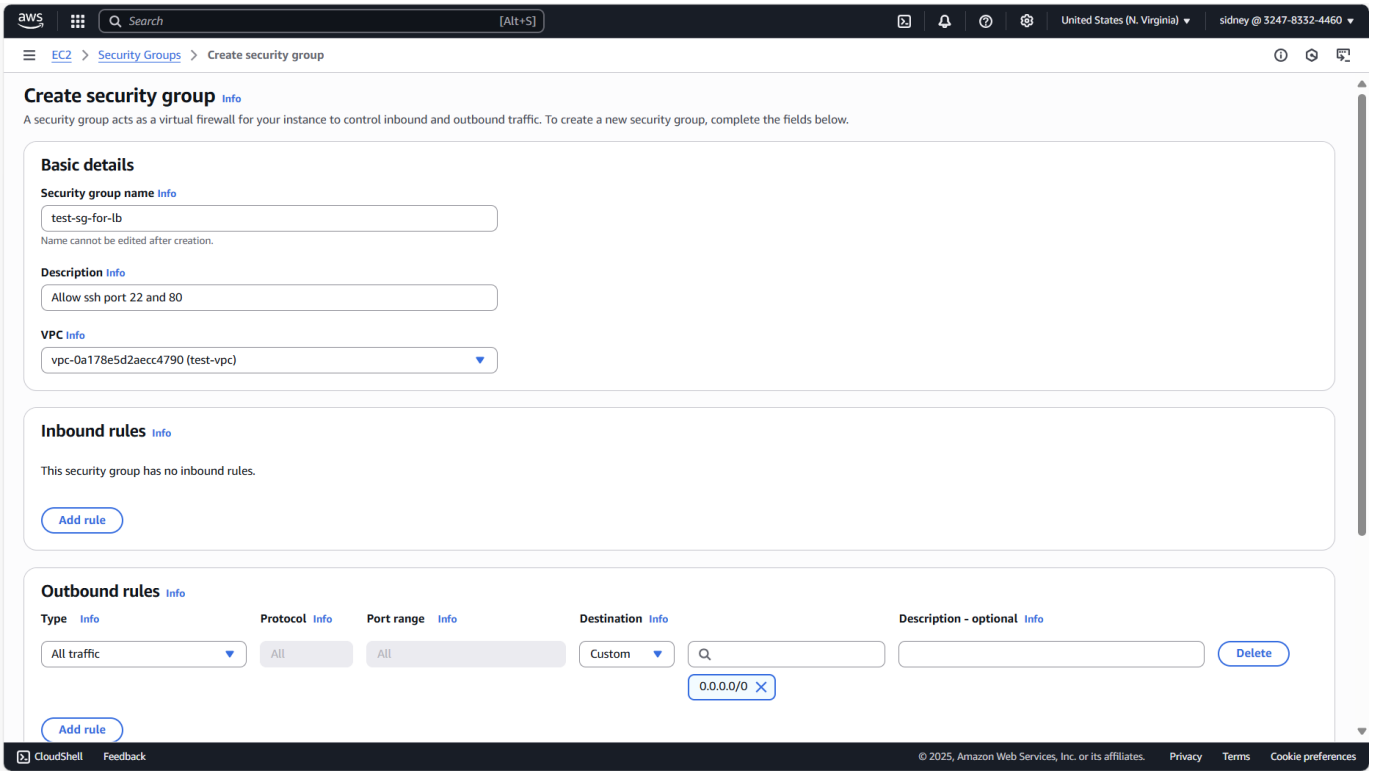
Click on “**create a new security group**”



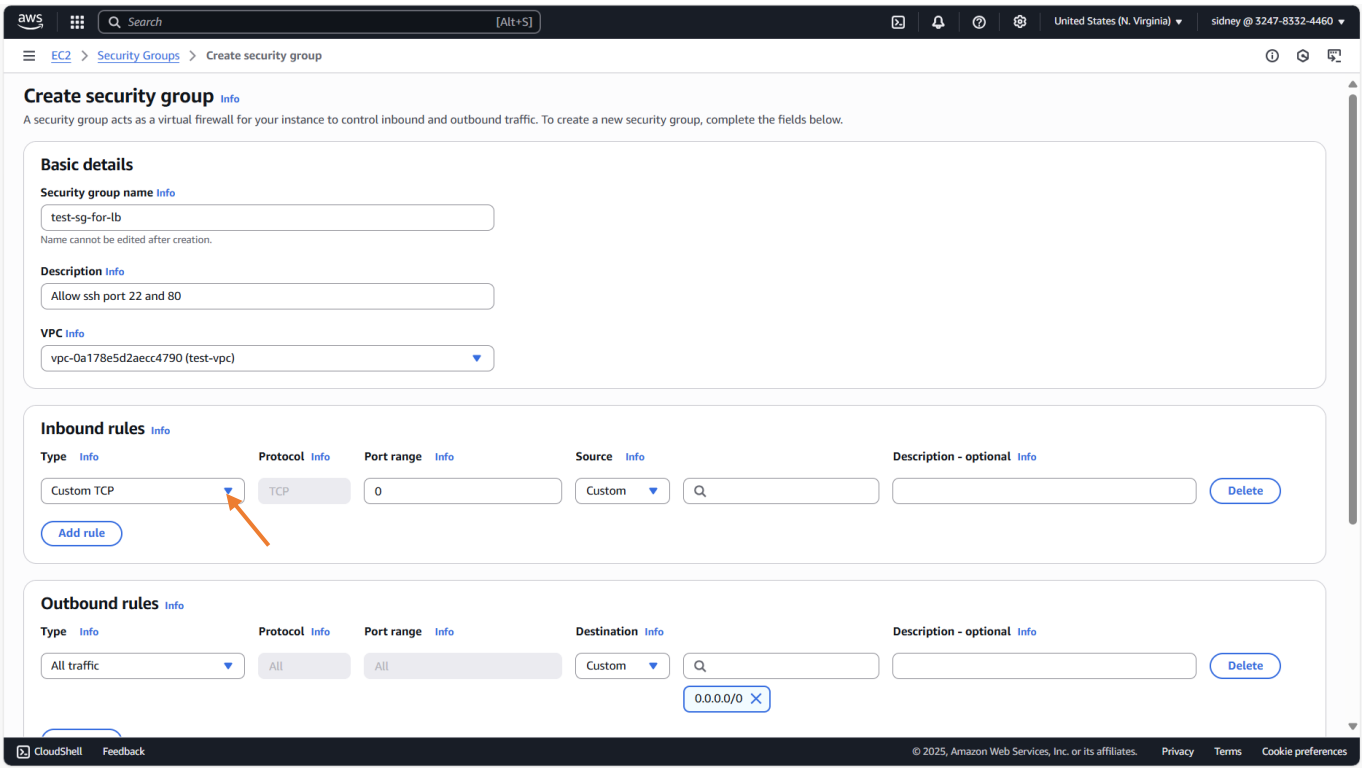
We will call the security group “**test-sg-for-lb**”. For description, we will type “**Allow ssh port 22 and 80**”



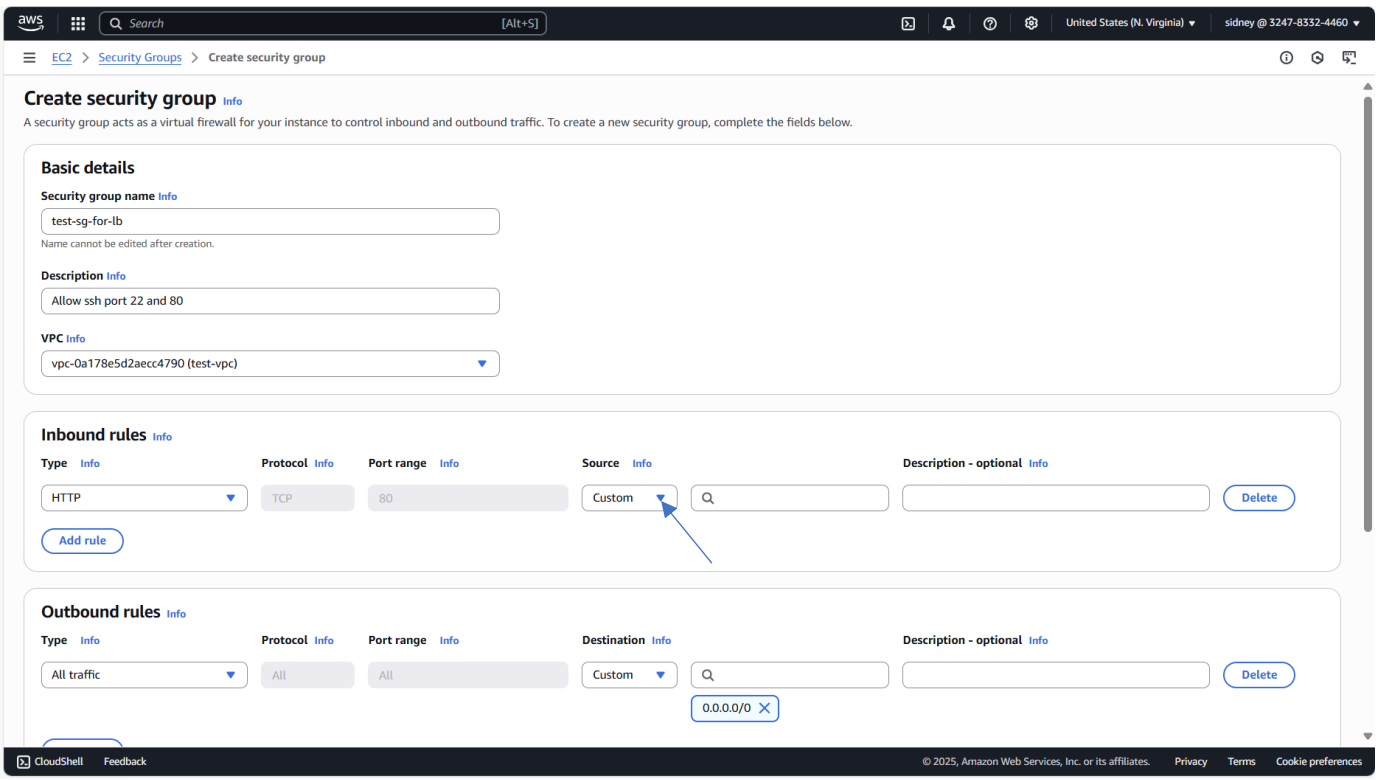
Then for "**VPC**", click on the drop down and select our VPC, that is "**test-vpc**"



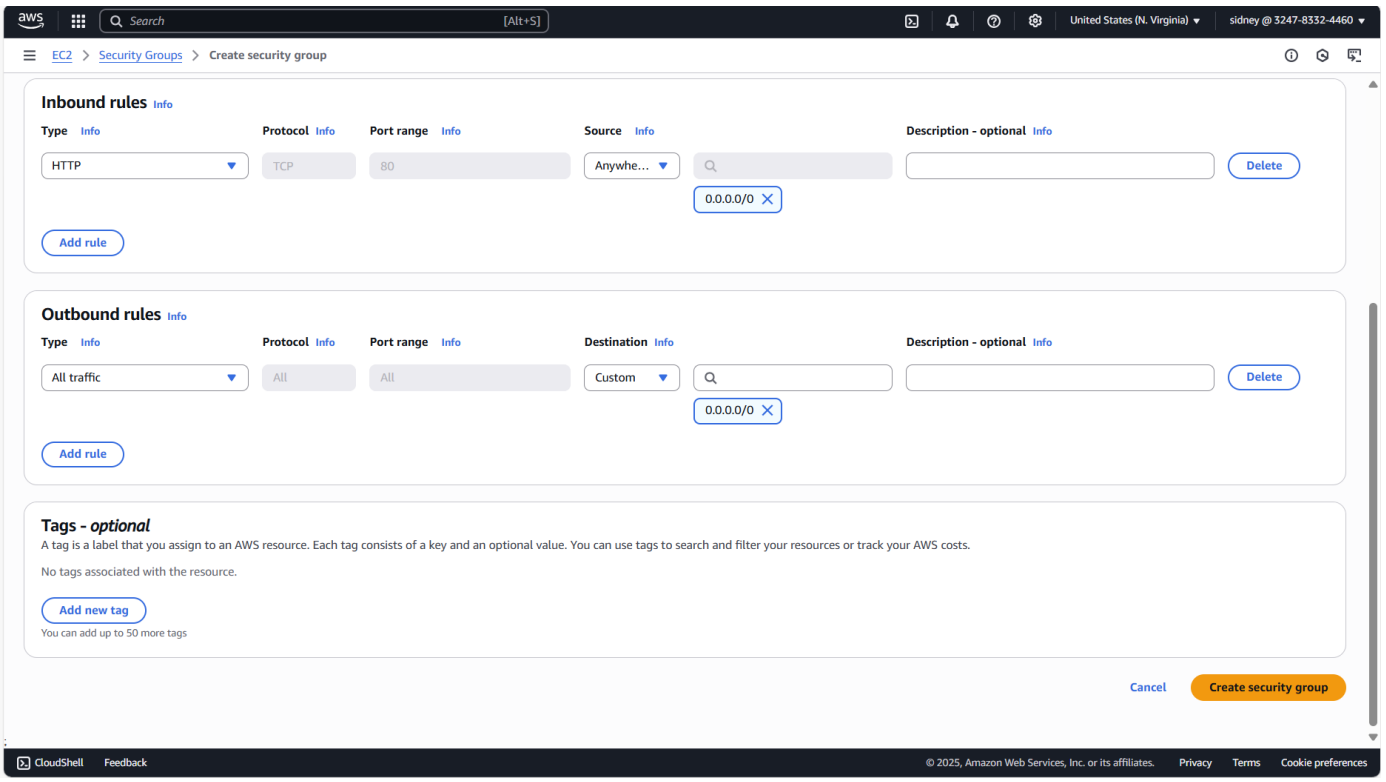
Then, under "**Inbound Rules**", click on "**Add rule**"



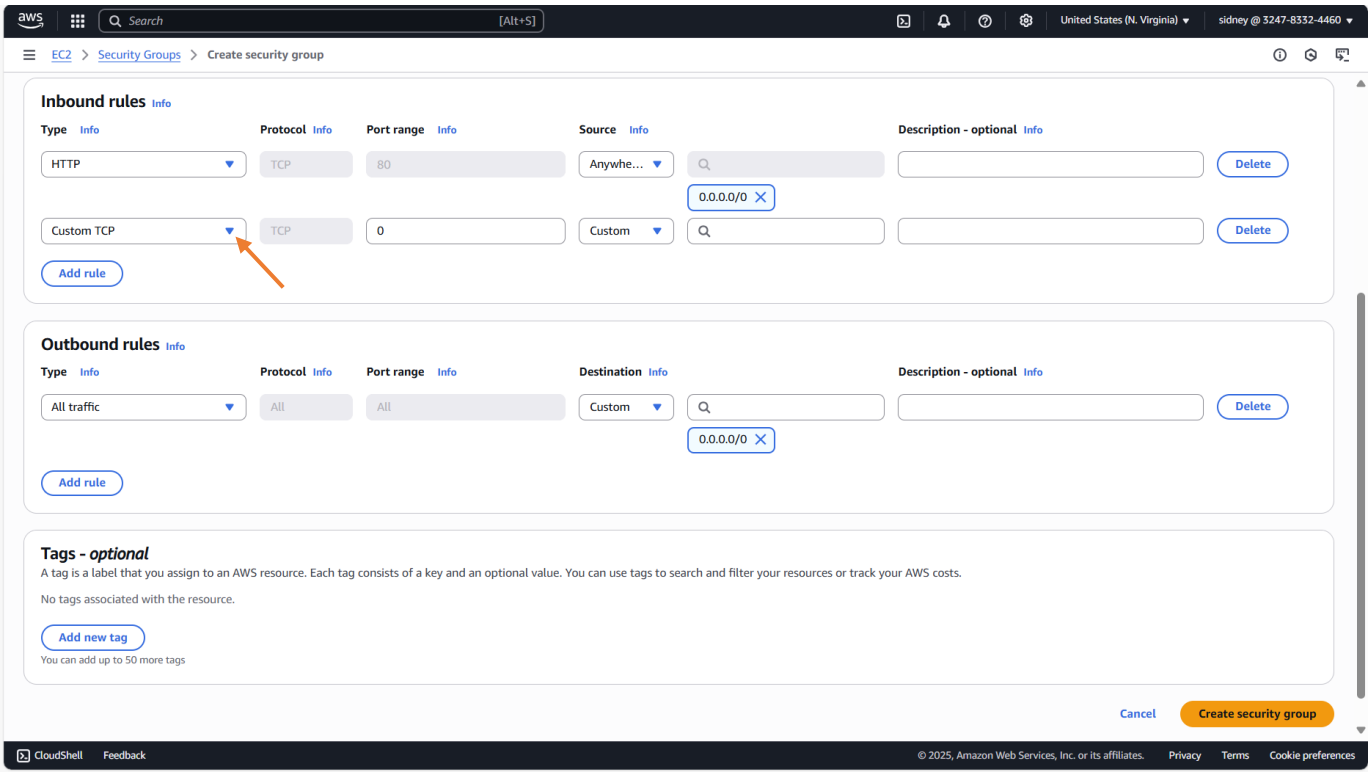
Now, we will add HTTP and SSH. Click on the drop down on "**Type**" and select "**HTTP**"



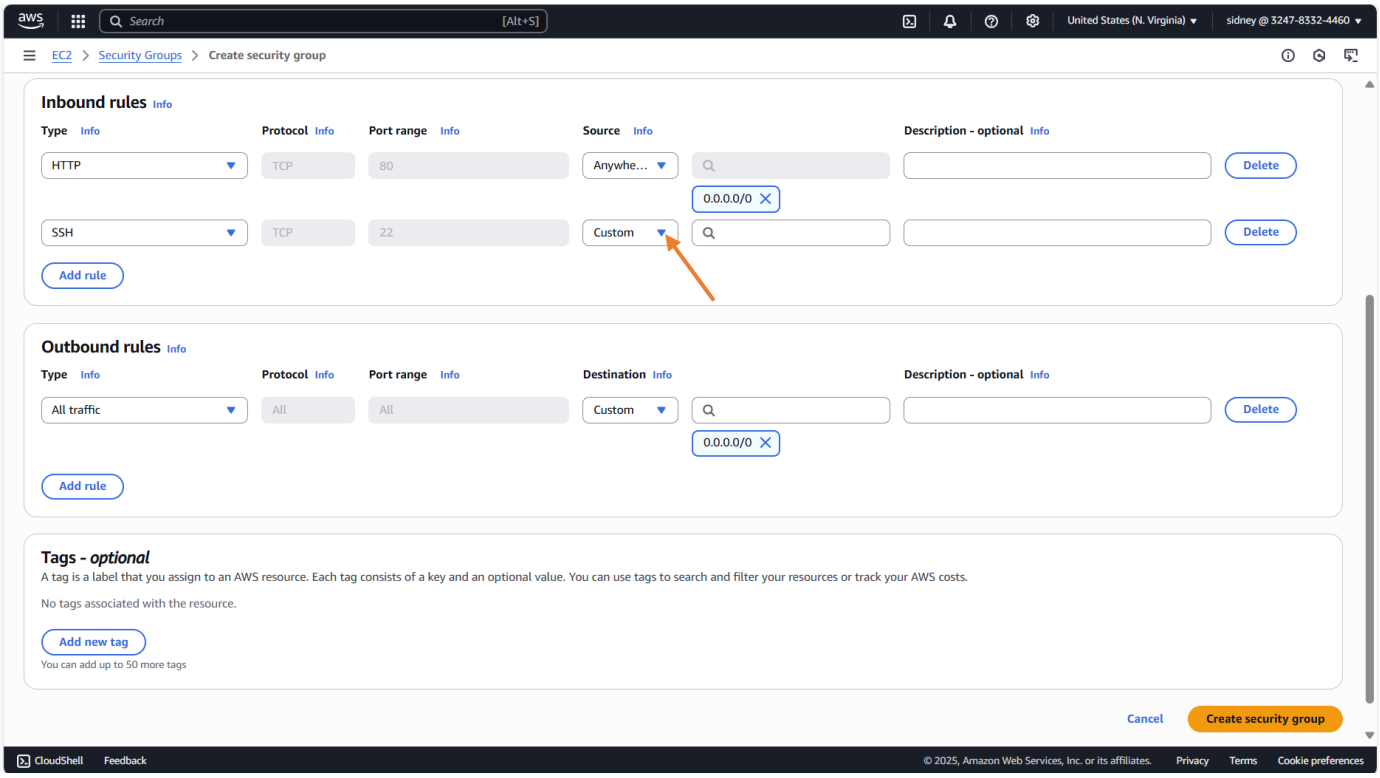
Click on the drop down on "**Source**" and select "**Anywhere-IPv4**"



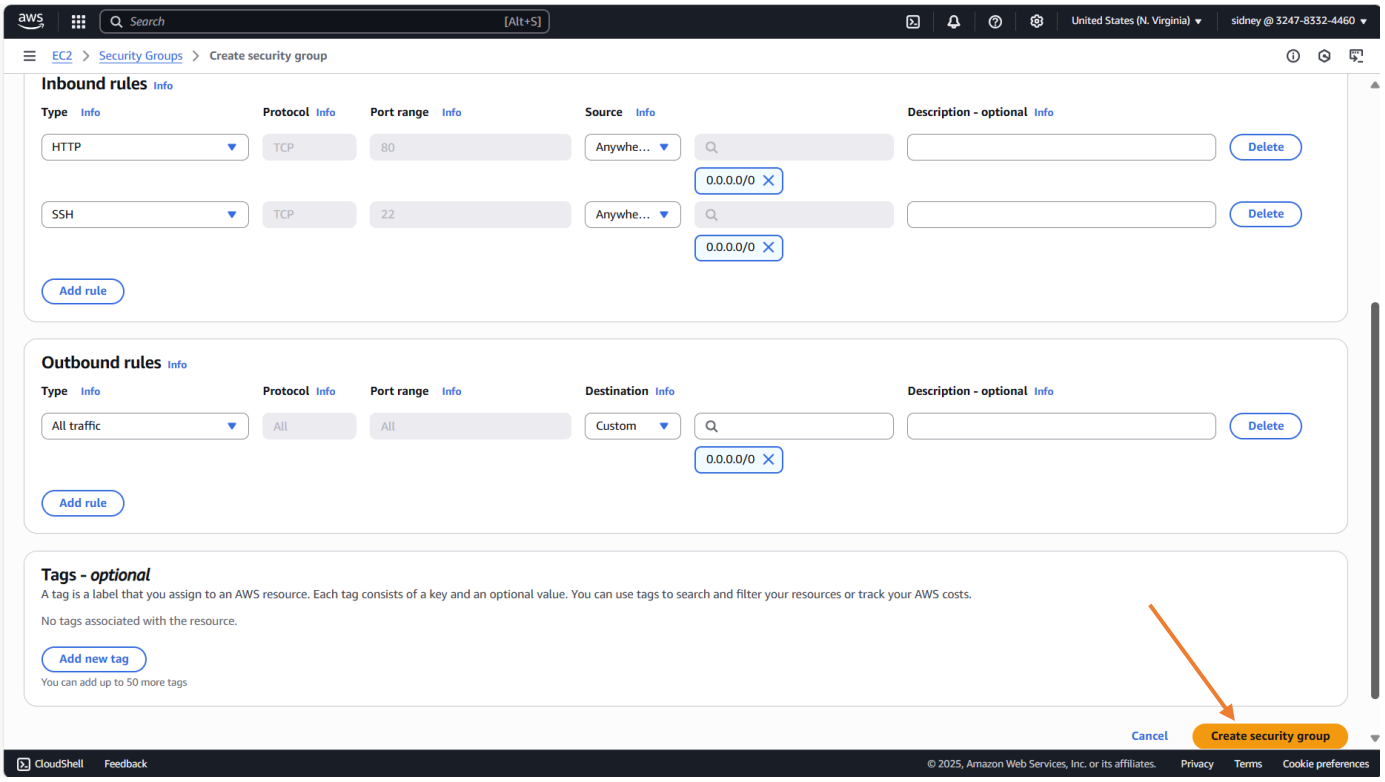
Now, let us add SSH. Click on "**Add rule**"



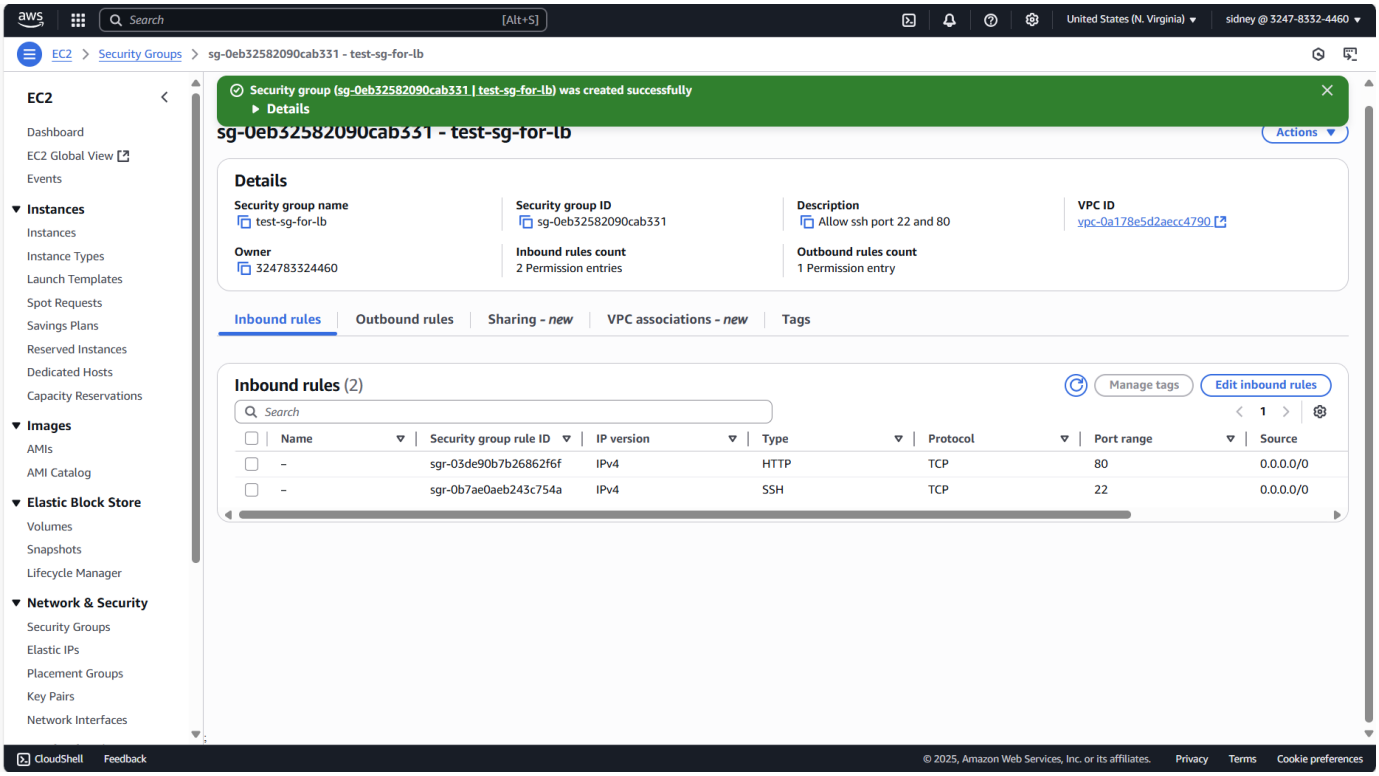
Click on the drop down on "**Type**" and select "**SSH**"



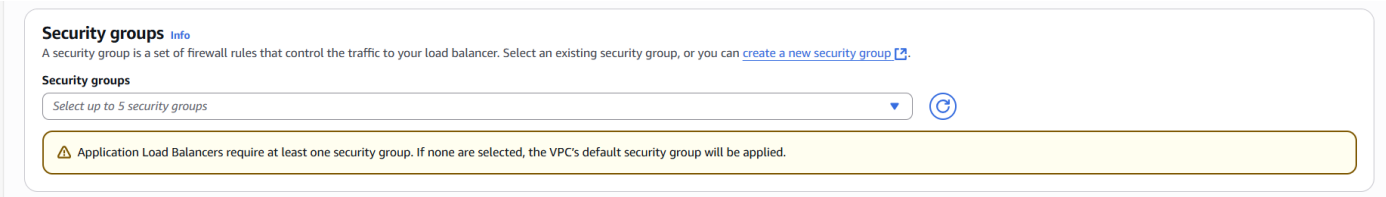
Then click on the drop down on "**Source**" and select "**Anywhere-IPv4**"



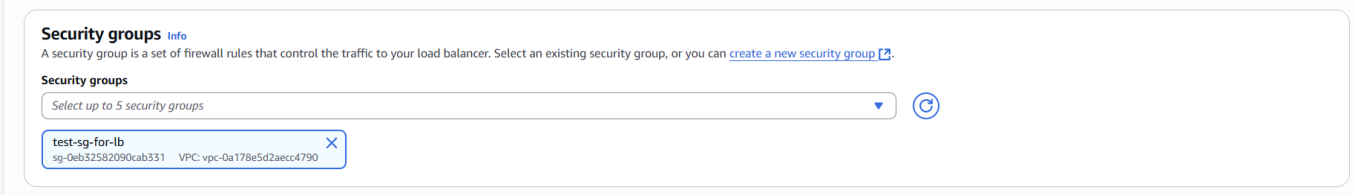
Click on "**Create Security Group**"



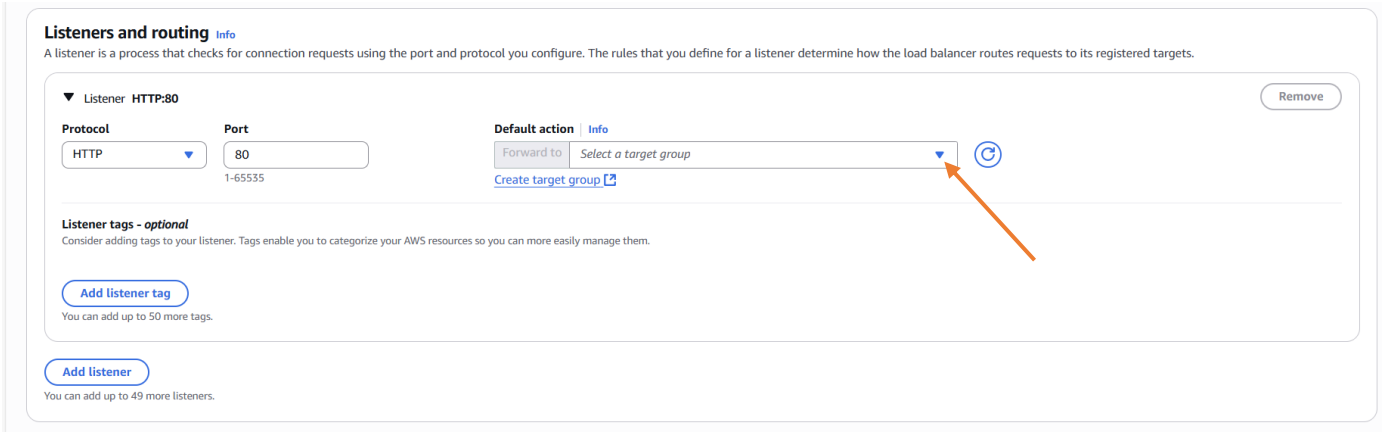
The security group has been created. Head back to "**security Group**" in our "**Application Load Balancer**" tab



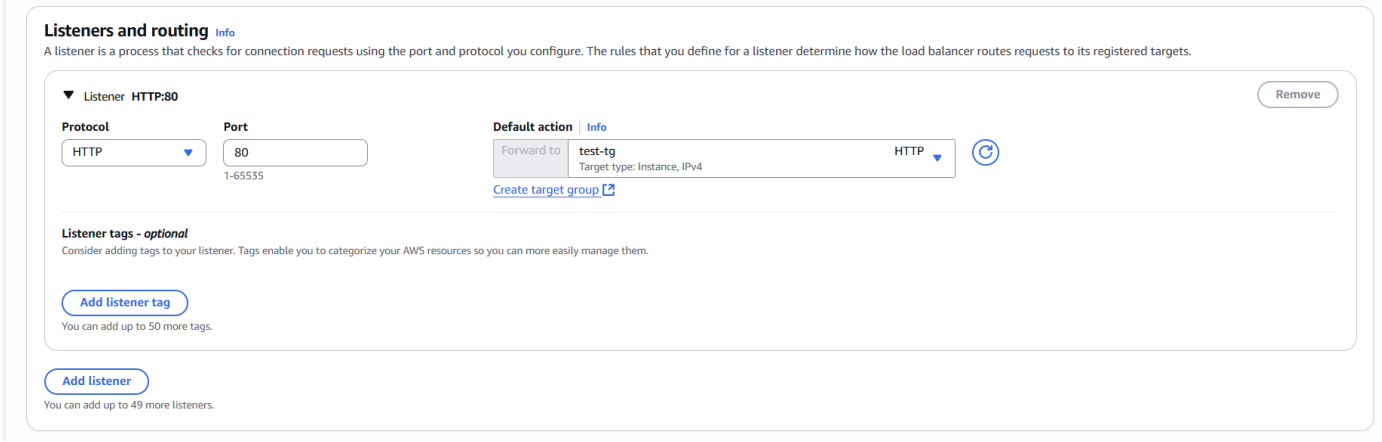
Refresh and select the security group we just created



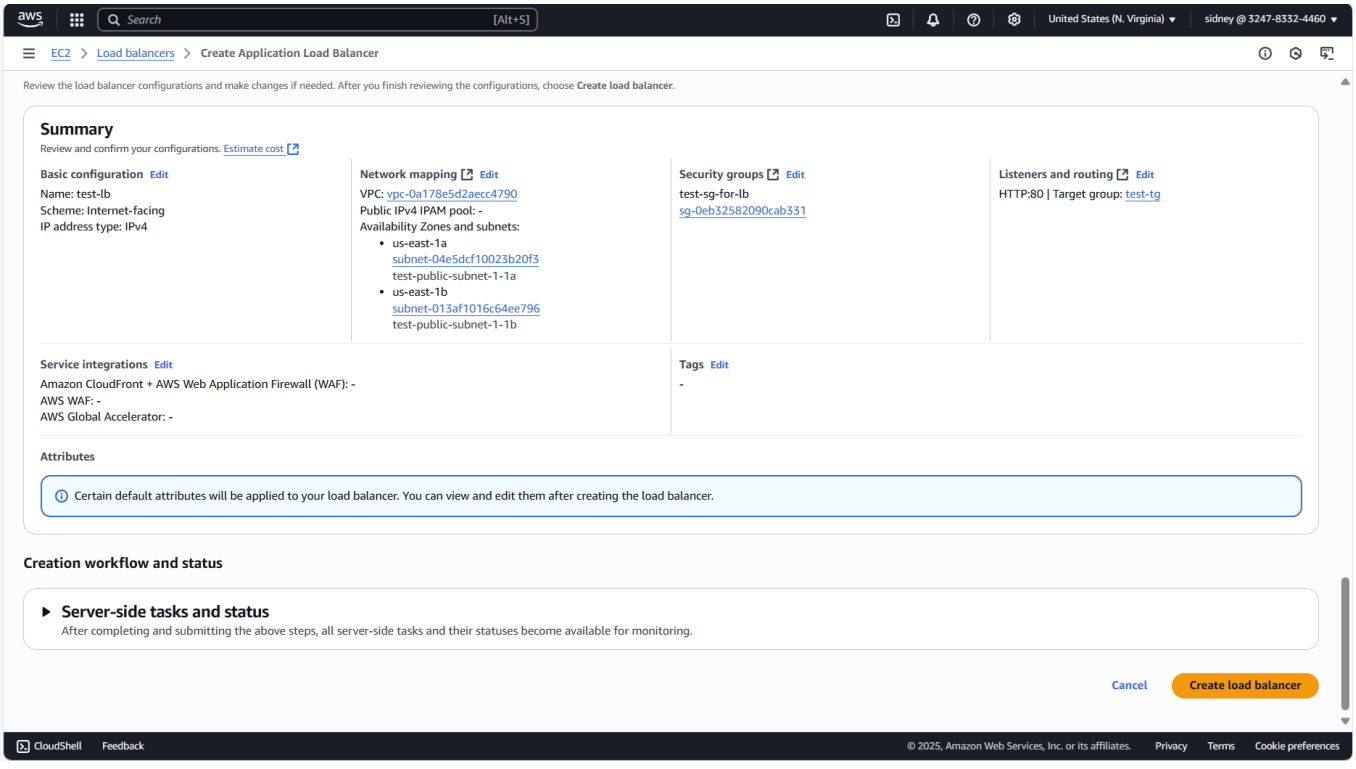
Scroll down to "**Listeners and Routing**"



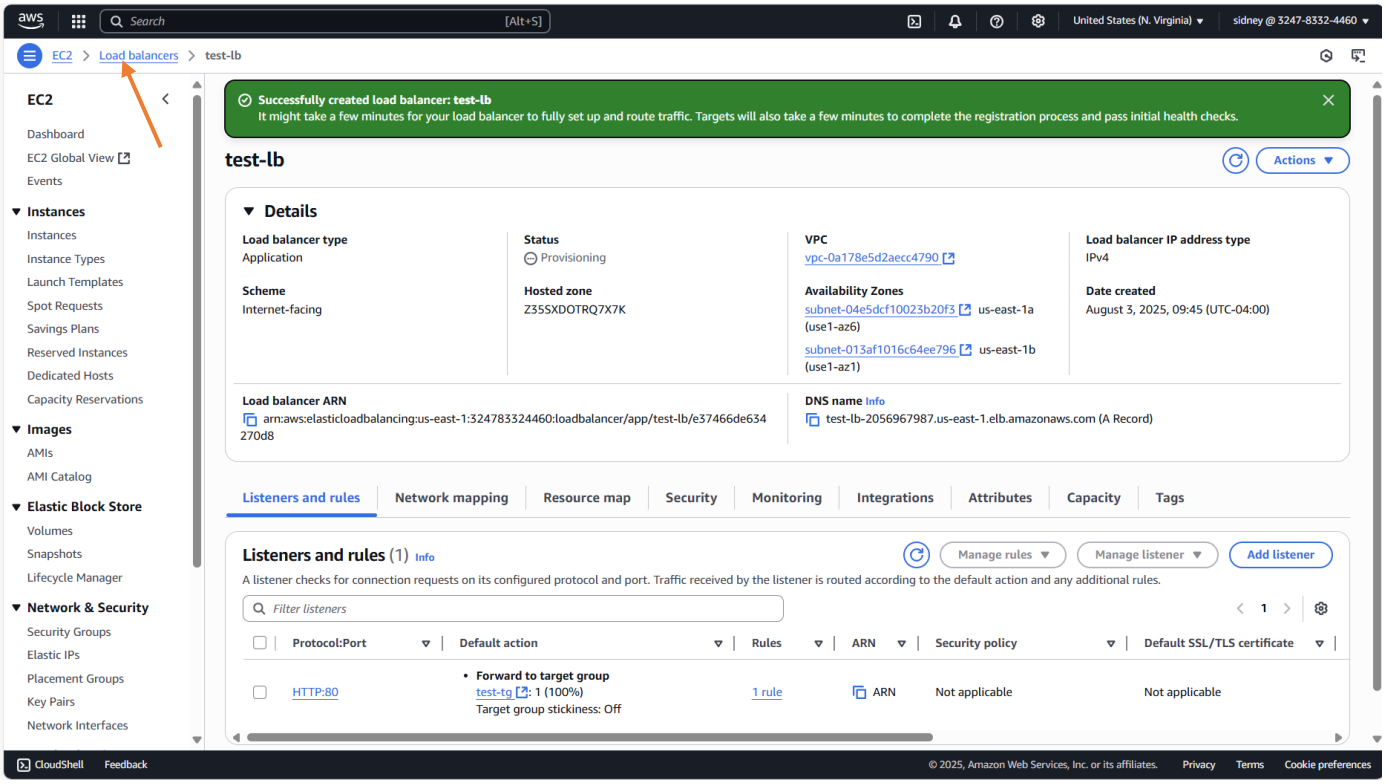
Click on the drop down on "**Target Group**" and select the target group we created



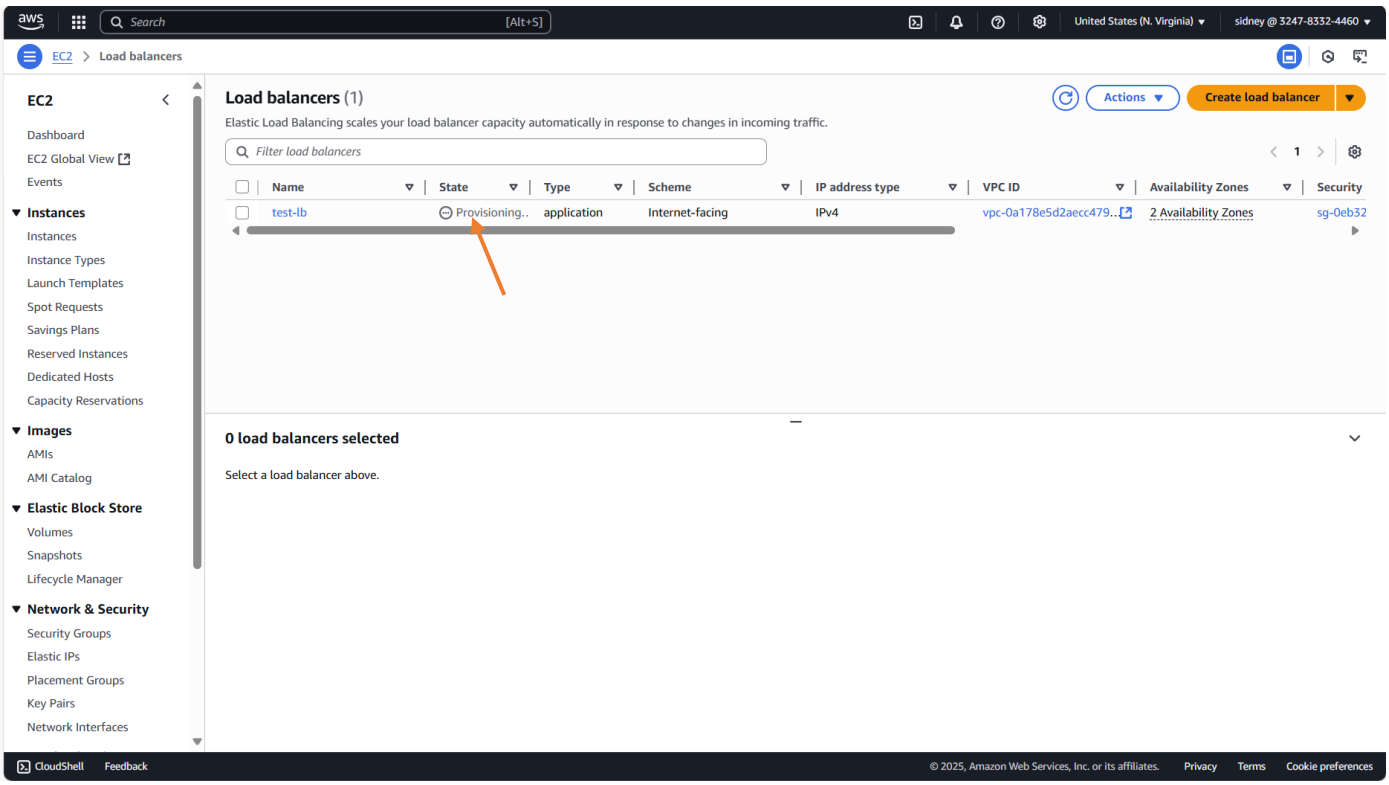
The scroll down to the end



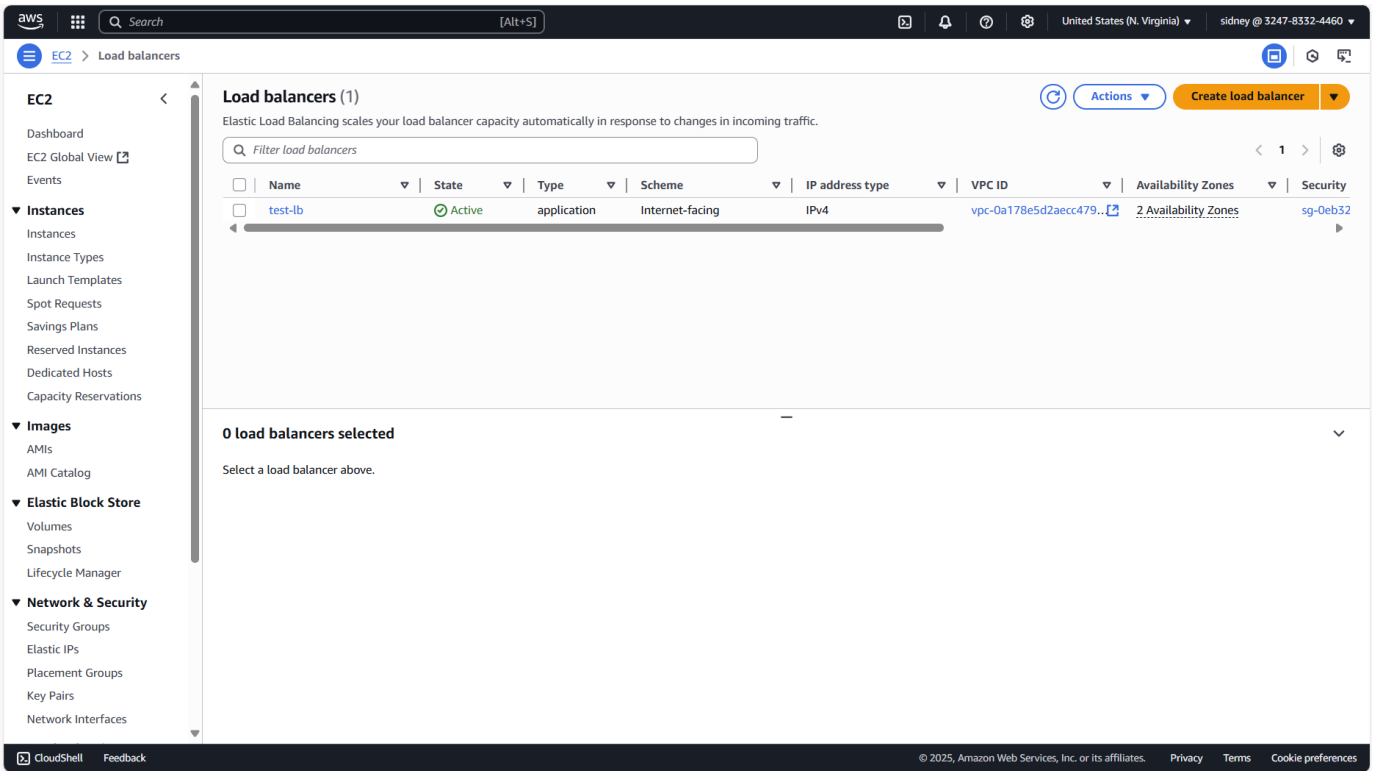
Click on “**create load balancer**”



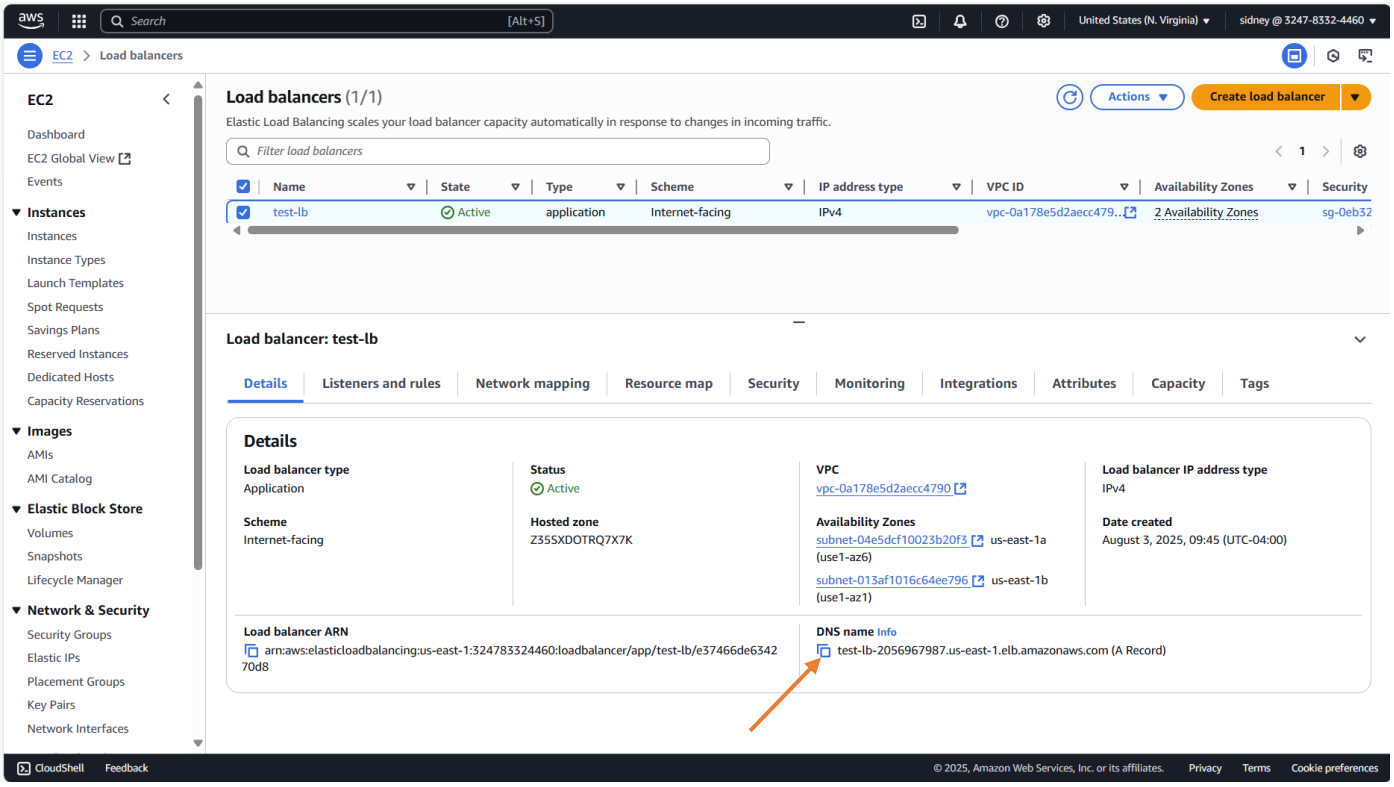
Our load balancer has been created. Click on “**Load Balancers**”



You can see that the load balancer is provisioning. It will take a couple of minutes for it to be provisioned.

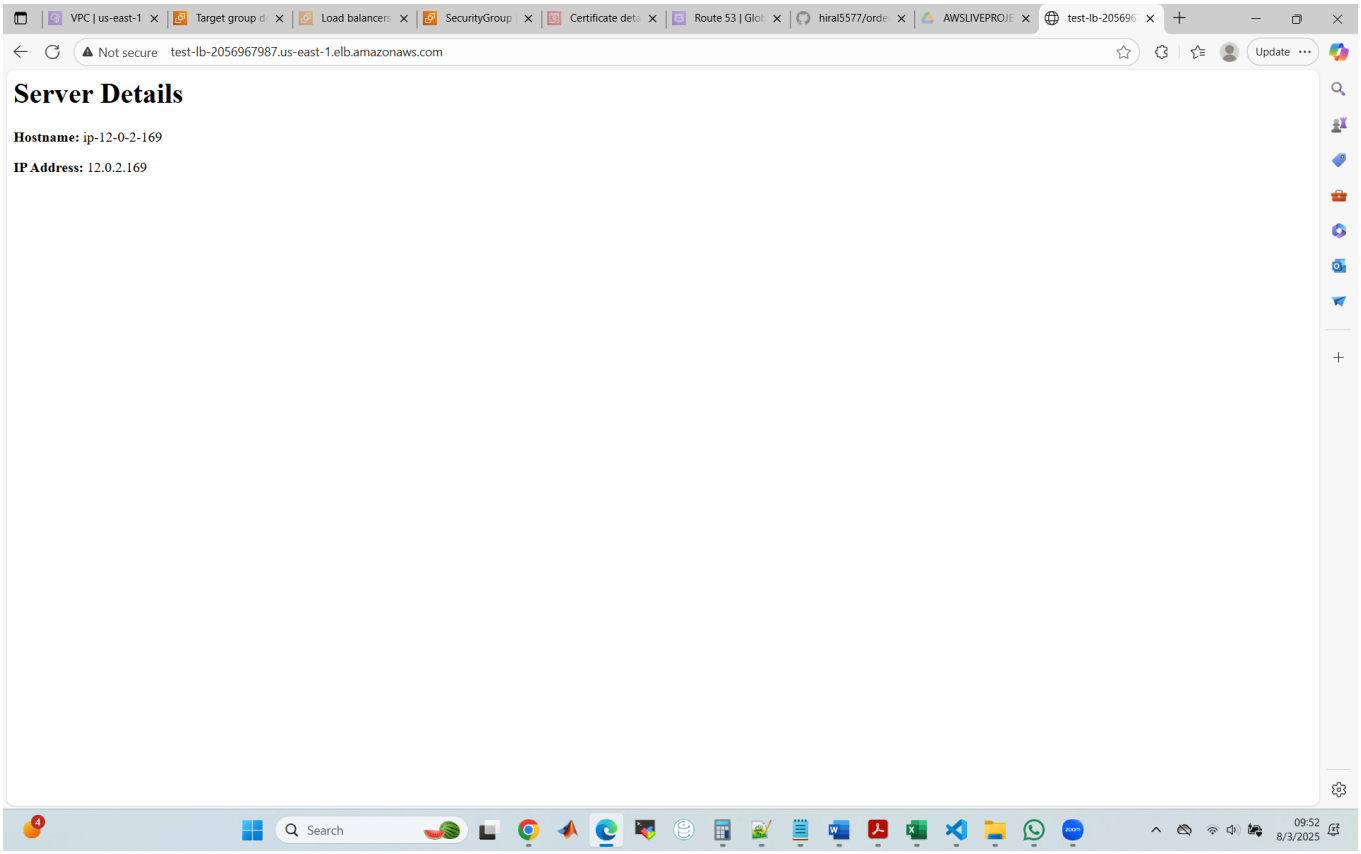


Now, the load balancer is "**Active**" and running. Select the load balancer.



Copy the DNS name and paste on your browser to test of the application load balancer is working.

```
test-lb-2056967987.us-east-1.elb.amazonaws.com
```



You can see that the load balancer is pointing to the EC2 instance which we have processed using the public IP. The DNS name has been used for this load balancer is pointing to our EC2 instance.

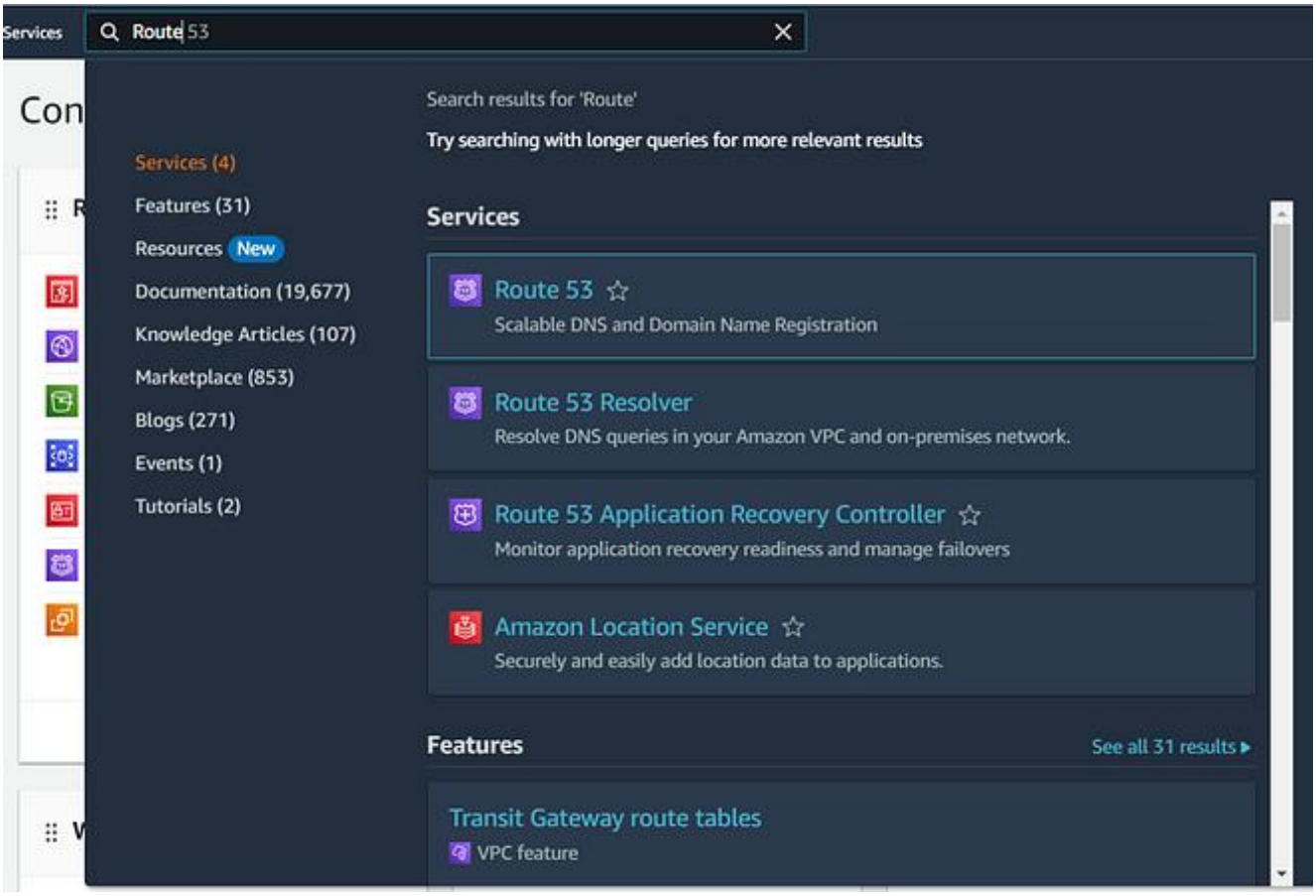
STEP 7: Setting up Amazon Route 53

In this step we are going to create a Hosted zone using Amazon Route 53. We do not want to use the Load Balancer DNS name; we want to use our own domain name. So, we need to introduce our Route 53.

Route 53 does not reside in the VPC, it is a global resource. Whenever you create a Route 53 or a hosted zone in Route 53, then it does not belong to any region or VPC. At last, we will have our domain name which will be used.

Once we create our Route 53, the user is going to use our URL and the URL is going to point to the Route 53 and then it will go to the Load Balancer and then the load balancer will point to our EC2 instance.

We start by searching for Route 53 on the AWS Management Console.

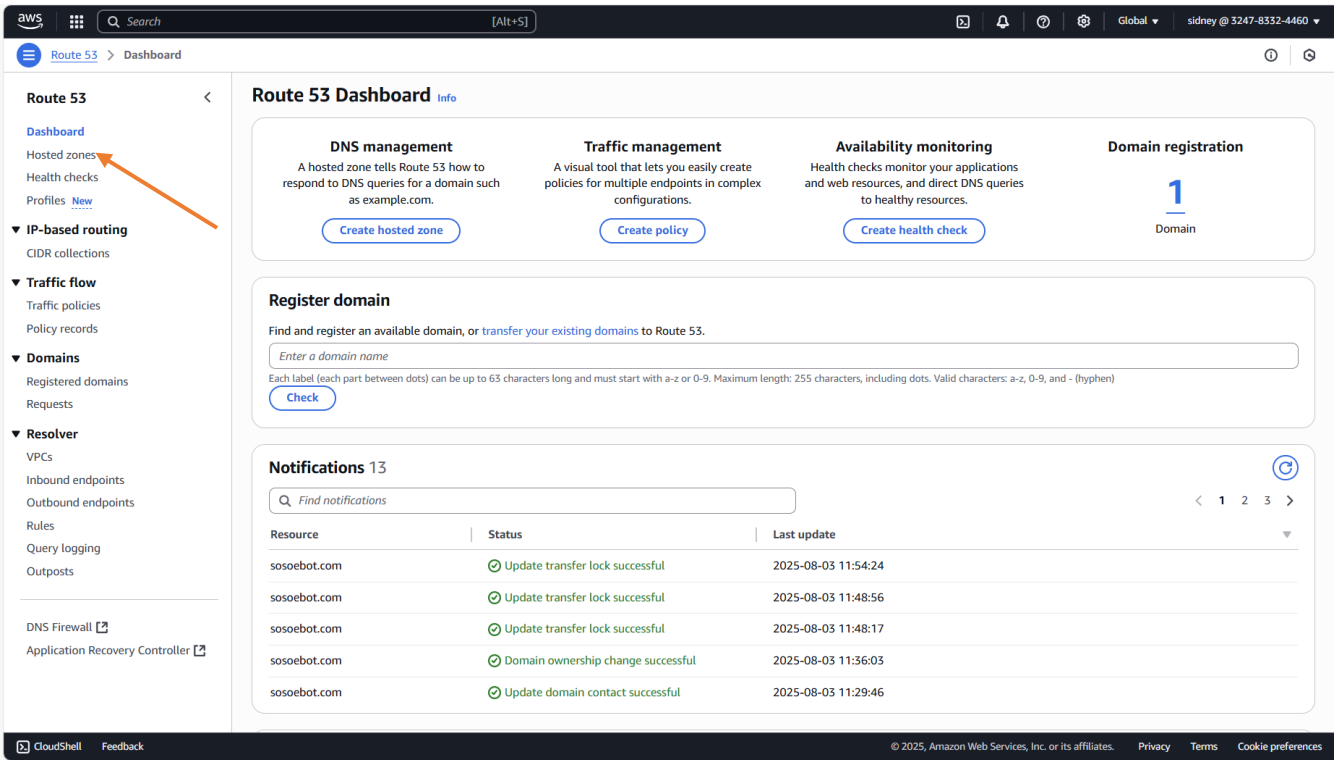


Click on "**Route 53**" under services

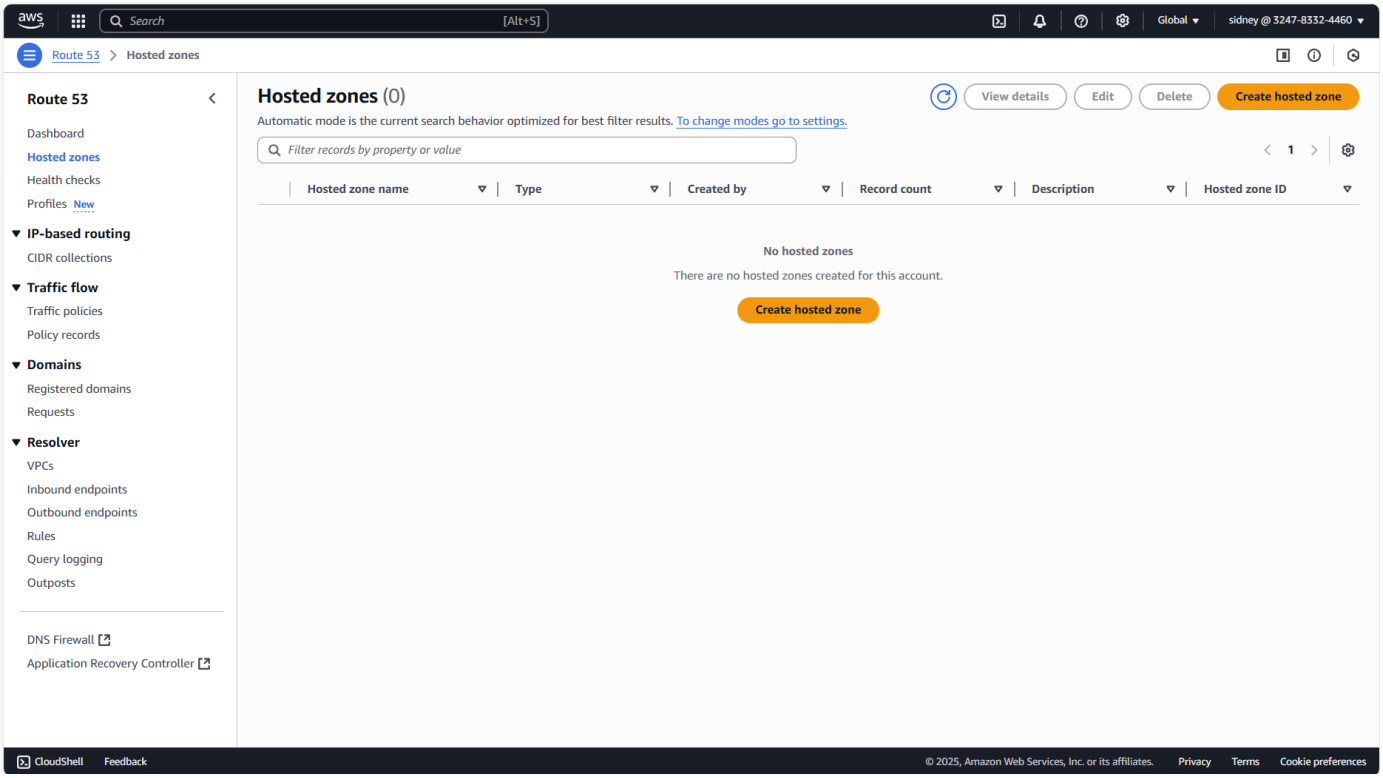
What is a Hosted Zone?

A hosted zone represents a set of records that belongs to a specific domain

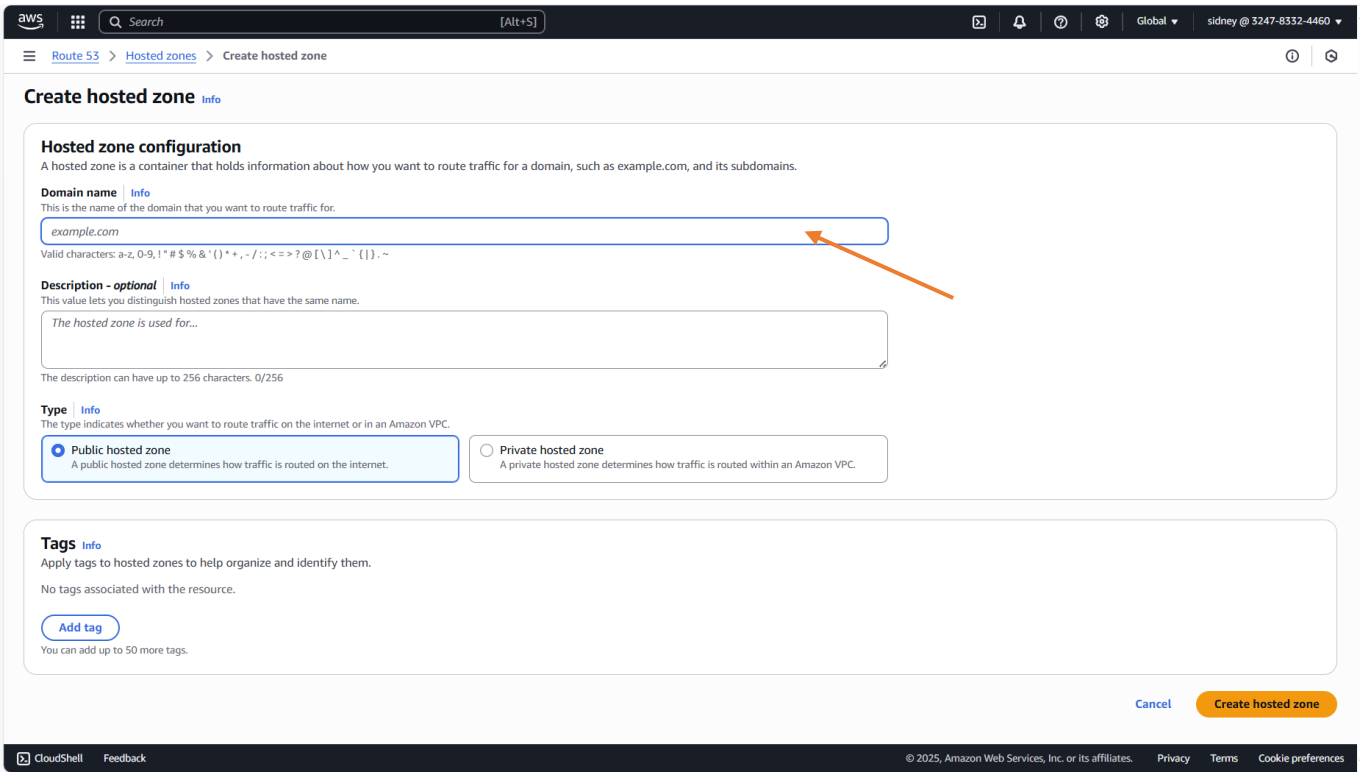
Click on "**Hosted Zones**" on the Left-hand side.



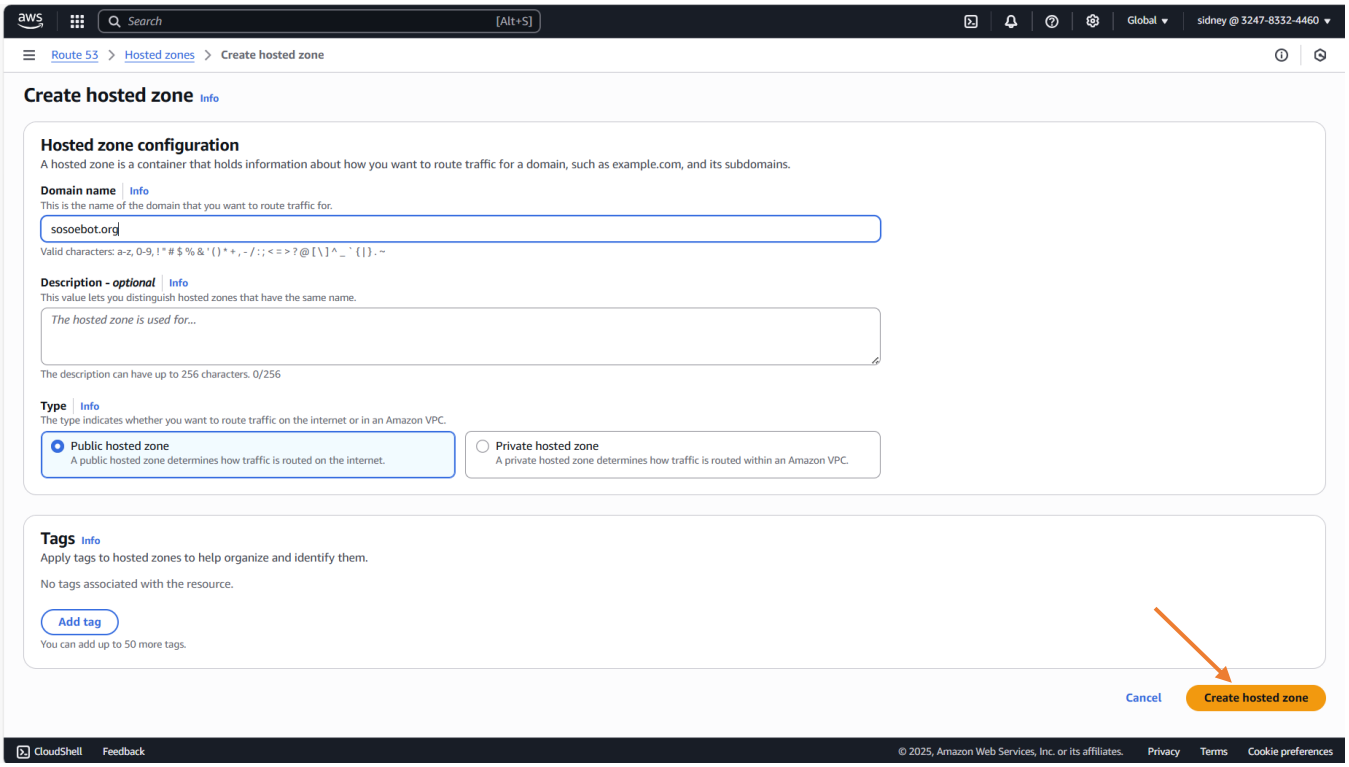
Click on “**Hosted Zones**”



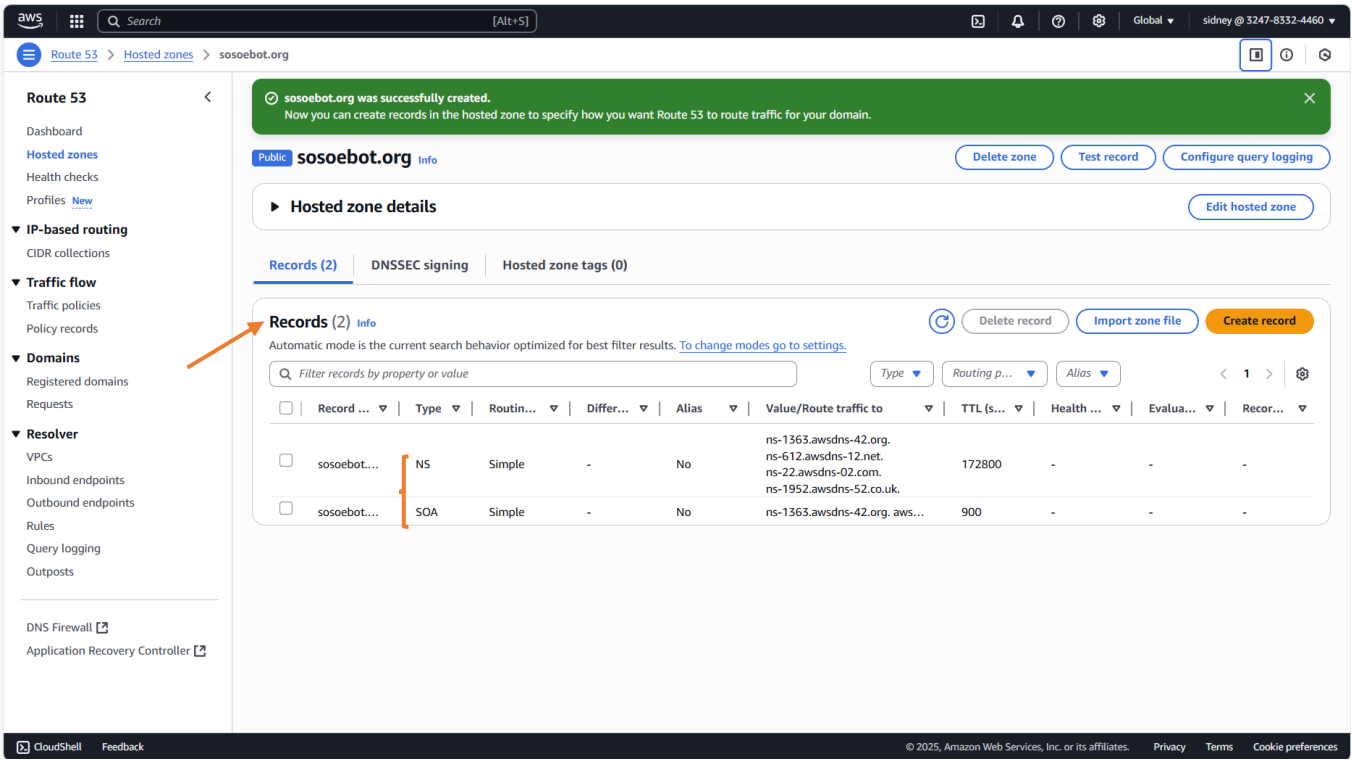
Click on “**create Hosted Zone**”



Enter our domain name "**sosoebot.org**"



Click on "**create hosted zone**"



You can see that our hosted zone has two records with types NS (Name server) and SOA in it.

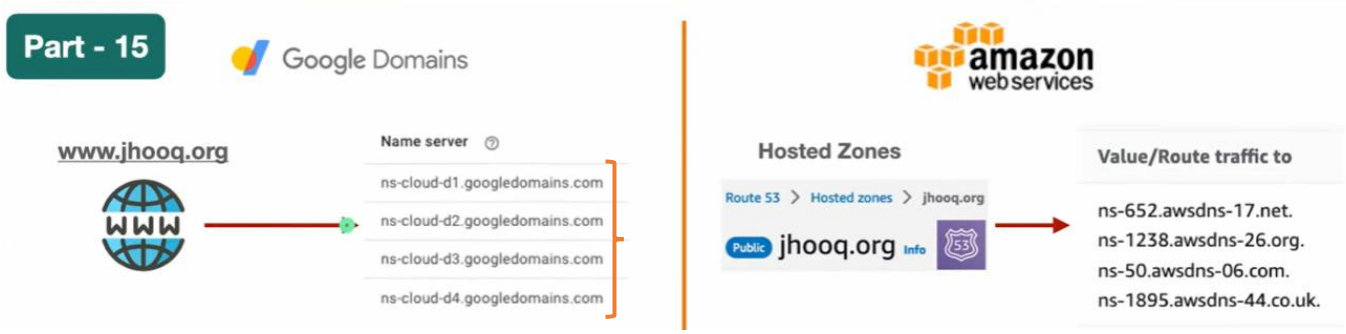
NAME SERVER

I would like to explain the concept of name server which is name server setting concept. If you purchase a domain name from a vendor other than Amazon, for example Google domains. You will have to do some name server settings.

We have created a hosted zone. Once we create our hosted zone, we get our name server record which is from AWS, given below:

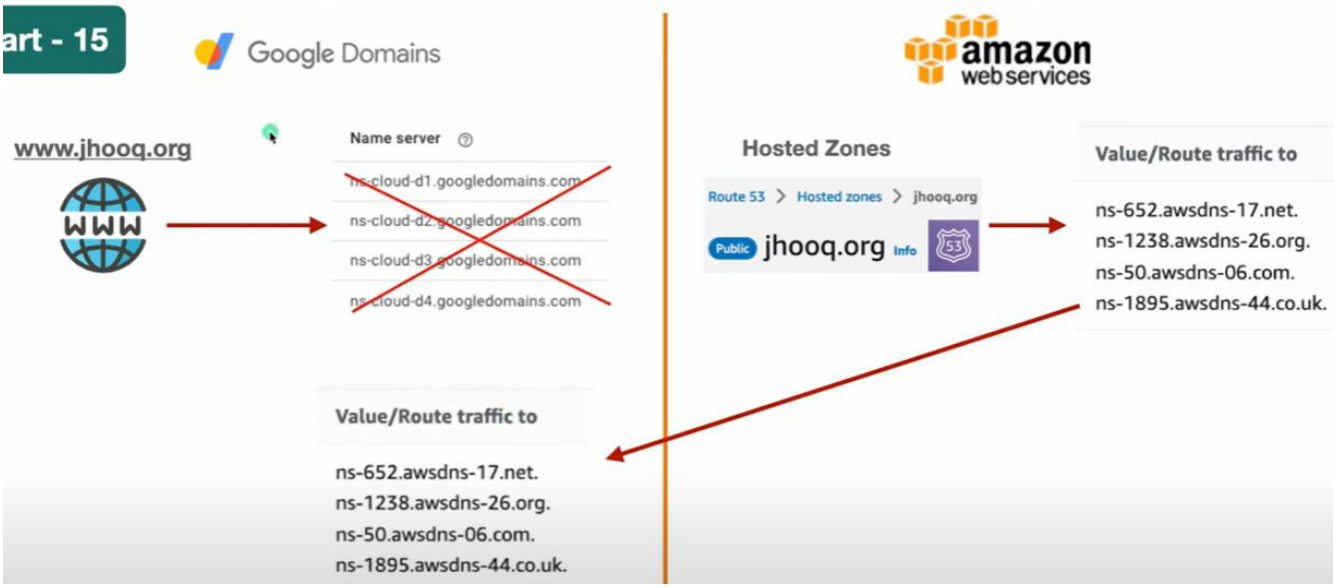
```
ns-1363.awsdns-42.org.
ns-612.awsdns-12.net.
ns-22.awsdns-02.com.
ns-1952.awsdns-52.co.uk.
```

Also, if you purchase a domain name from any vendor other than AWS, for example: Google domains. The domain name will also have their own name servers.

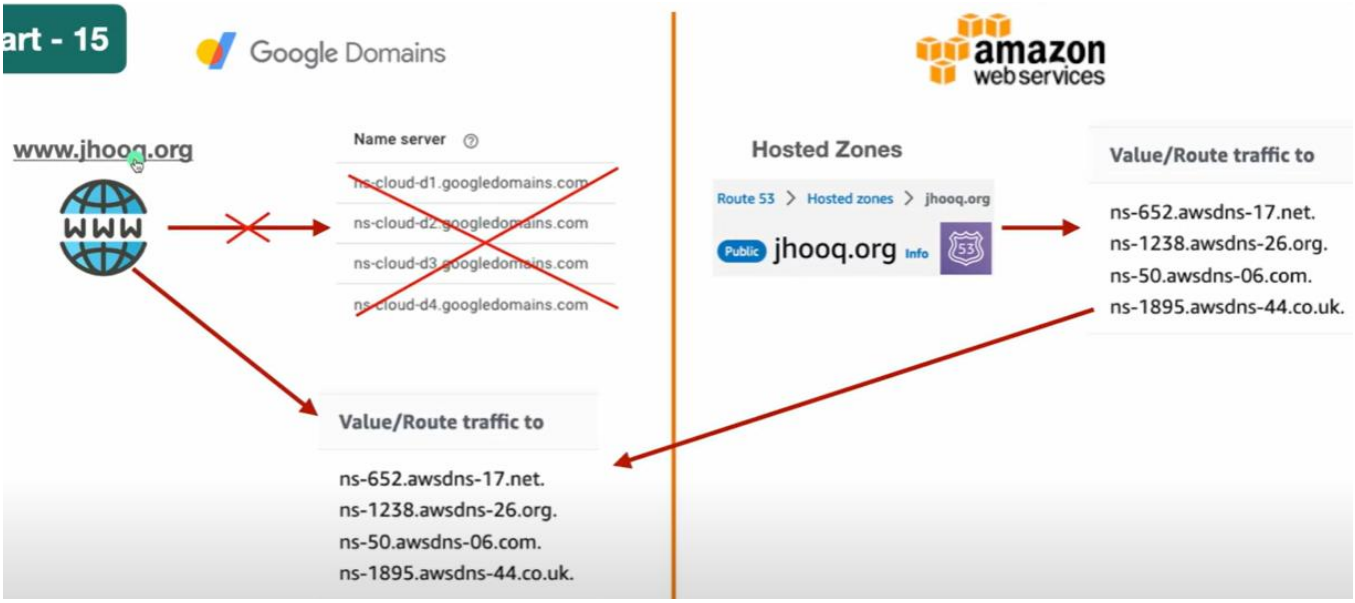


As you can see above, these are records from Google domain. To point the Google domain to AWS services. We will not be able to use the Google Domain name servers.

So, we need to reply on the name server record of AWS. We will copy the records from AWS and paste the record in the Google Domain's name server record.

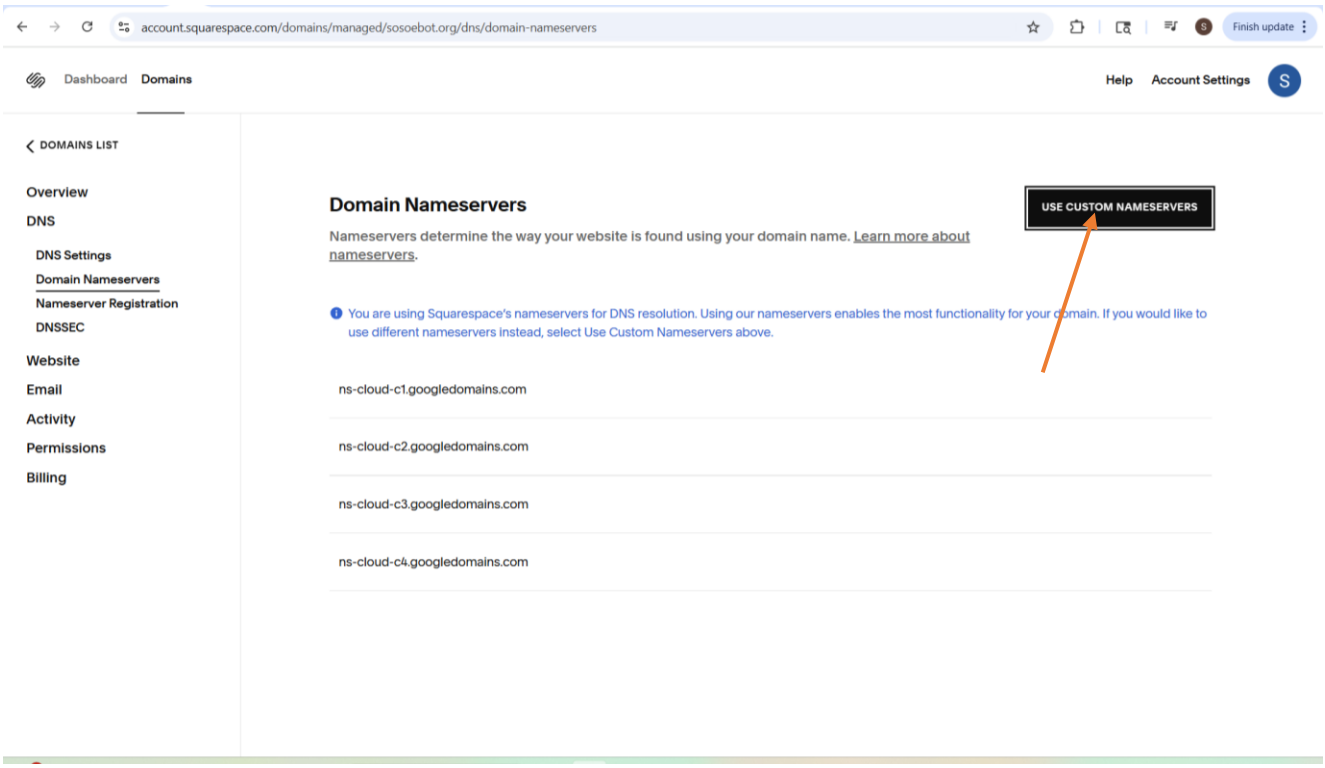


So, the Google Domain will be pointing to the Domain name server of AWS instead of the Google name server.

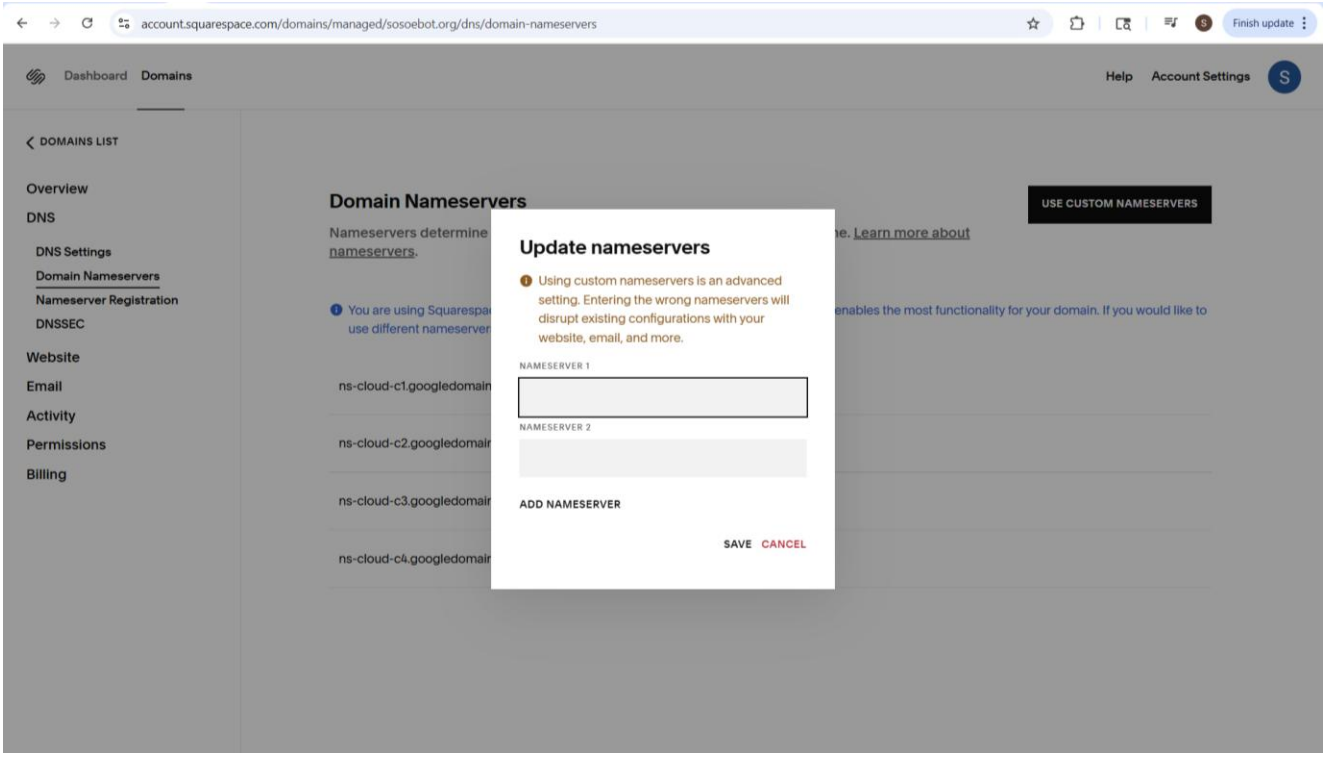


This is how the name server settings works when you purchase the domain name from another service provider.

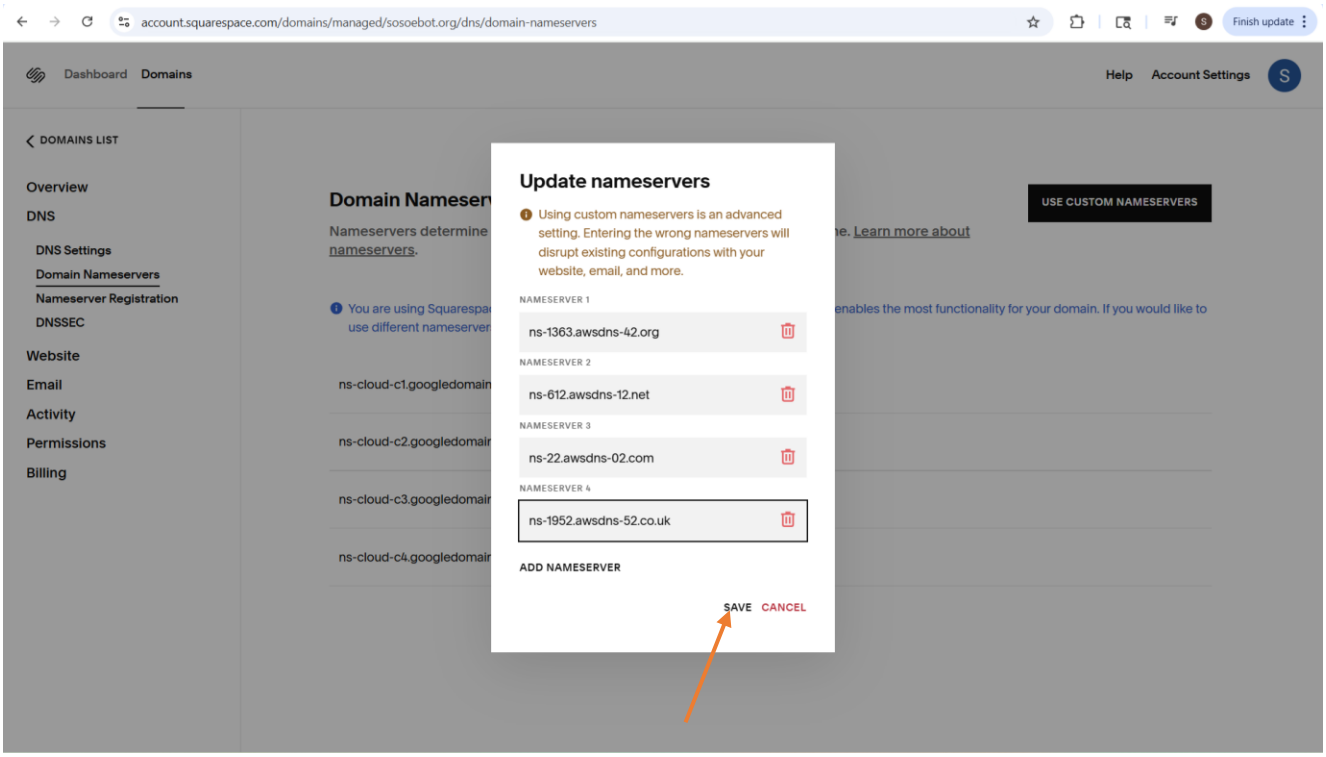
So, we have to modify the Name servers in our third-party Domain name



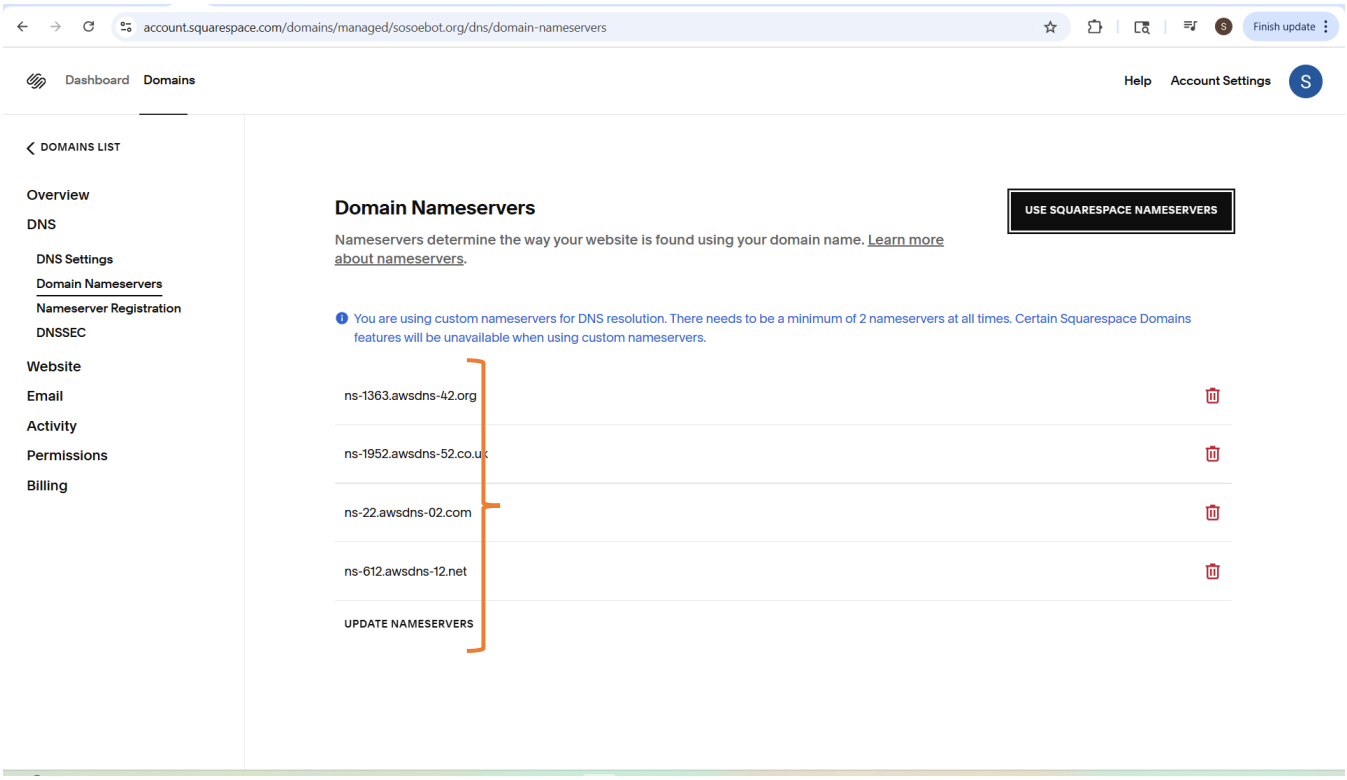
Click on "**Use Custom Nameservers**"



Paste the four copied Name servers we copied from AWS here leaving out the **point (.)** at the end

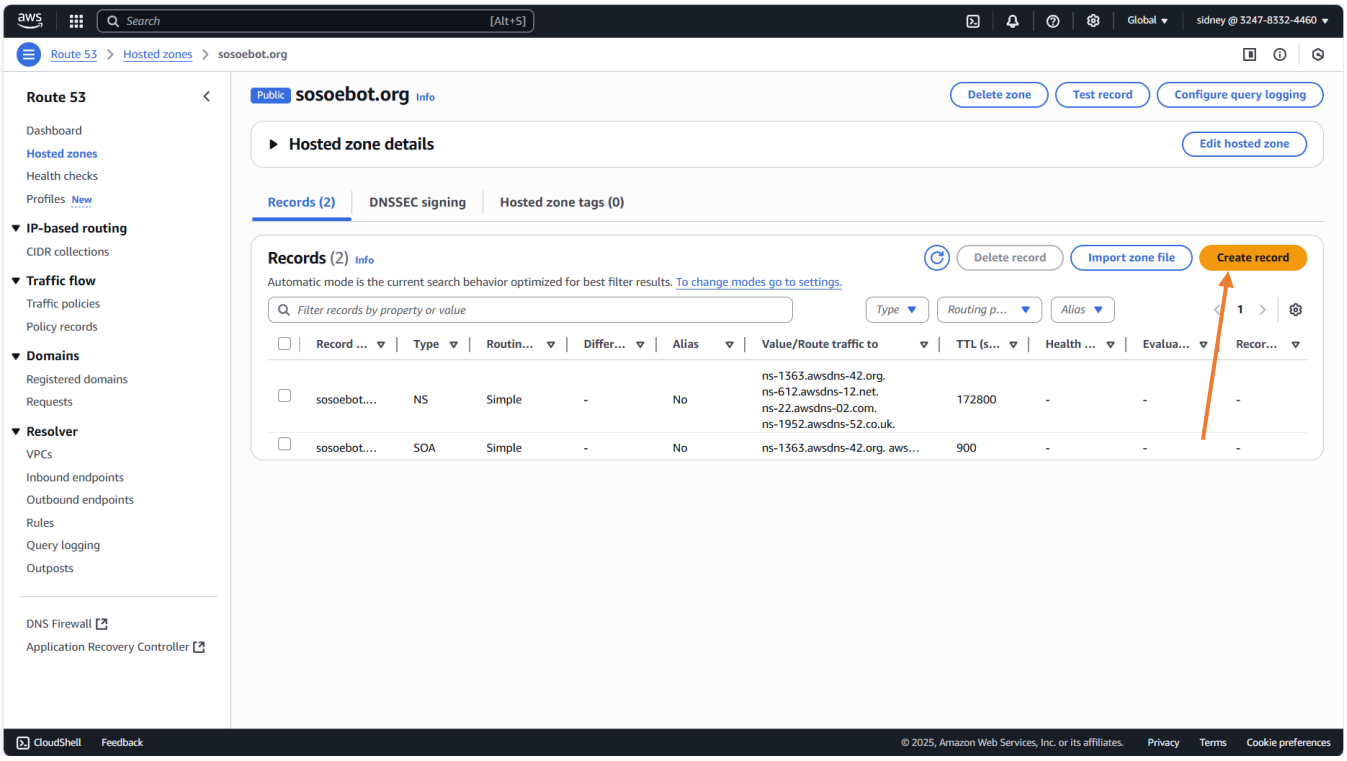


Click on "**Save**"



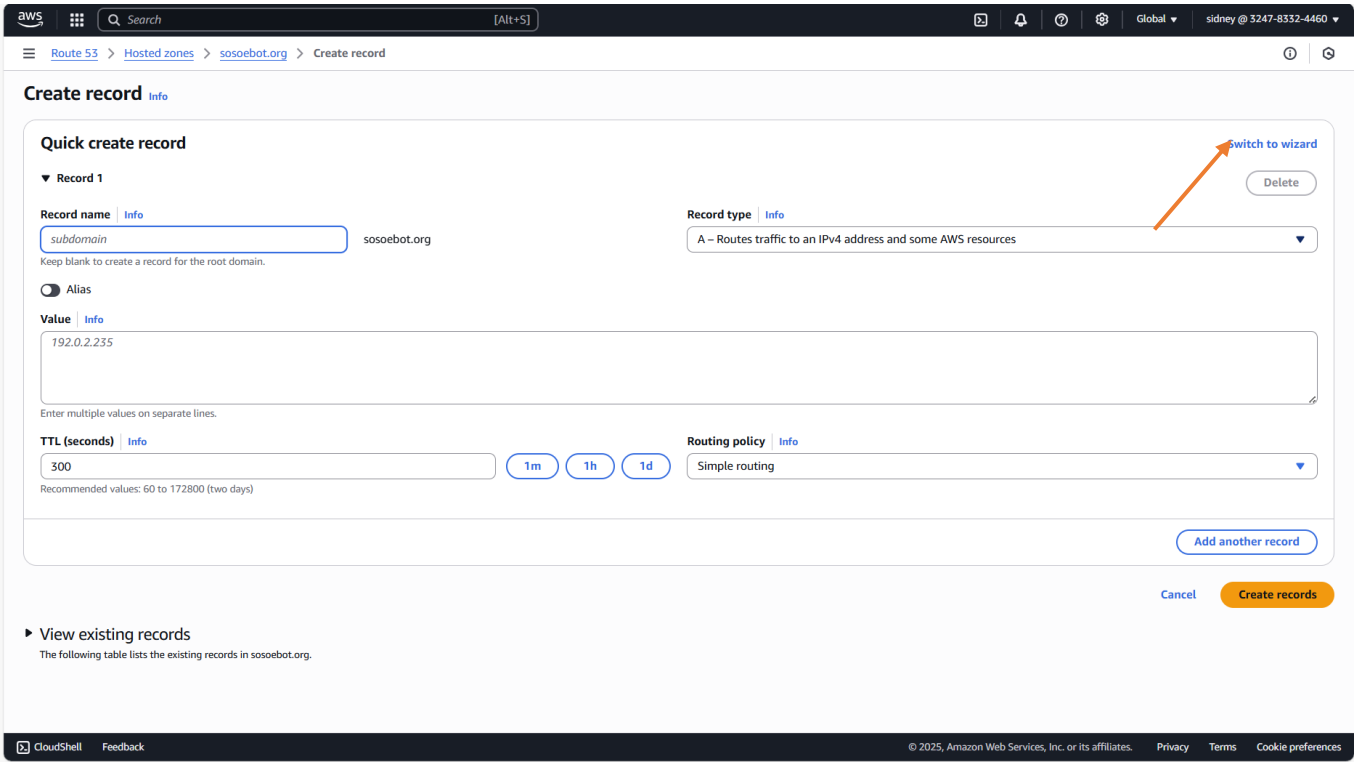
You can see that we have modified the name servers in our third-party DNS to match those in AWS.

Let us head back to our Hosted Zone and create an "**A -record**"

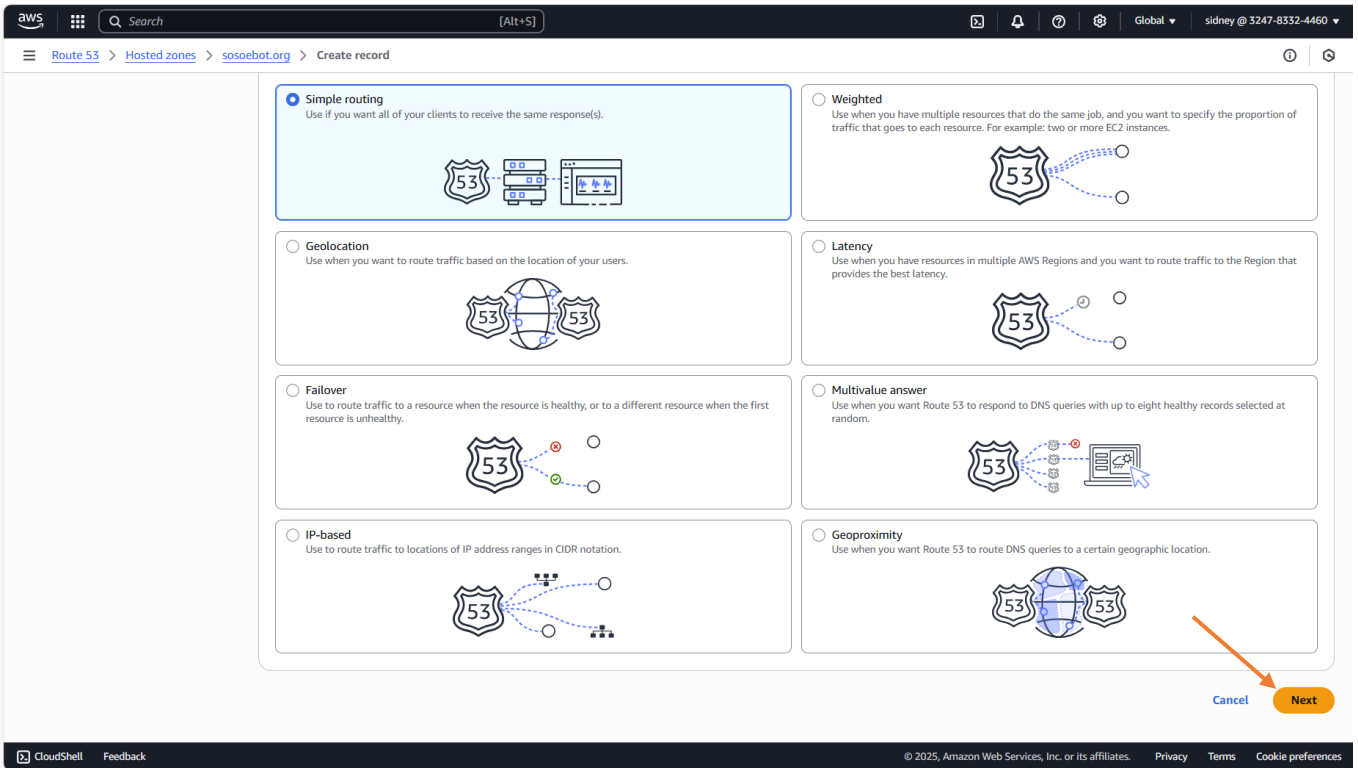


Now, we need to create an "**A record**" so that it to points to our Load Balancer.

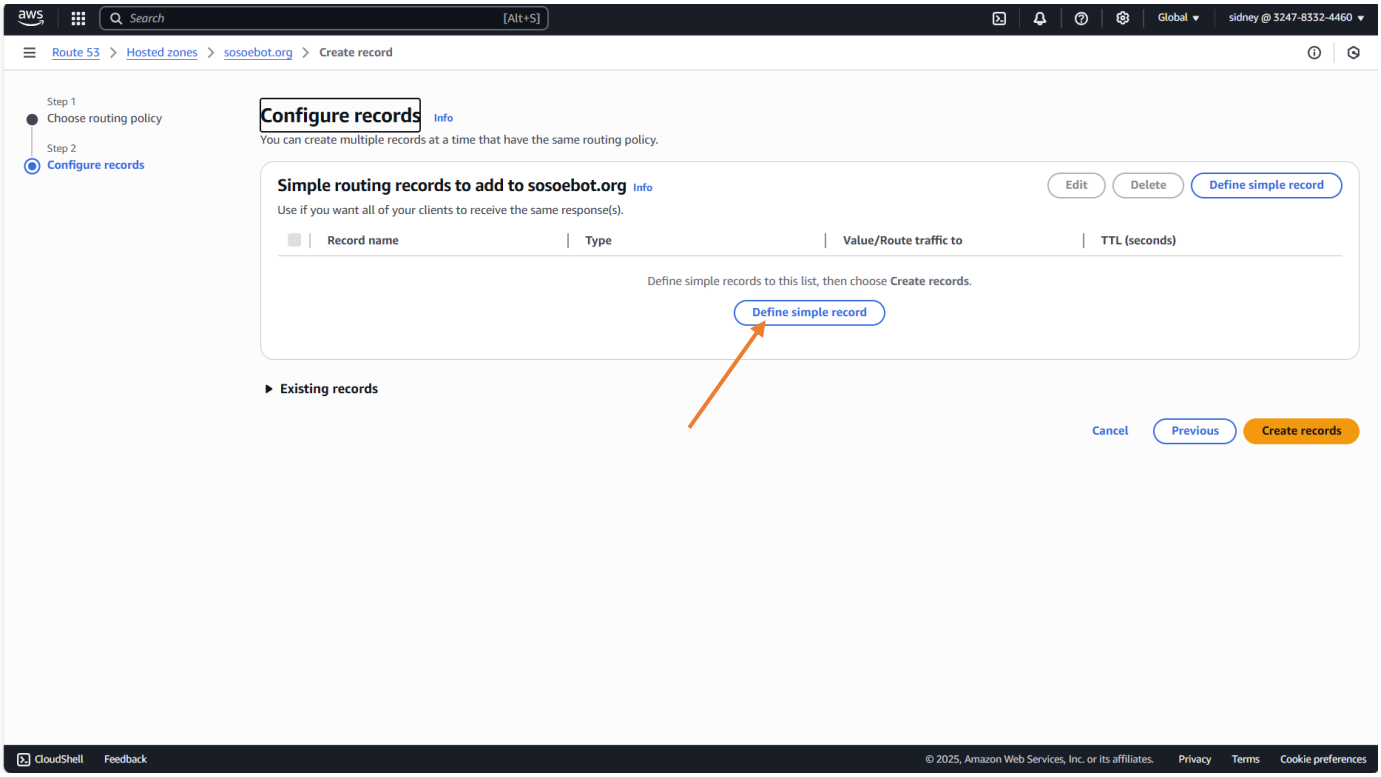
Click on "**Create Record**"



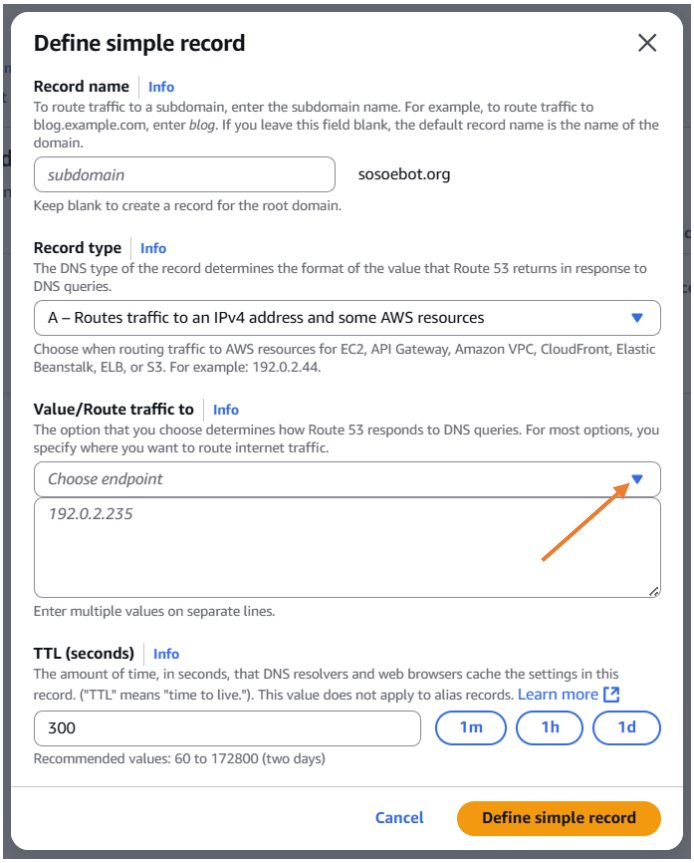
Change the view by clicking on "**Switch to Wizard**"



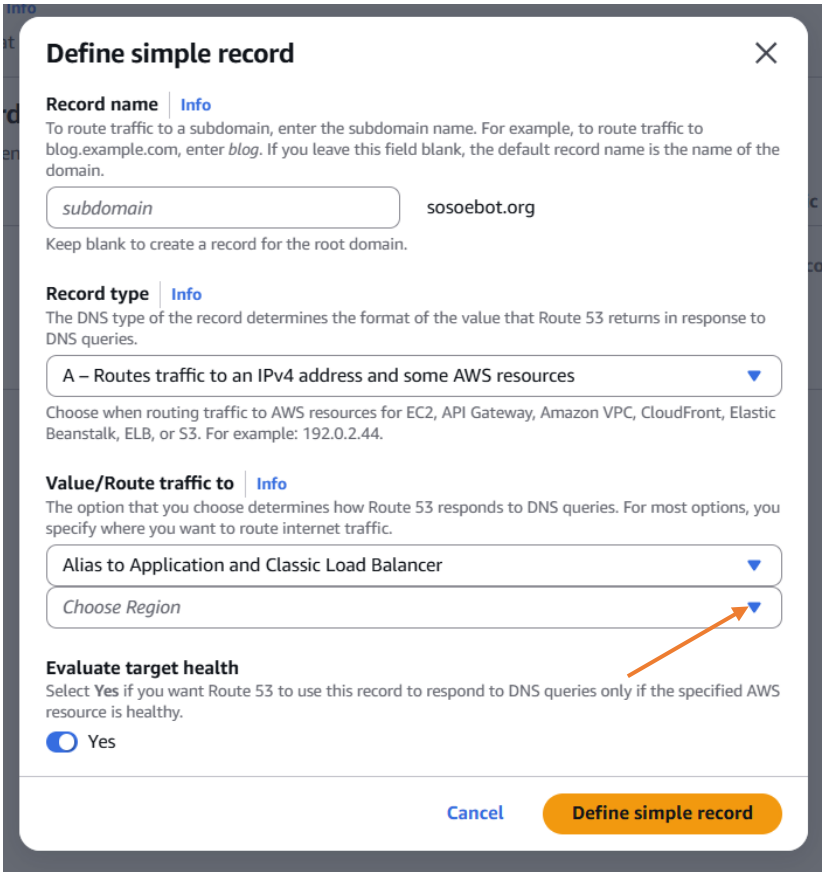
Select "**Simple Routing**" and click on "**Next**"



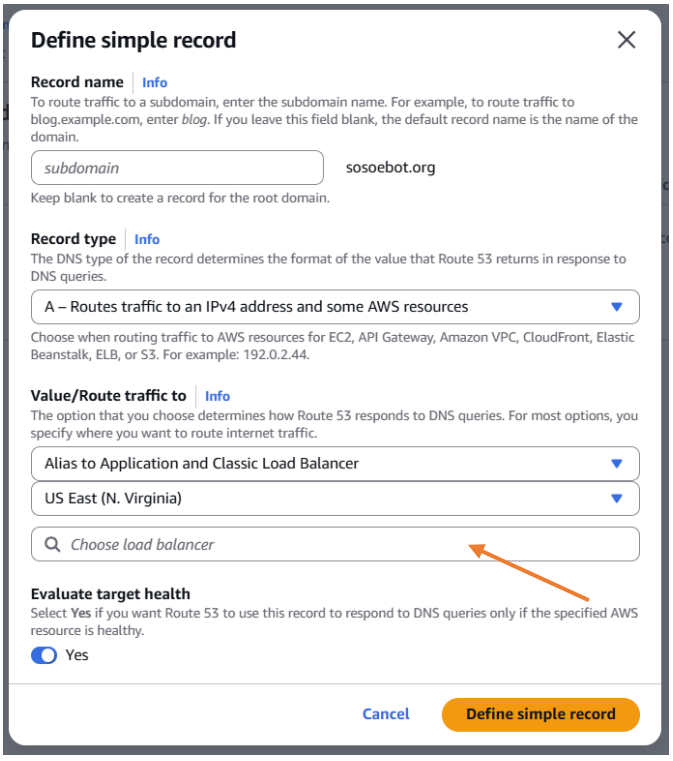
Click on "**Define simple record**"



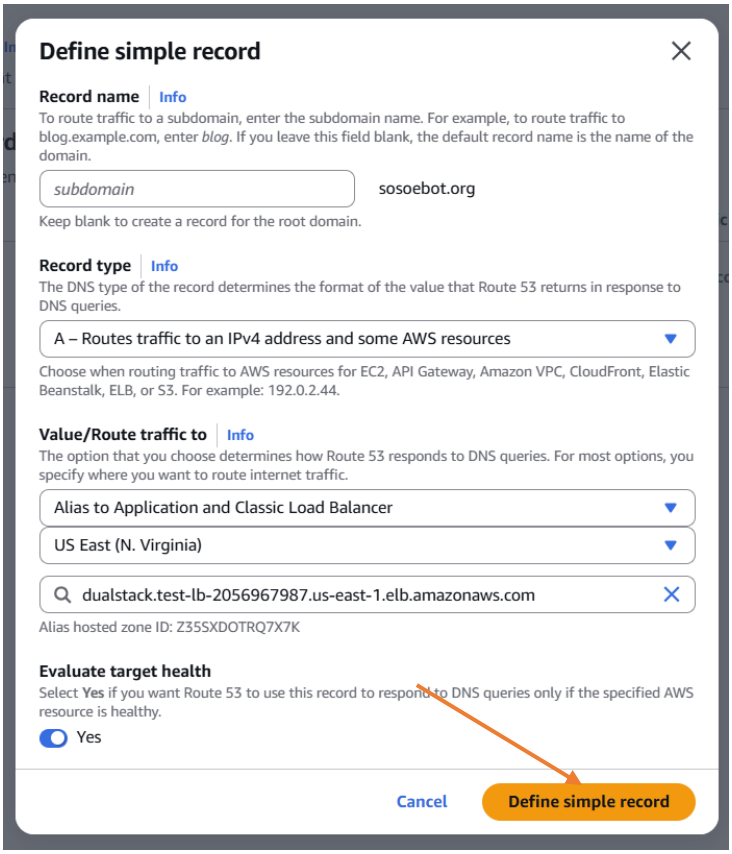
Click on the drop down on "**Route Traffic to**" and select "**Application and Classic Load Balancer**"



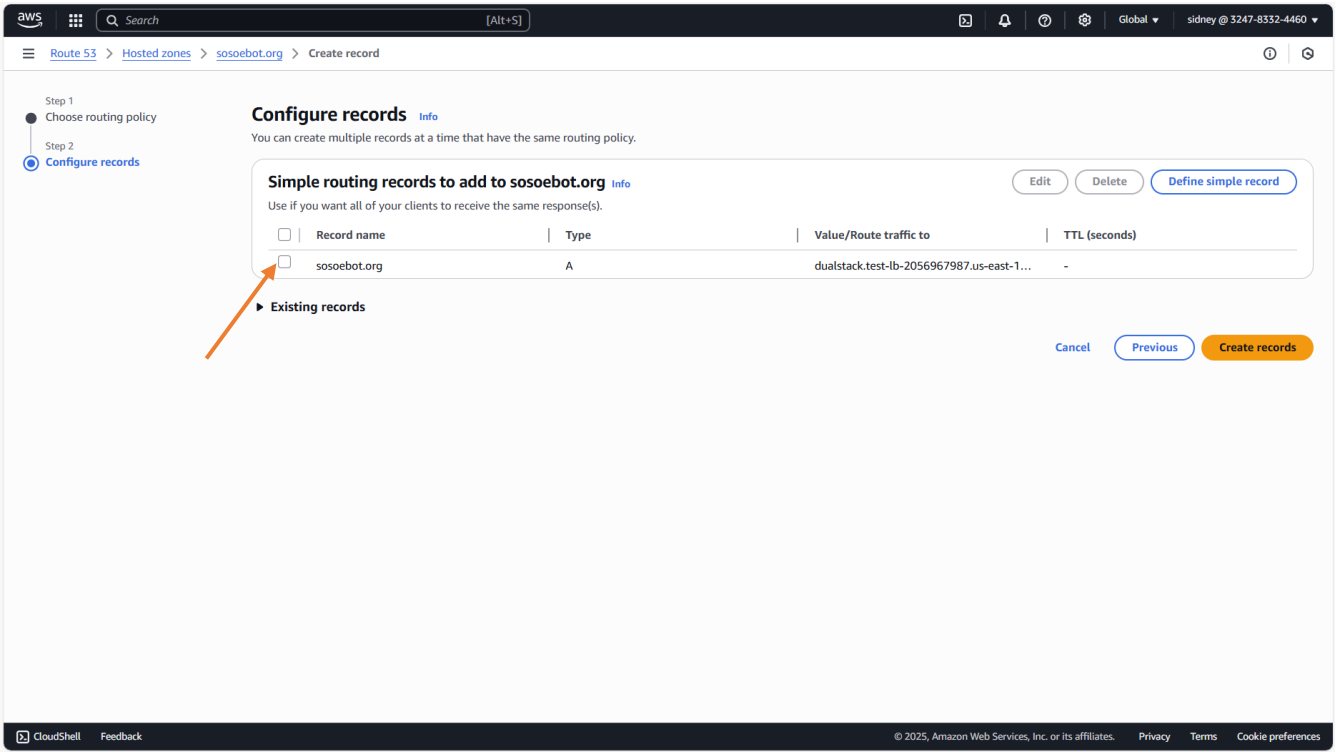
Click on the drop down on "**Region**" and select "**us-east**"



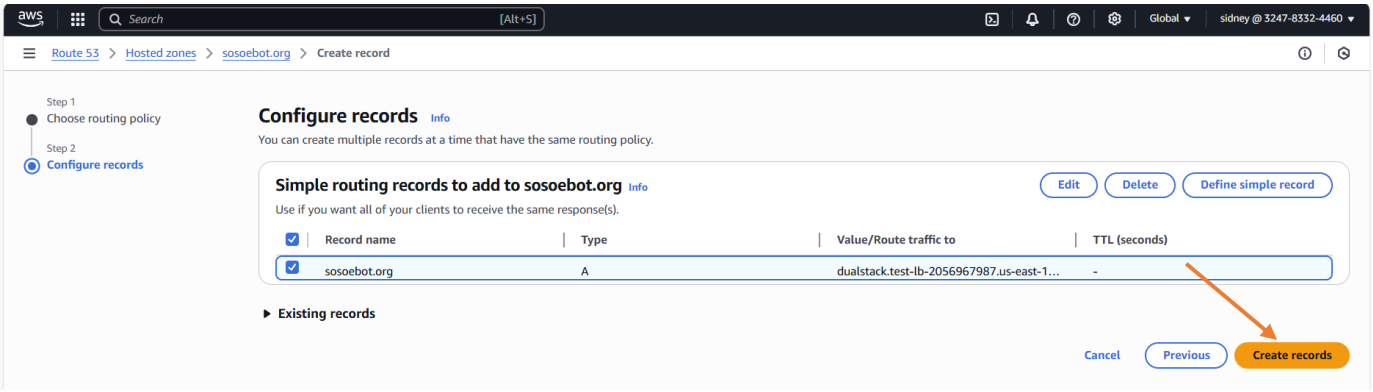
Then click on "**Choose Load Balancer**" and select the application load balancer we created.



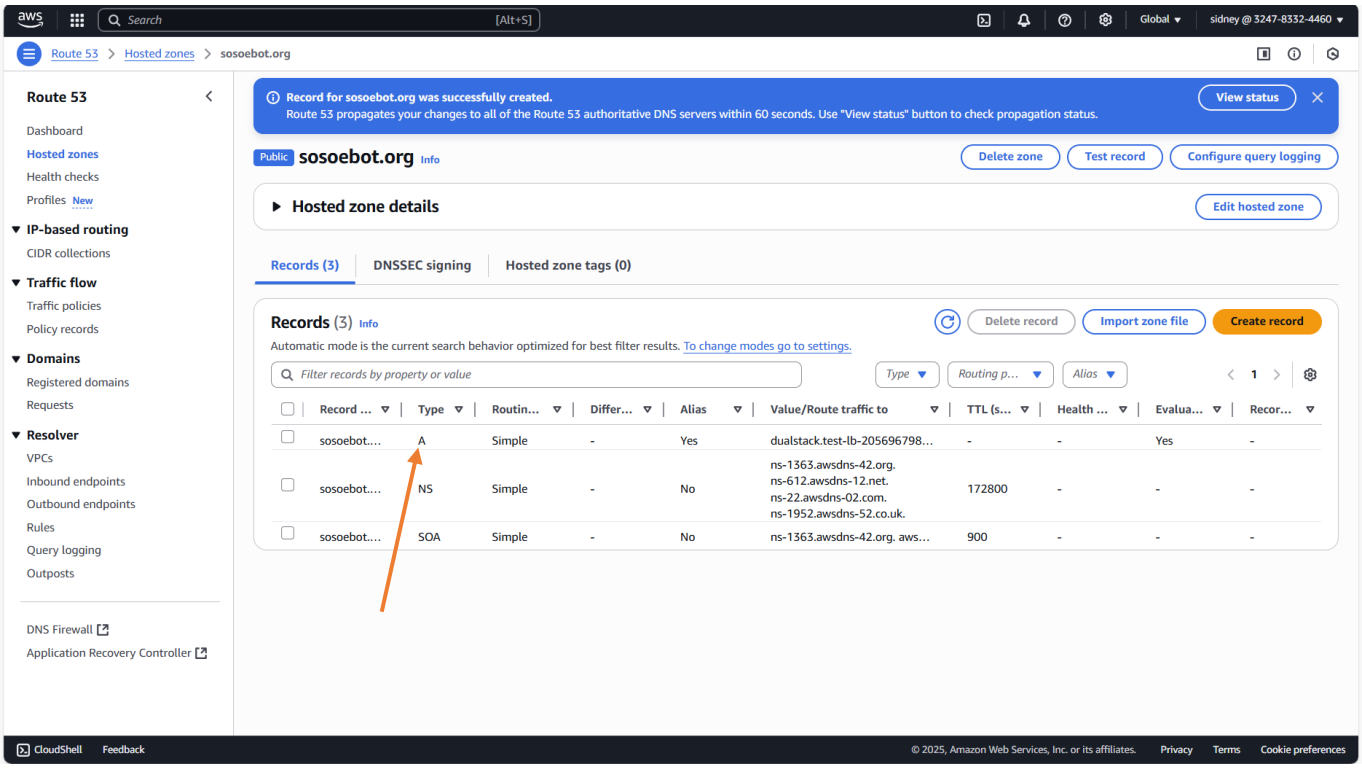
Click on "**Define simple record**"



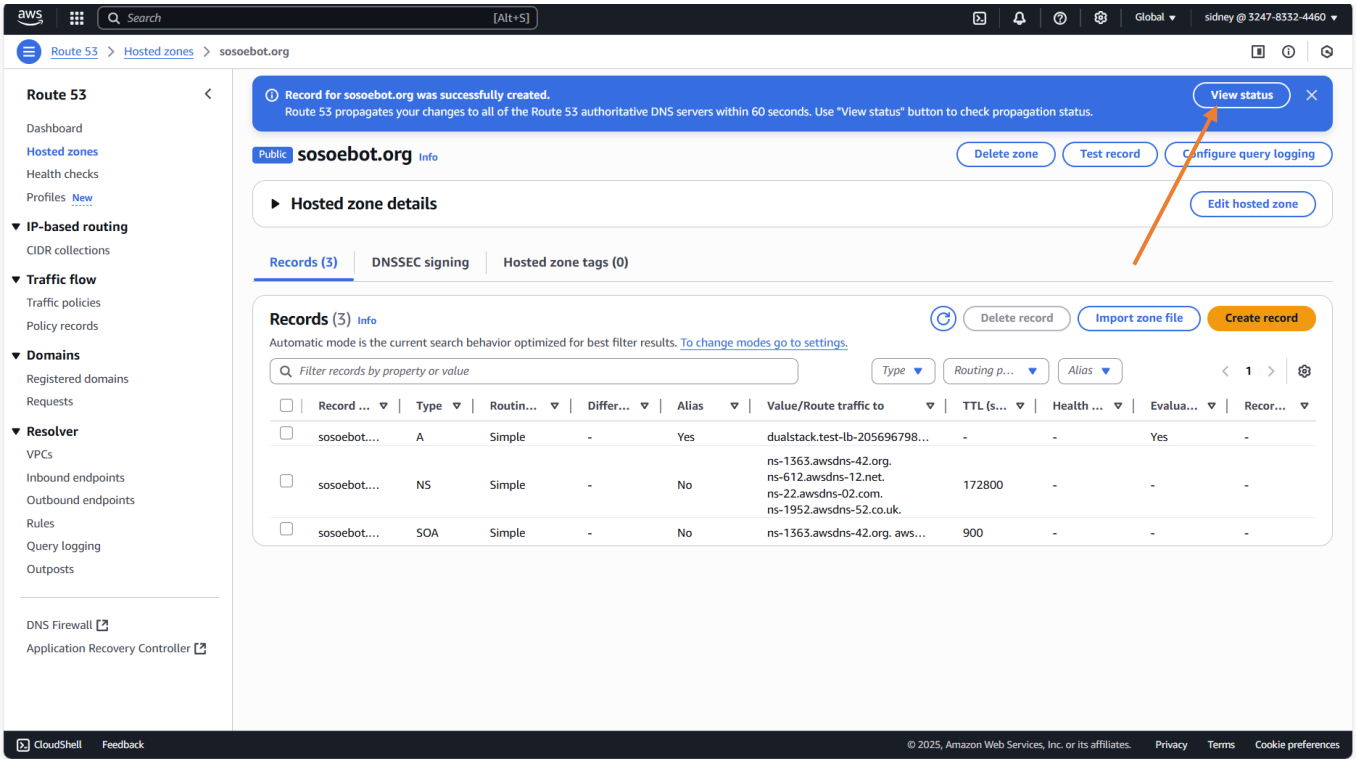
Select the Record name



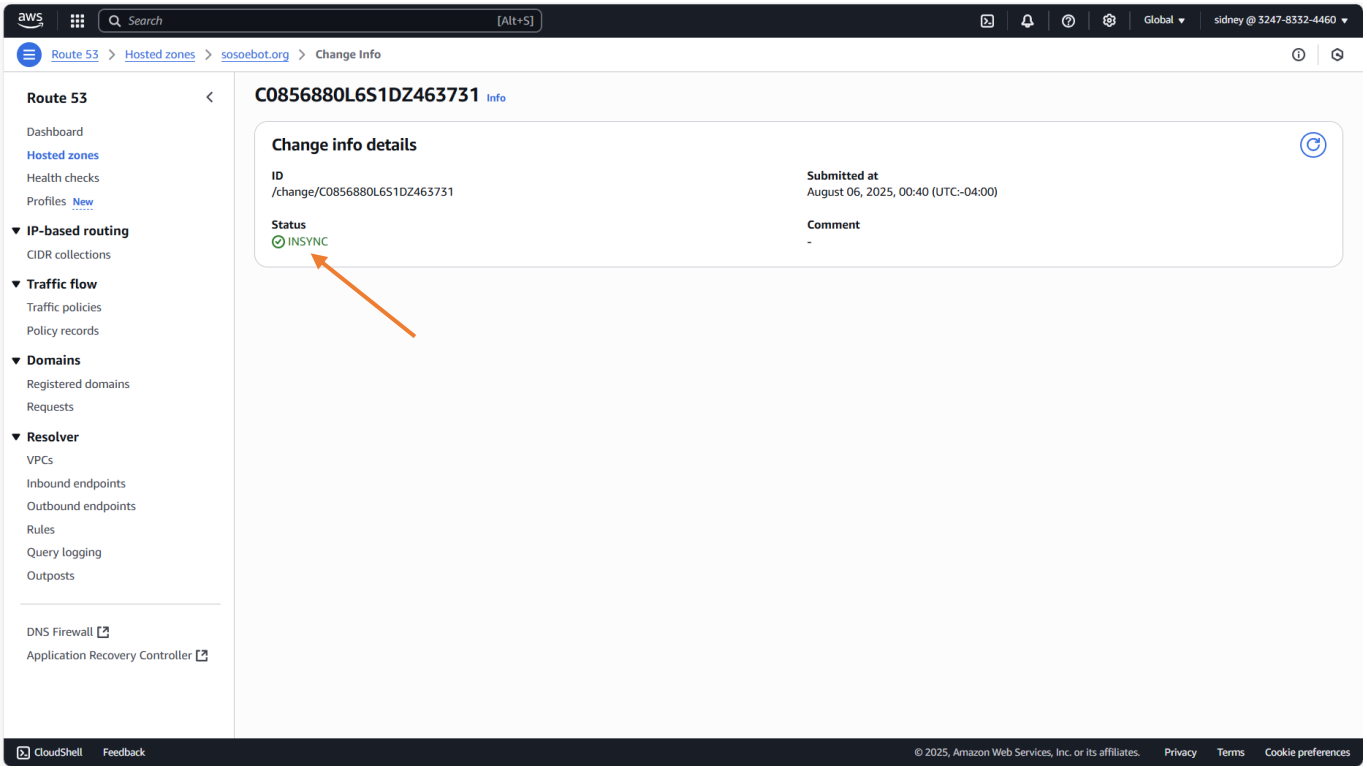
click on “**Create Record**”



You can see that we have one more record that is pointing to our application load balancer.

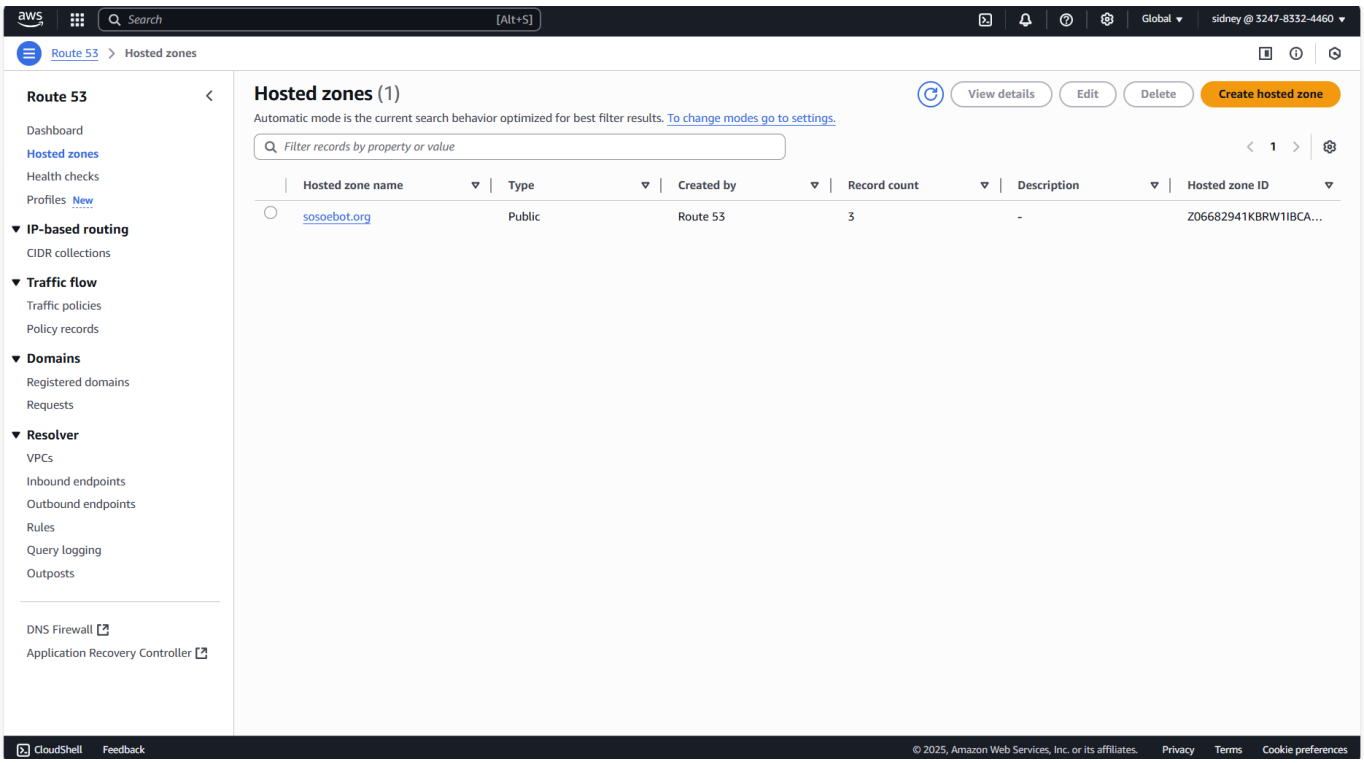


Click on “**check status**”

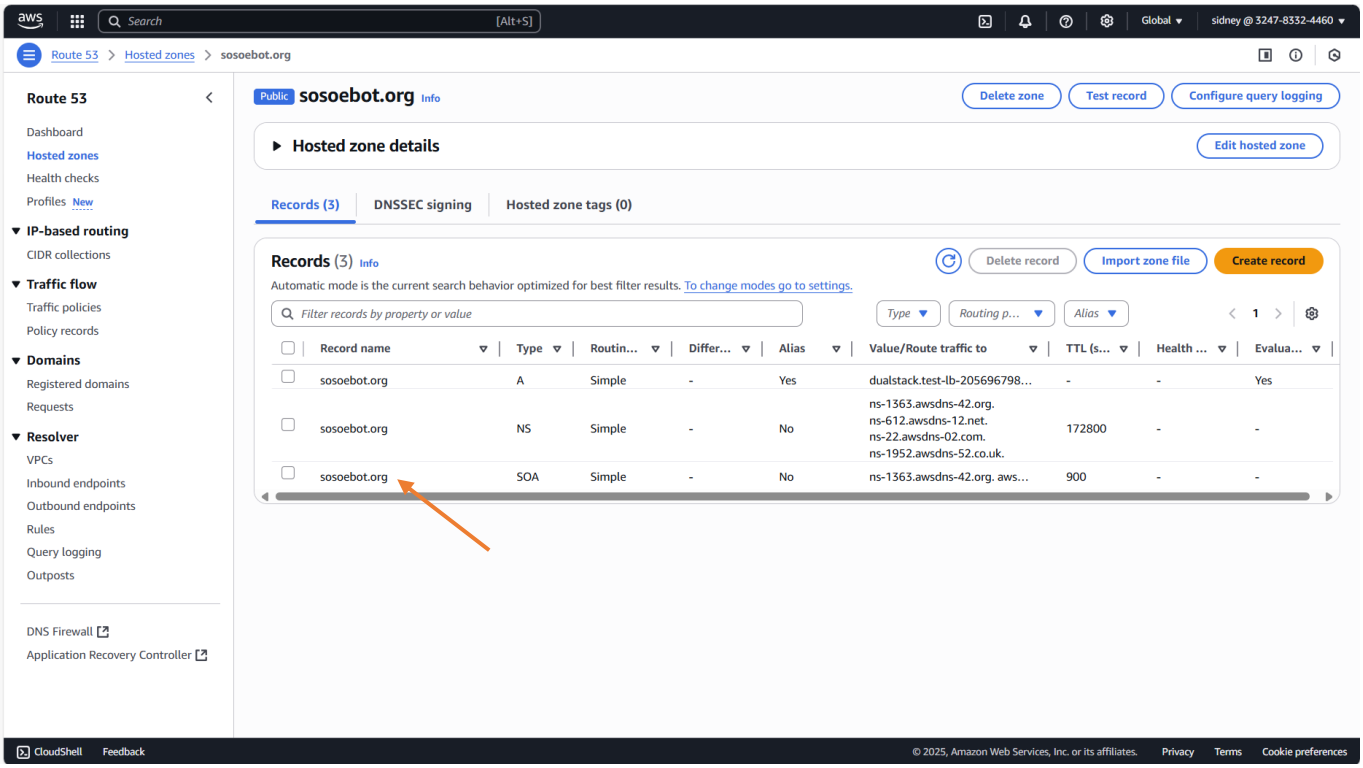


You can see that the status is “**INSYNC**”.

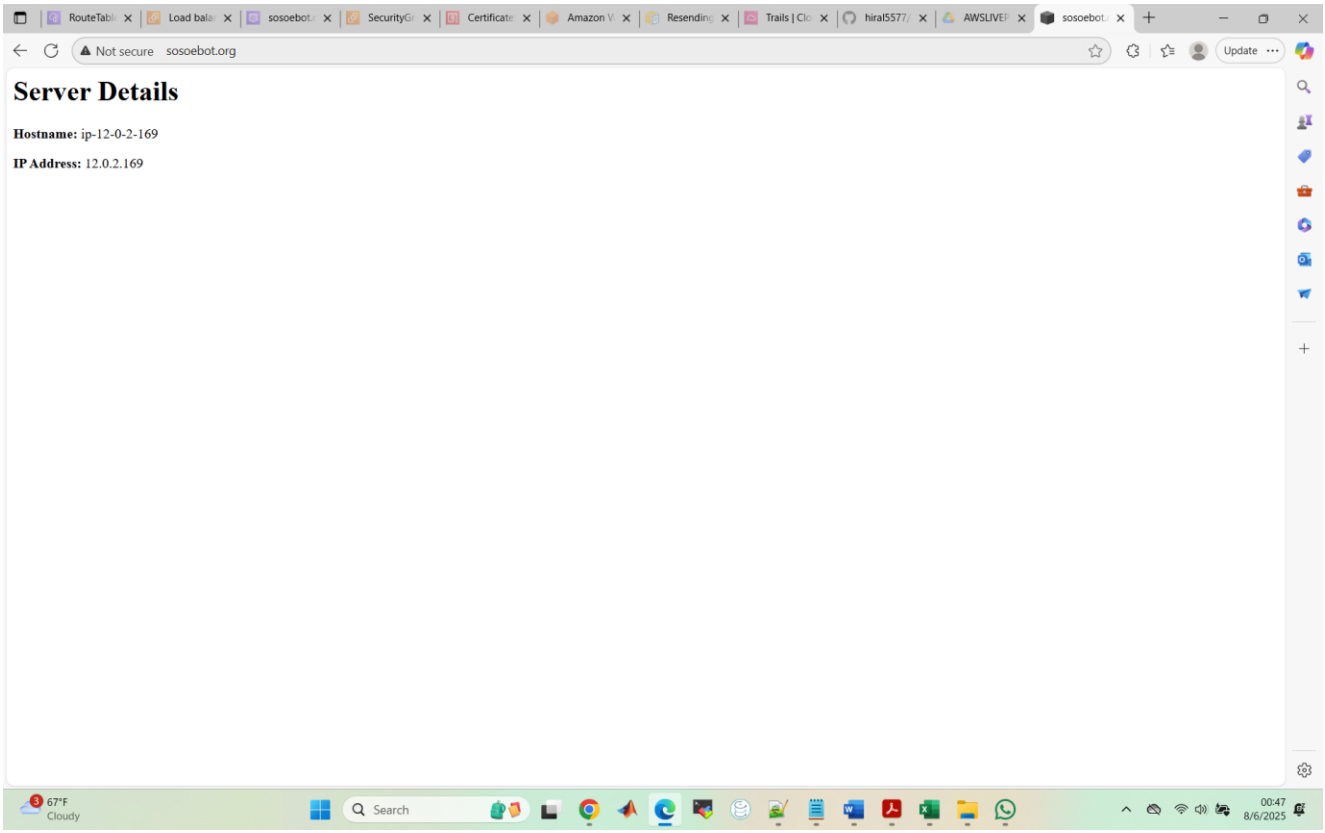
Let us go back to our hosted zone by clicking on “**Hosted Zones**”



Click on the “**Hosted Zone Name**” that is “**sosoebot.org**”



Copy the record name (**sosoebot.org**) and test on your browser



You can see that I am able to access the home page of my EC2 instance using my domain name "**sosoebot.org**"

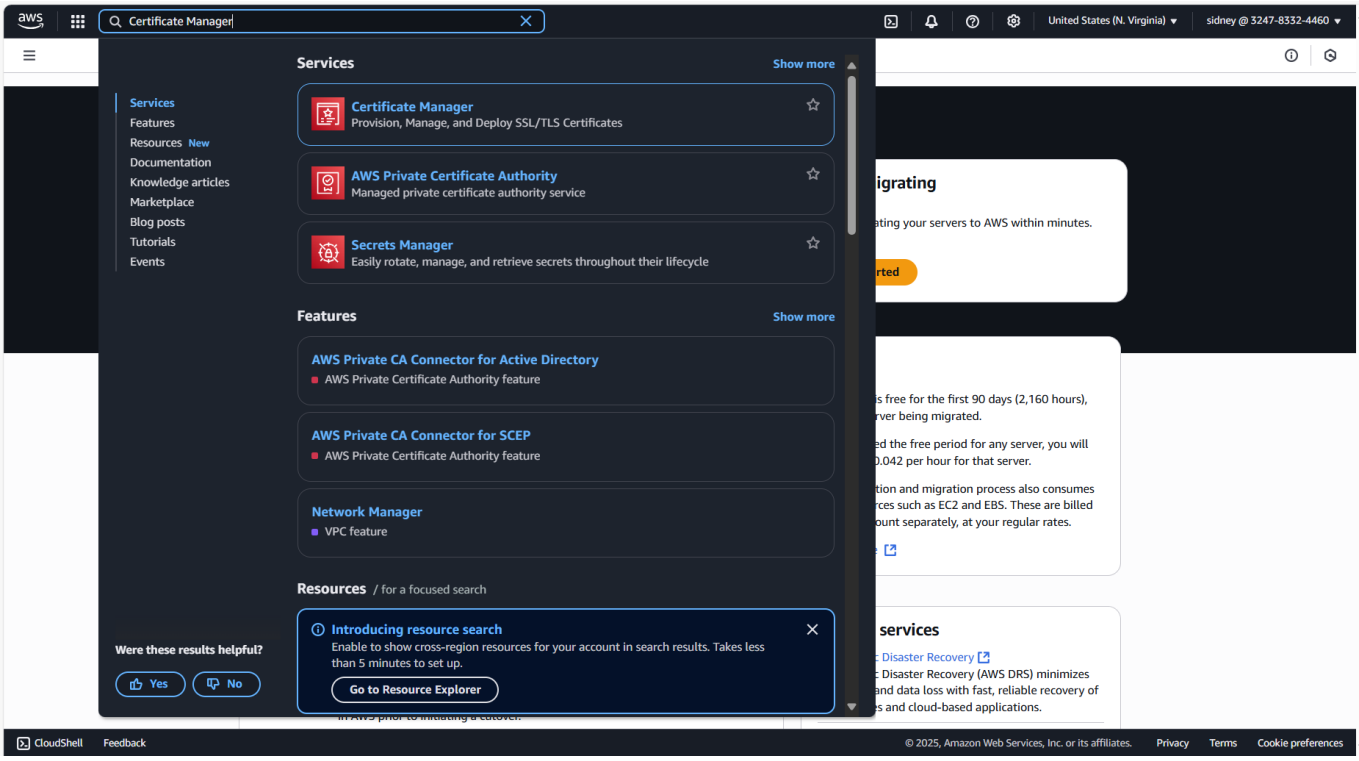
We have completed the set up and the user is able to access the URL without HTTPS. The HTTP is pointing to the Route 53 and the Route 53 is going through the Load Balancer which is accessing the EC2 instance.

STEP 8: Setting up General Certificate

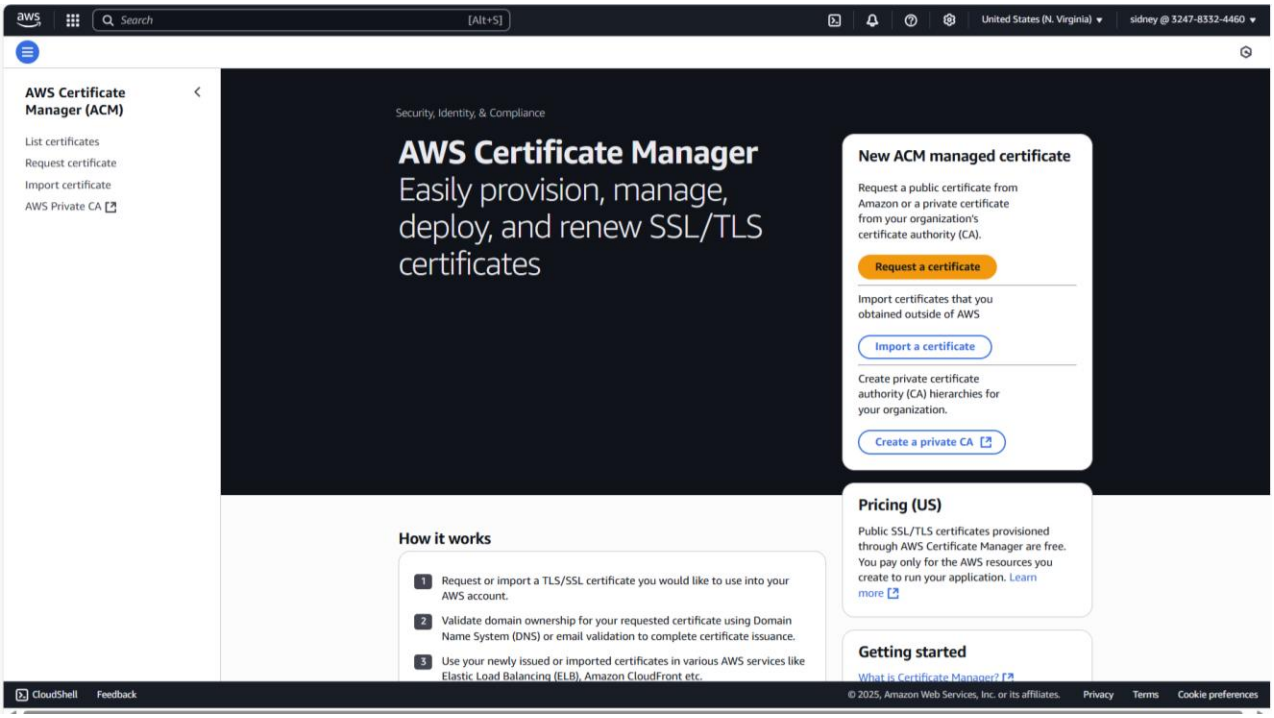
The next thing we have to set up is the HTTPS and SSL. For that we need to have our AWS Certificate Manager. We are going to use the AWS Certificate Manager to request the certificate and once we get those certificates, we are going to create those records inside our AWS Route 53. Once we import or create those records inside our Route 53, then we will be able to use the HTTPS URL along with our domain.

Enabling HTTPS with AWS Certificate Manager (ACM), this ensures that secure connections are established between your visitors and your website.

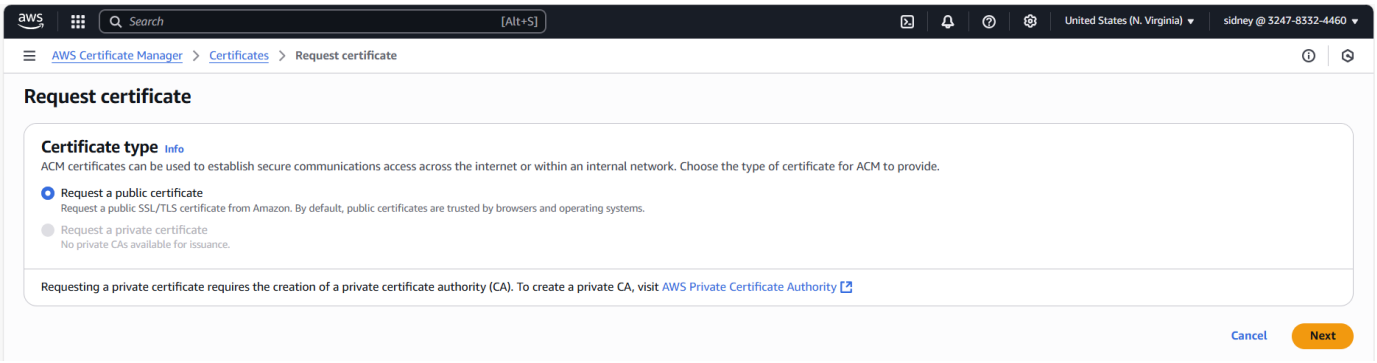
Search for "**Certificate Manager**" on AWS Management Console



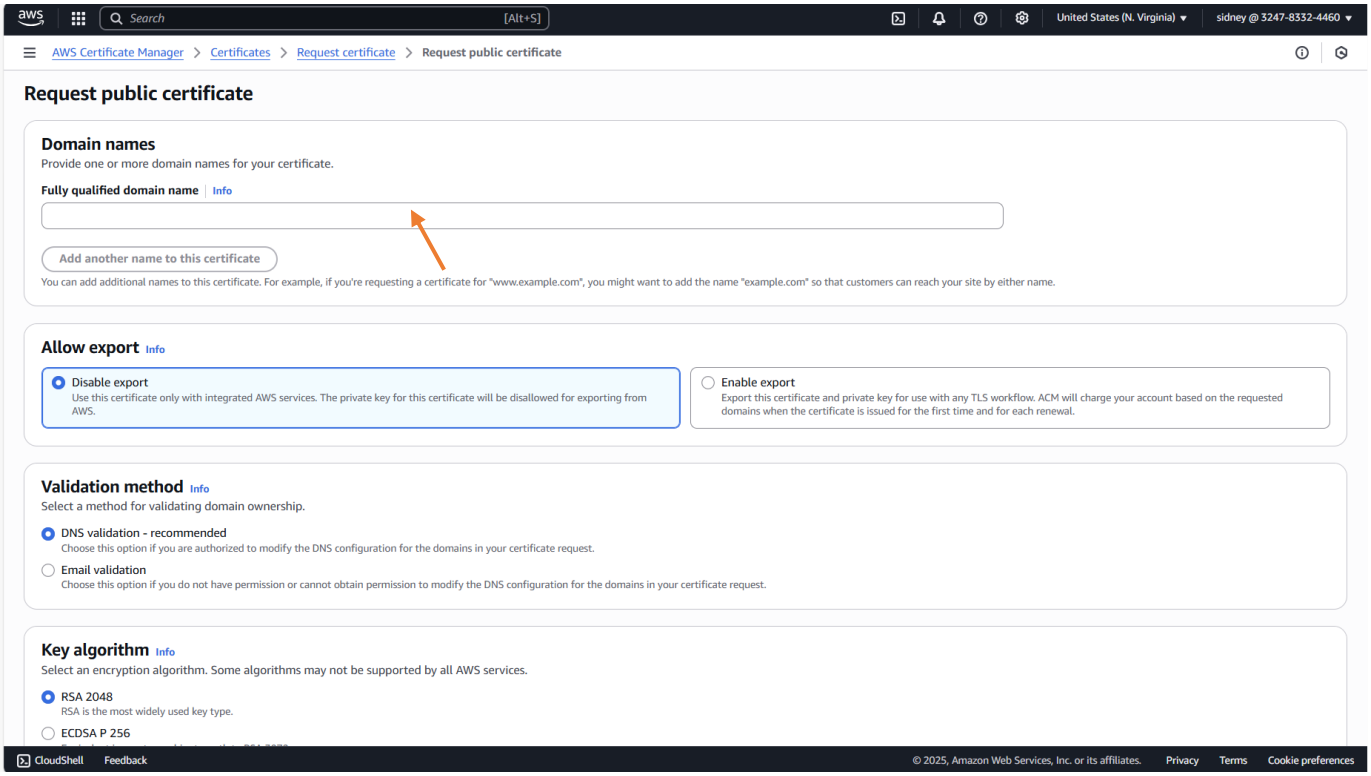
Click on "**Certificate Manager**"



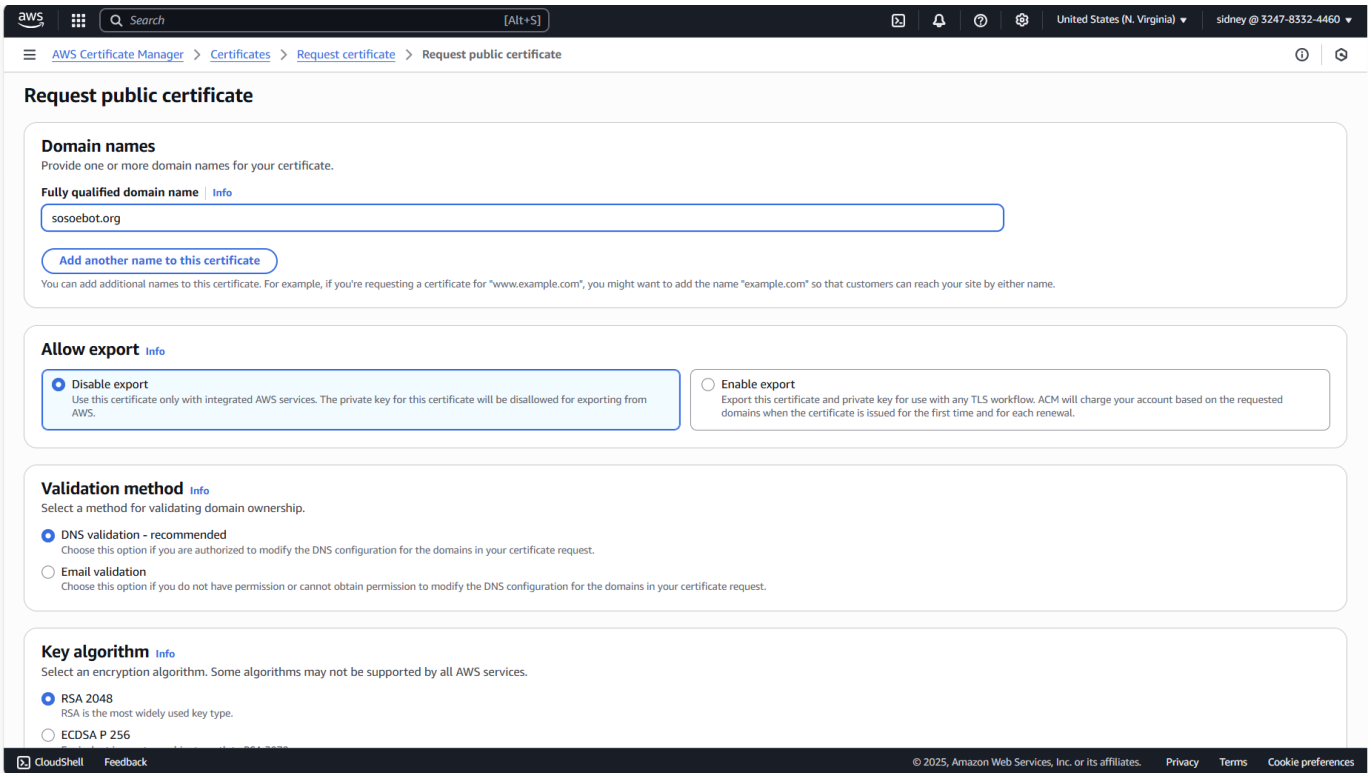
Click on “**Request a Certificate**”



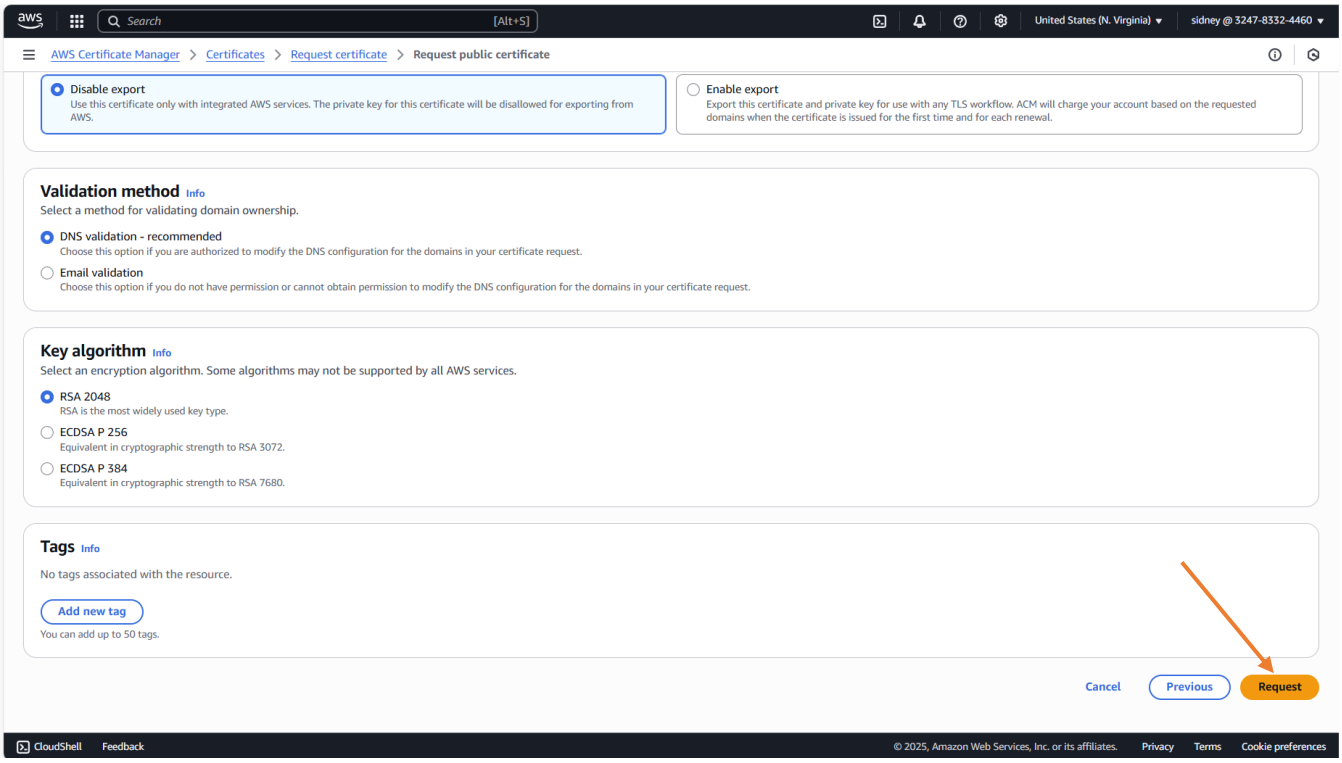
Click on “**Next**”



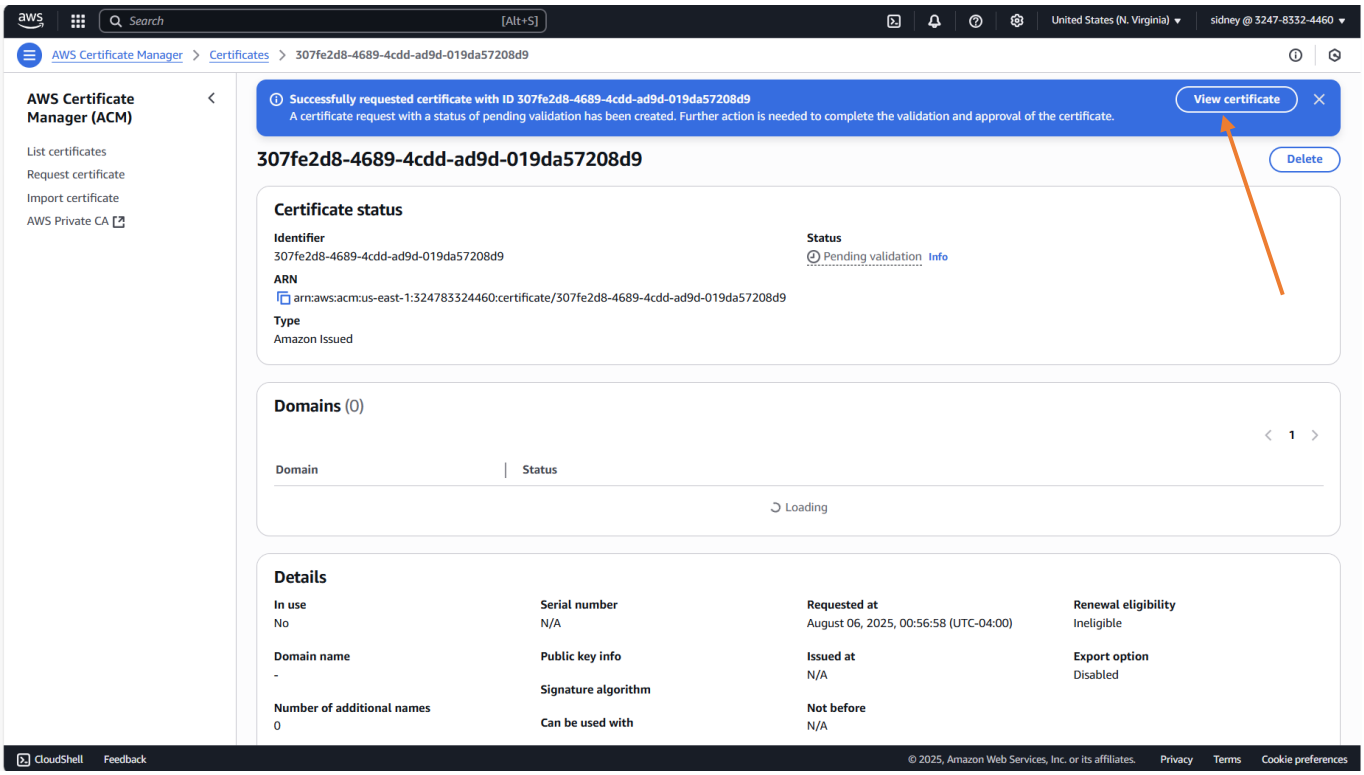
Enter your domain name here. My domain name is "**sosoebot.org**"



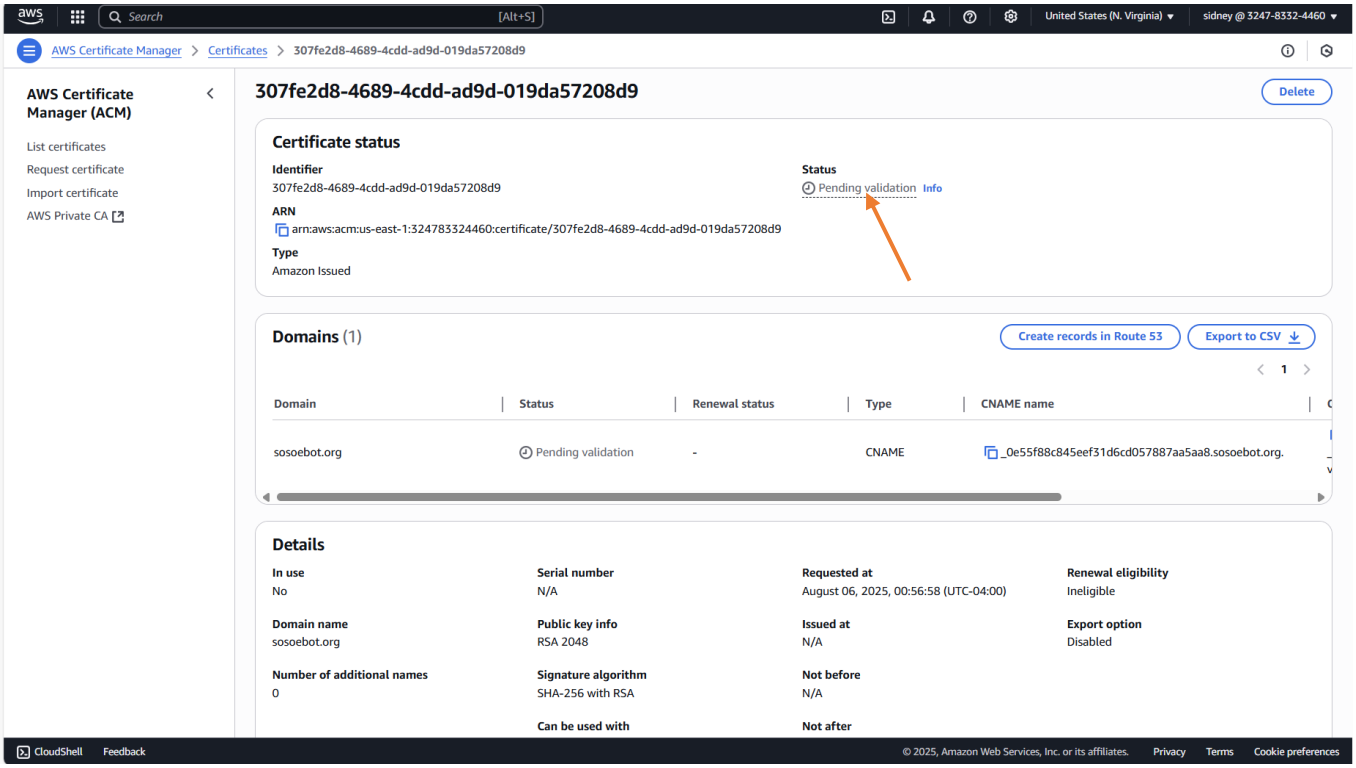
Scroll down



Click on "**Request**"

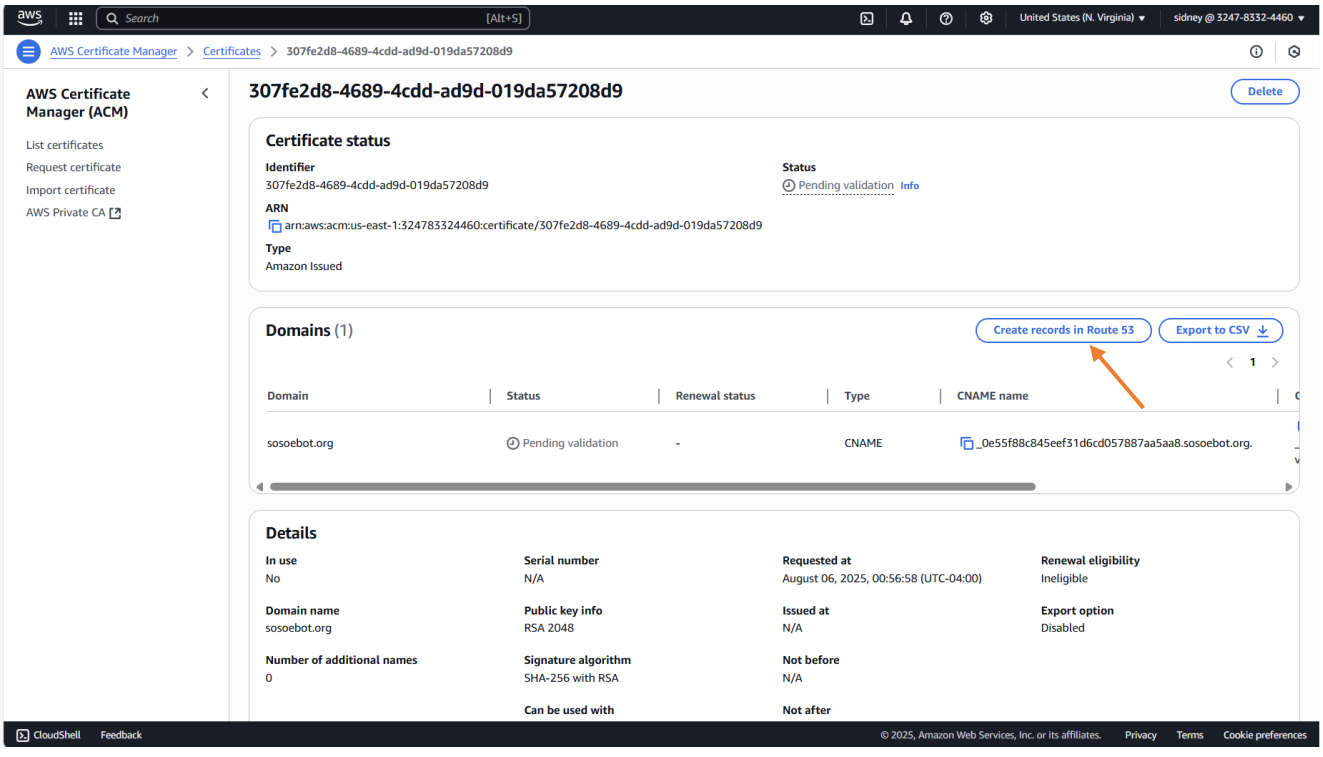


Click on "**View Certificate**"

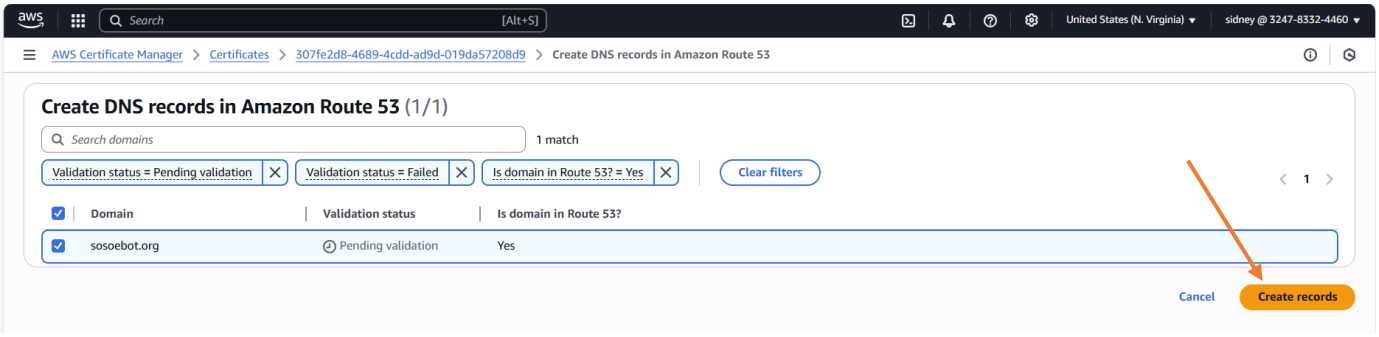


Our certificate has been created but the status is "**Pending**"

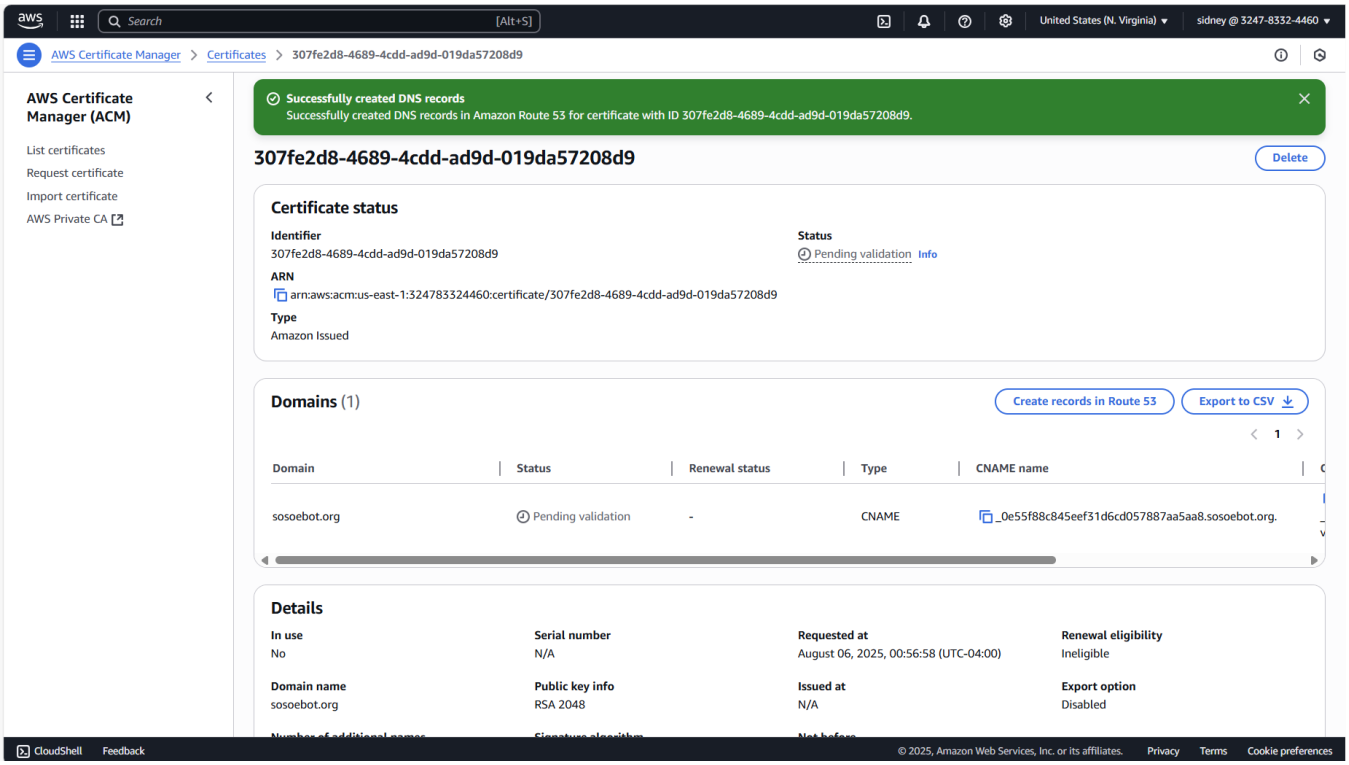
Now we have to create records on Route 53



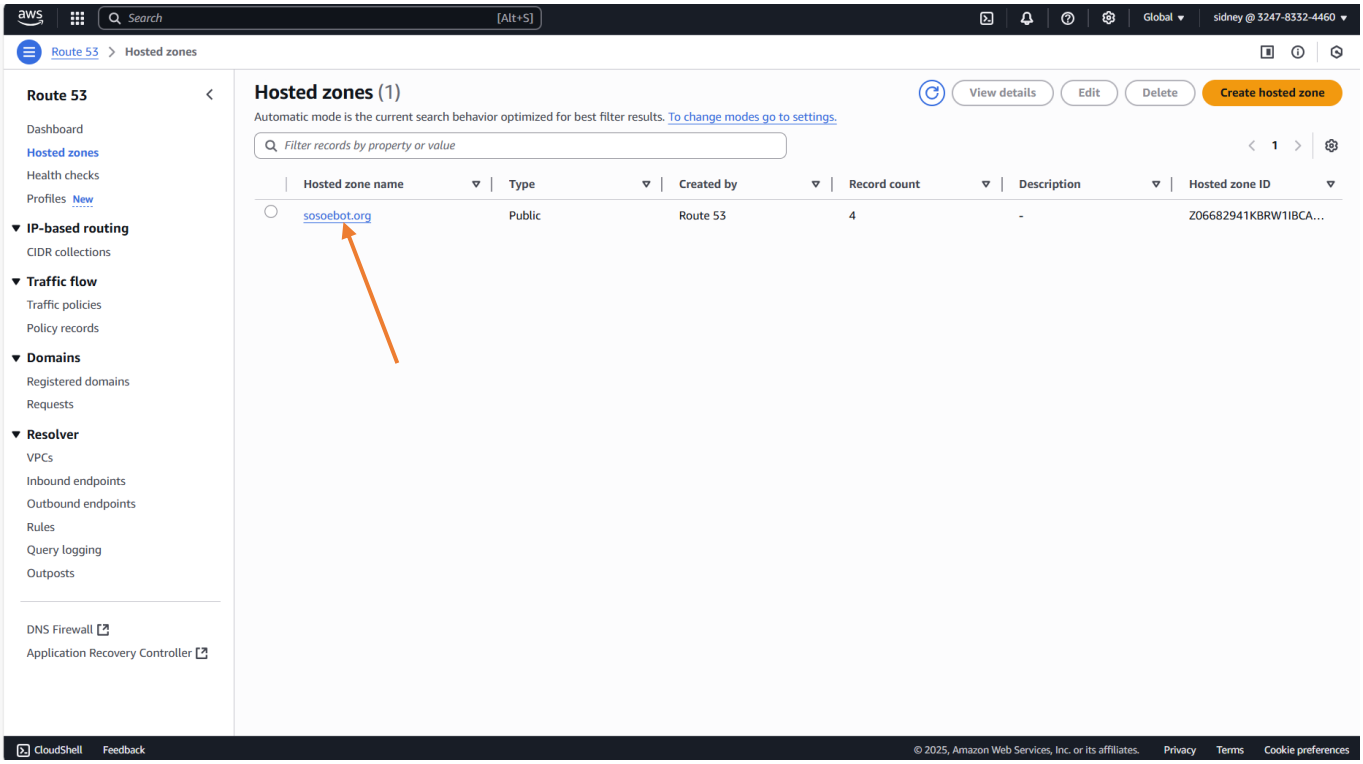
Now, click on "**Create Records in Route 53**". This is automatically going to create some records in AWS Route 53.



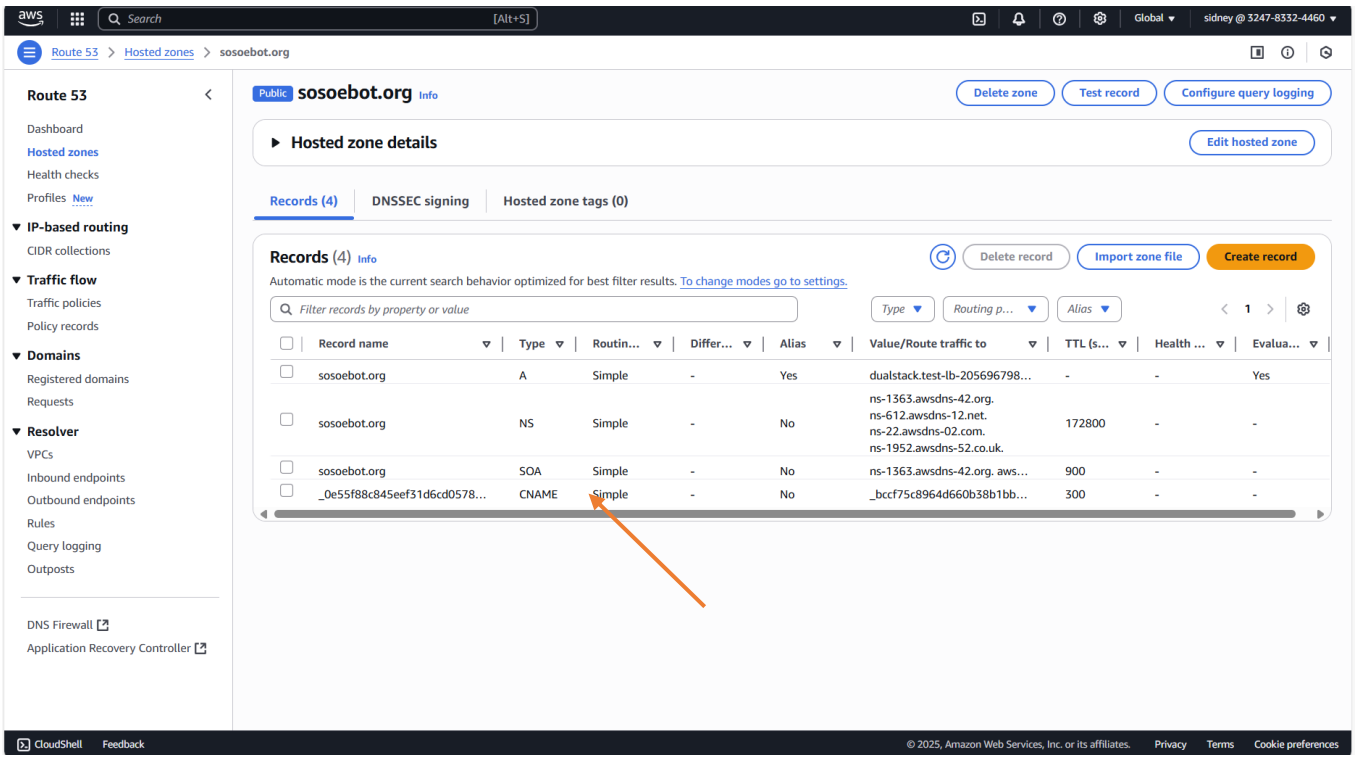
Click on "**Create Records**"



Head back to Route 53 and click on "**Hosted Zones**"

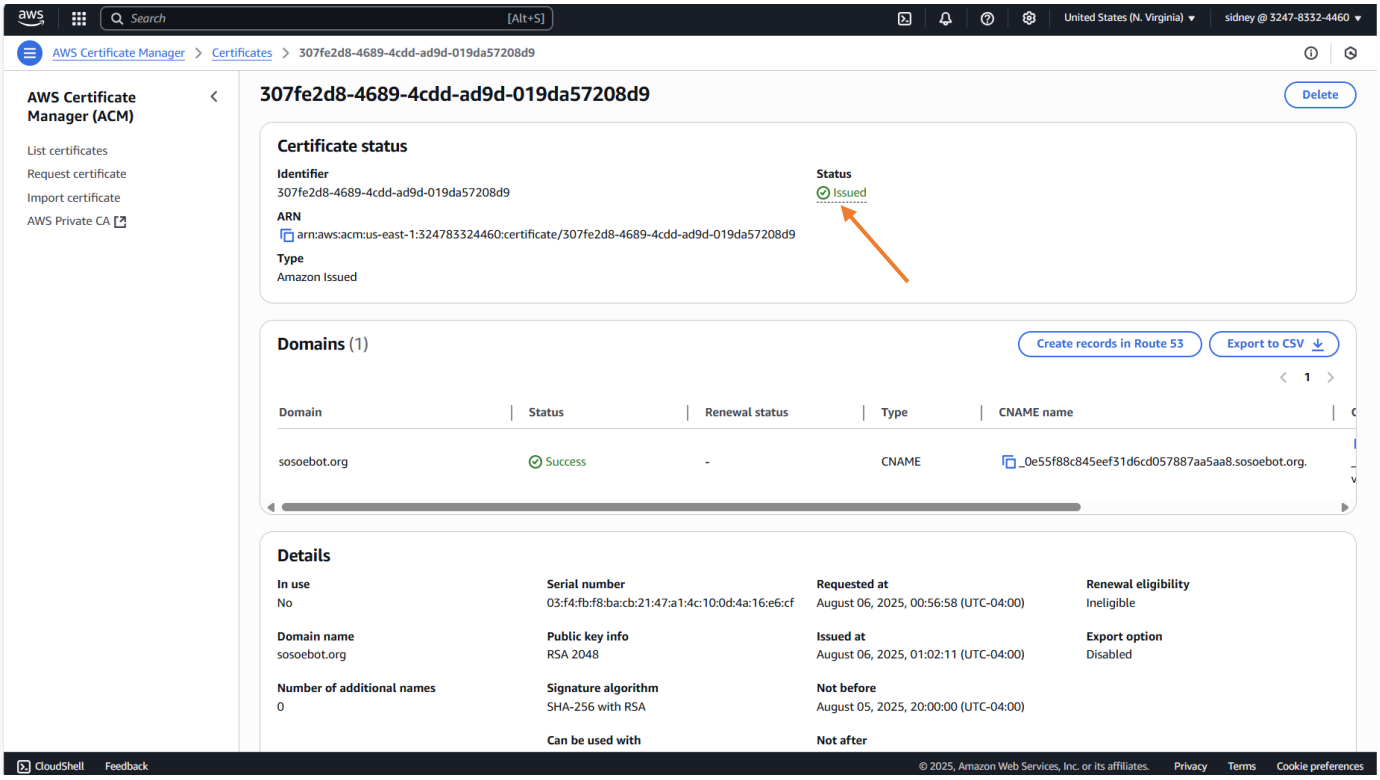


Click on our Hosted Zone name



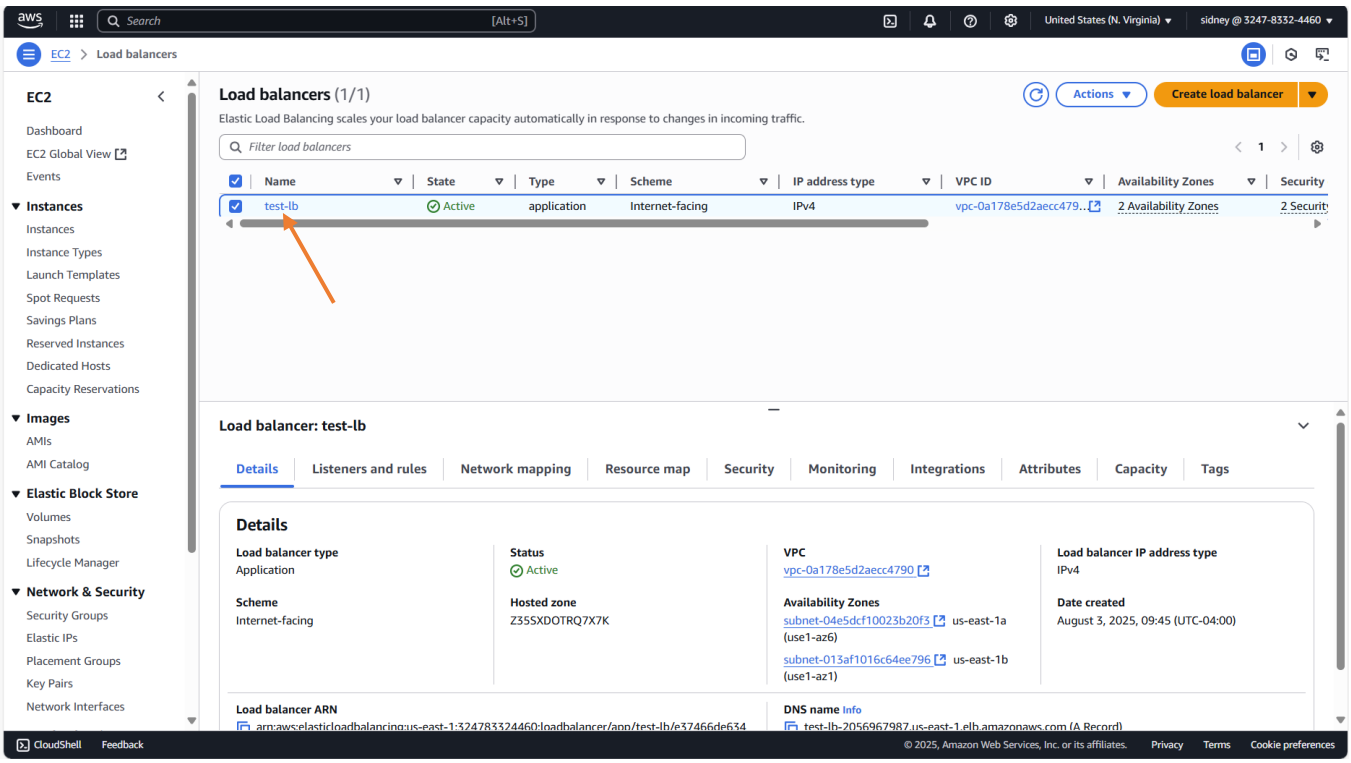
You can see that we now have an additional record of type “**CNAME**”

Now, go back and check the status of the created certificate

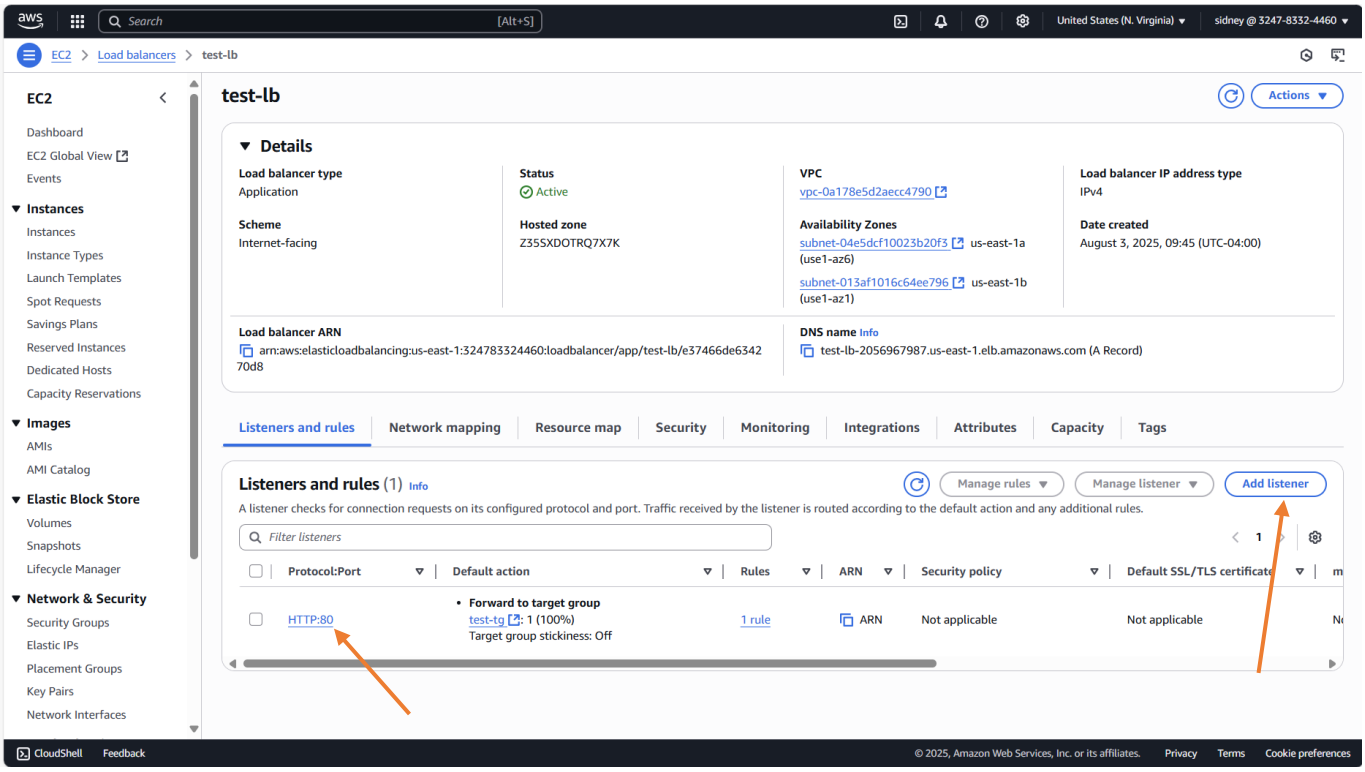


Now, the certificate has been issued. We now move to the next step,

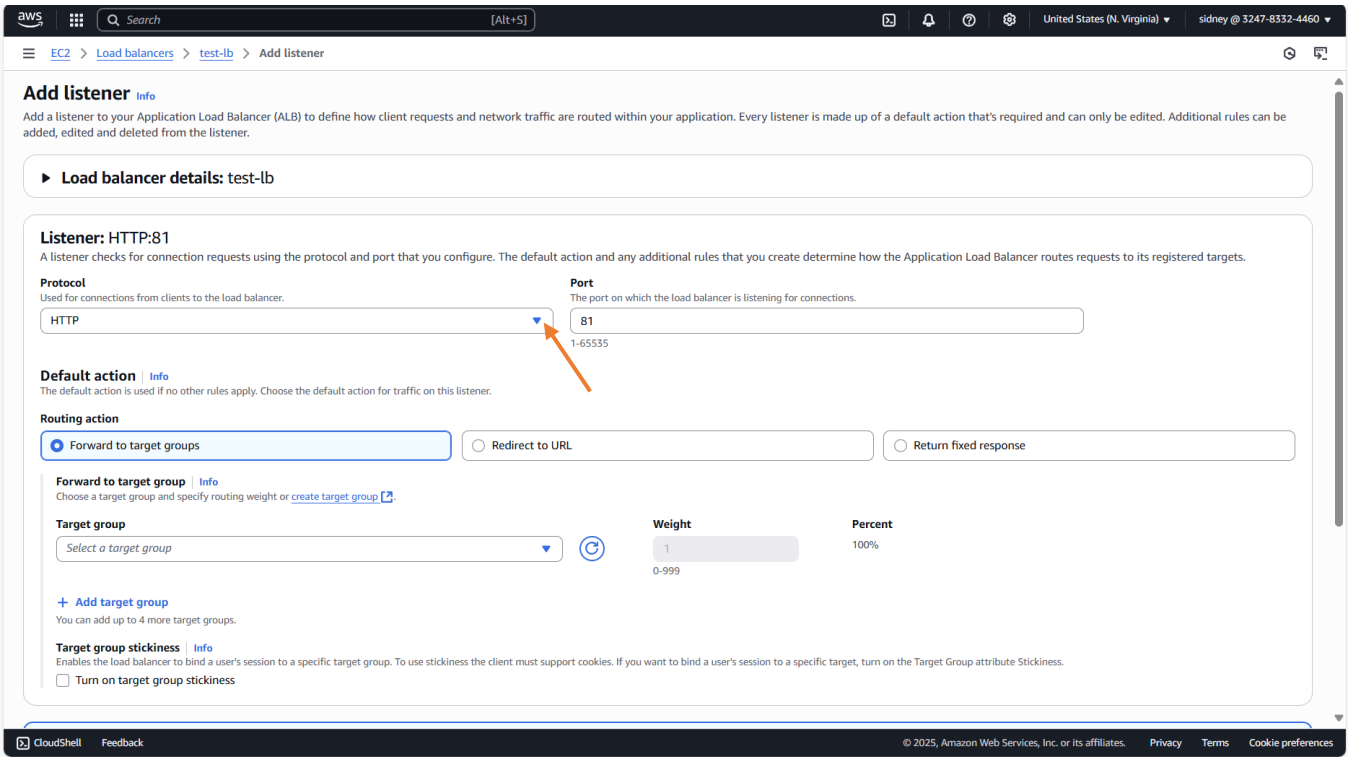
Before we access the URL with HTTPS, we still need to work on the load balancer. We have to give access to HTTPS port. Go back to the load balancer.



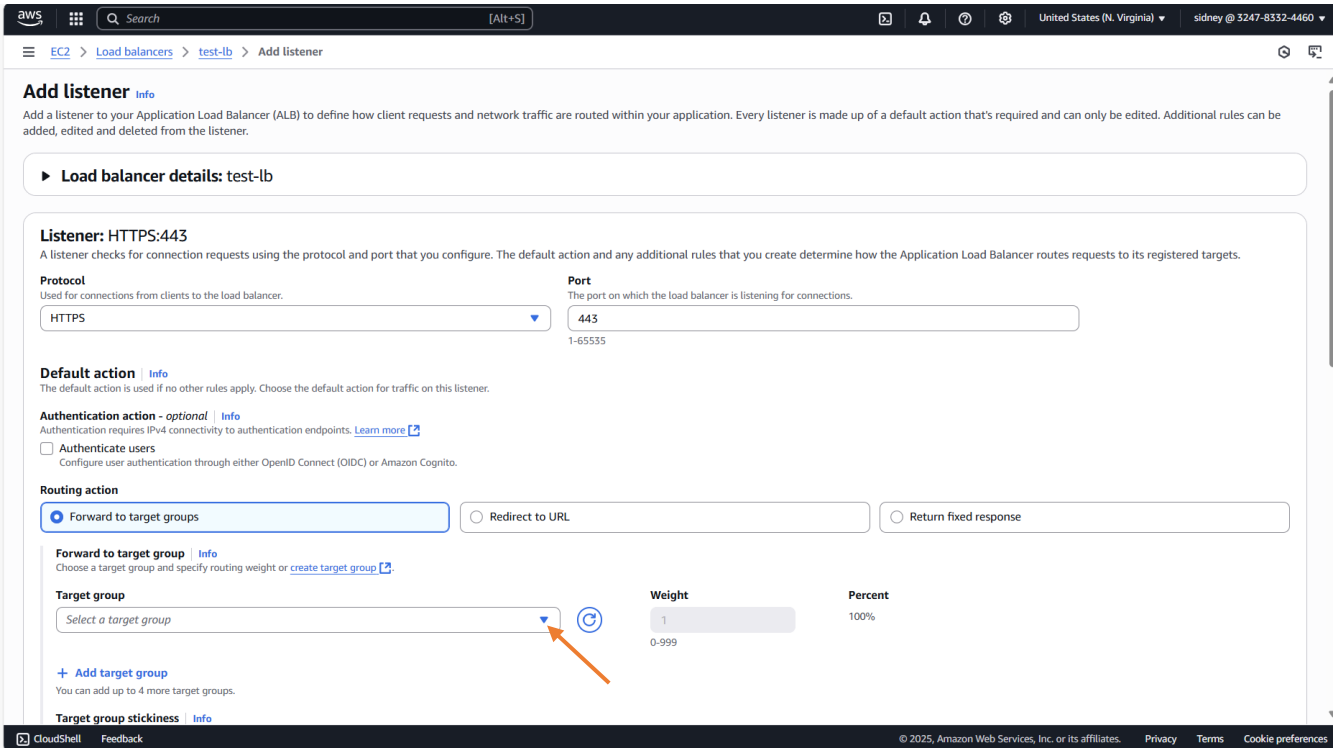
Click on the Load Balancer name



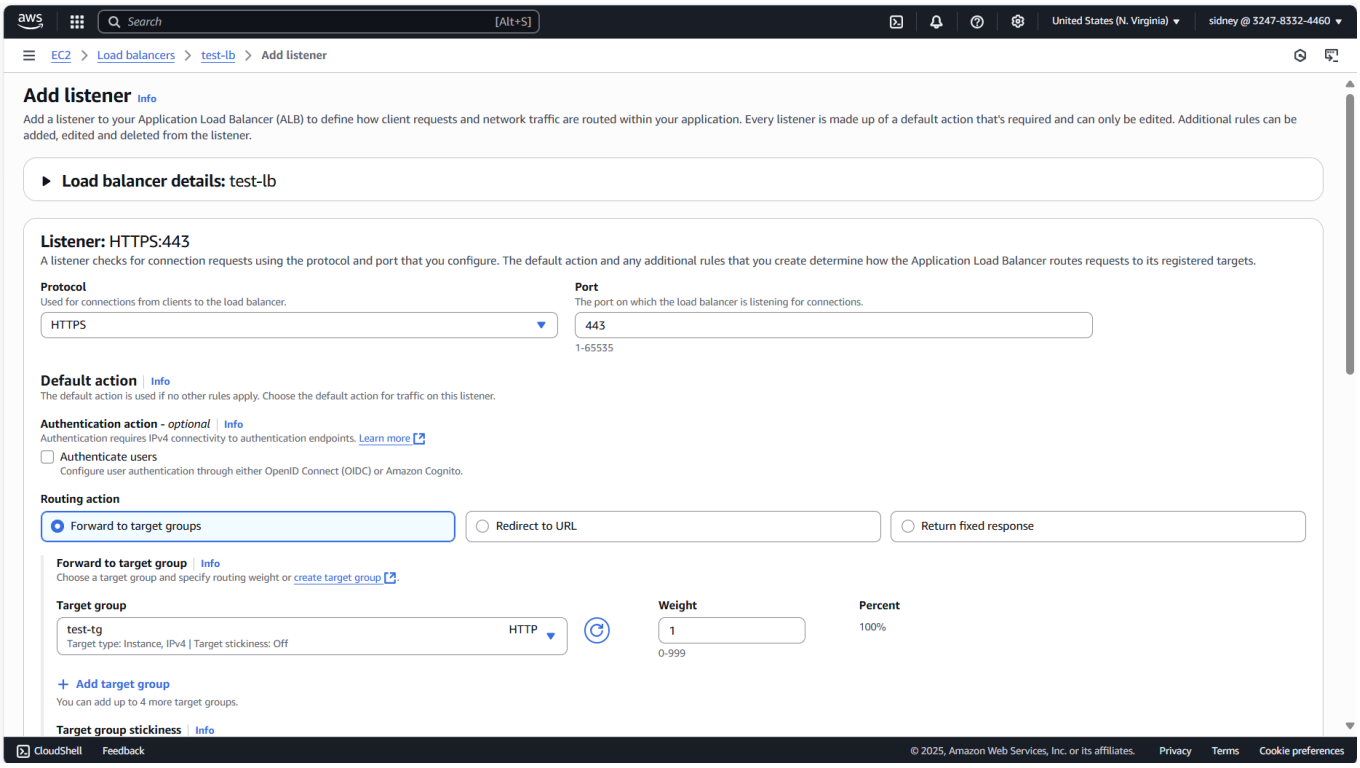
You can see that in the "**Listener and rules**", the protocol port is only for HTTP, so we have to add HTTPS. Click on "**Add Listener**"



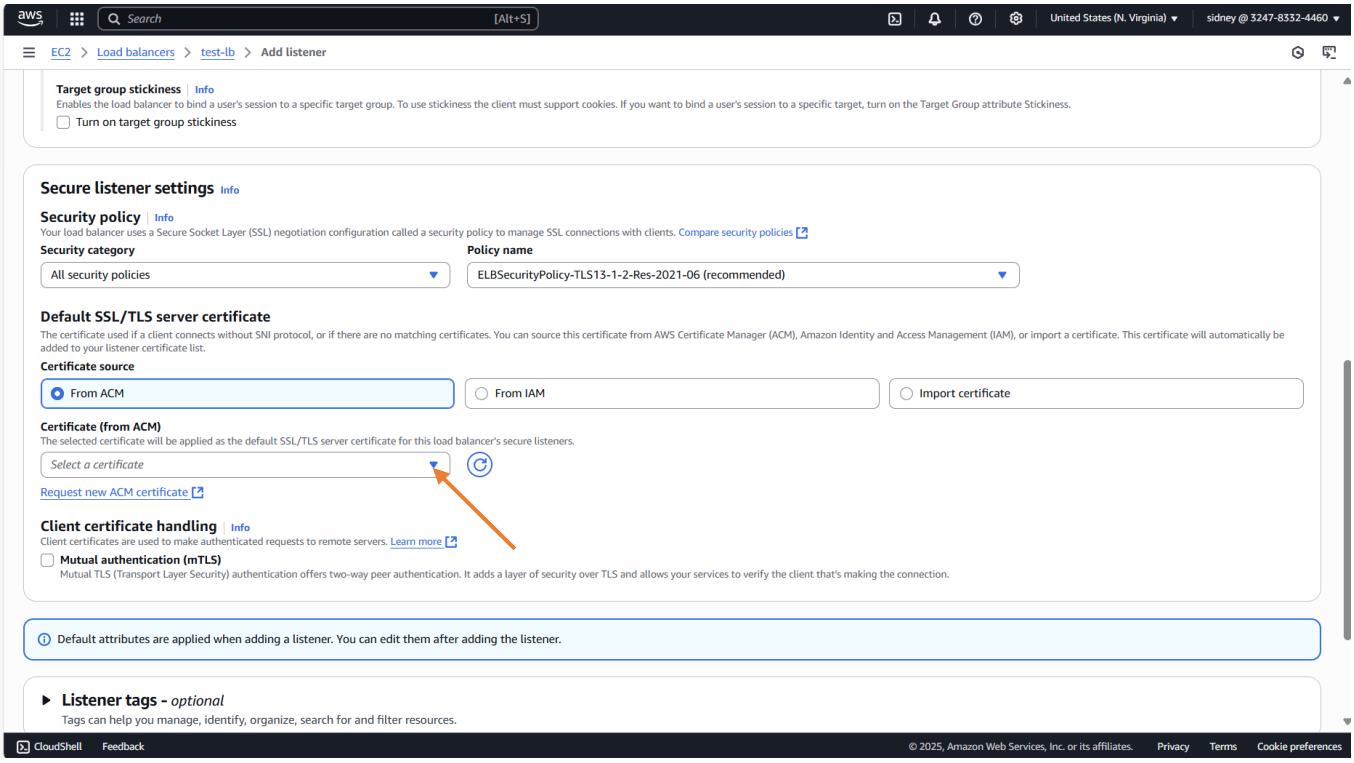
Click on the drop down and select "**HTTPS**"



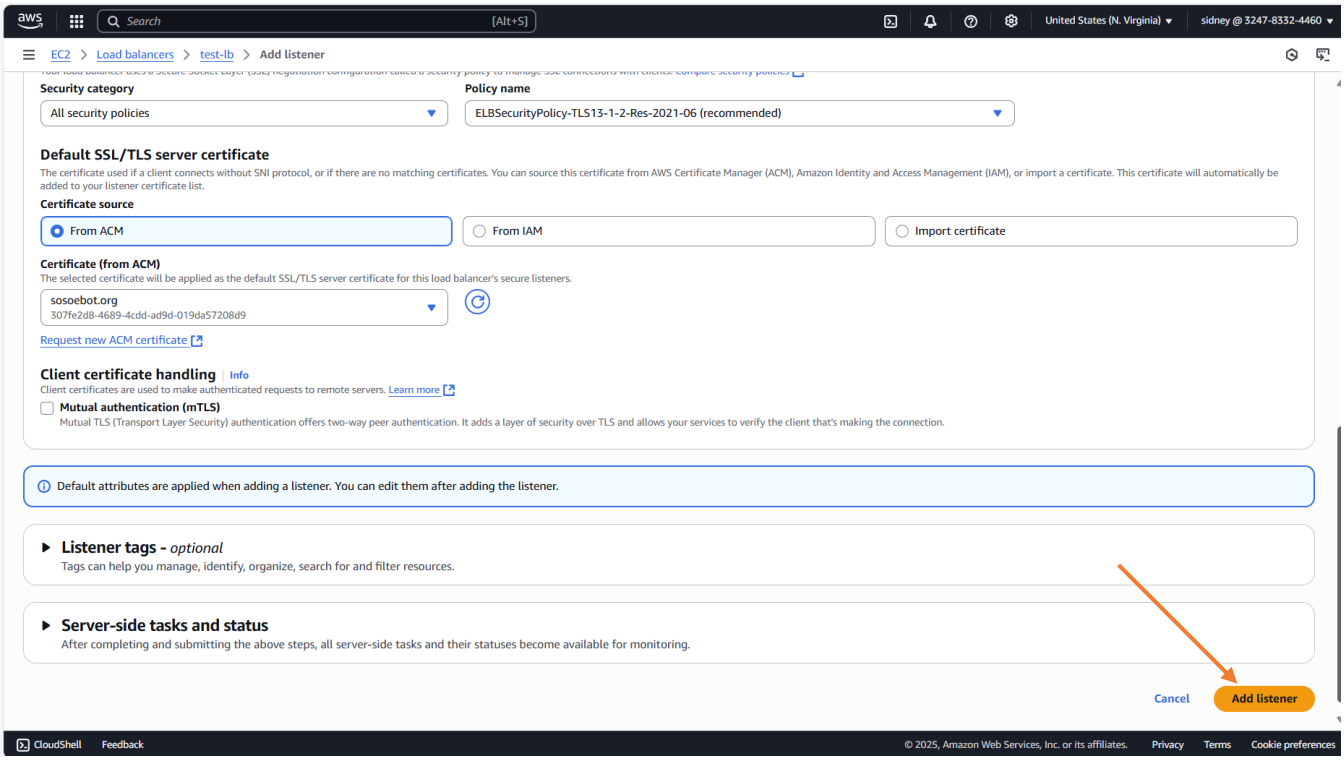
Click on the drop down on Target group and select the target group we created, that is "**test-tg**"



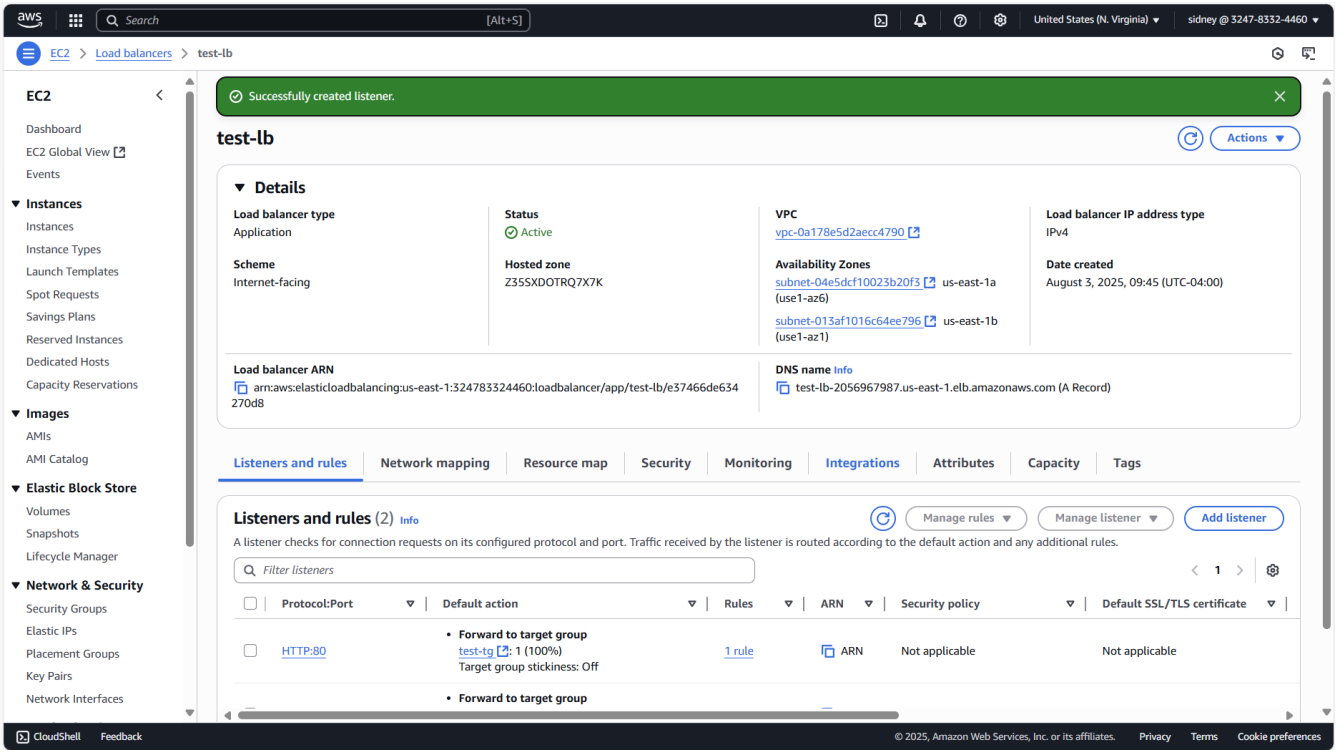
Scroll down to "**Secure listener settings**"



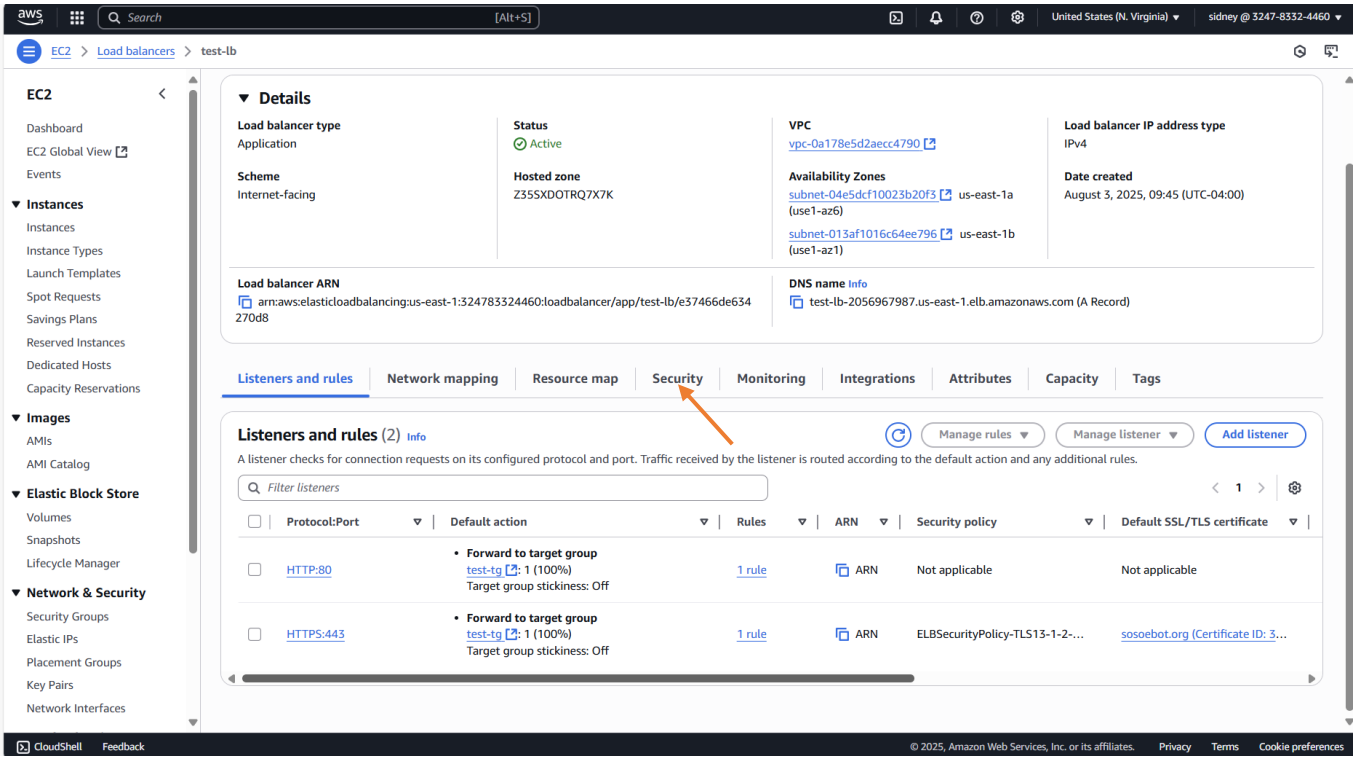
Click on the drop down and select our certificate for "**sosoebot.org**"



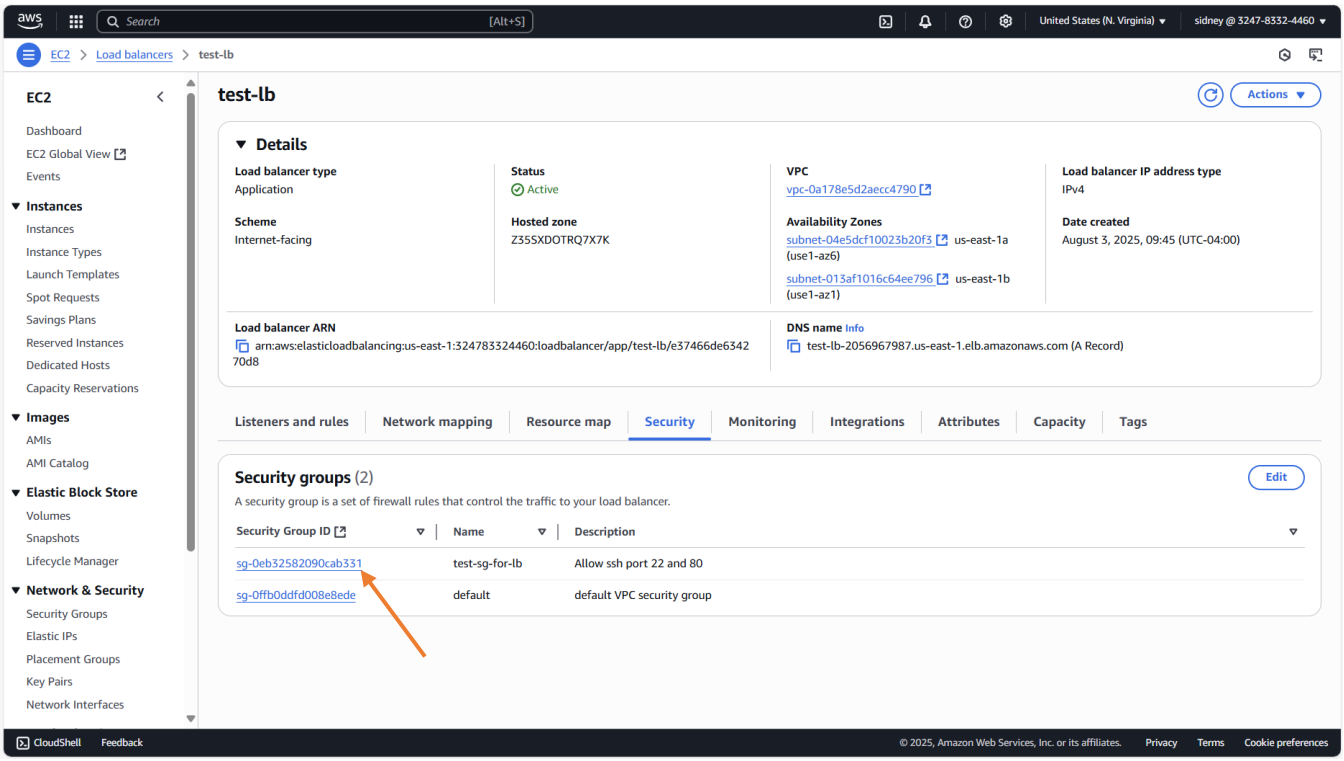
Click on "**Add Listener**"



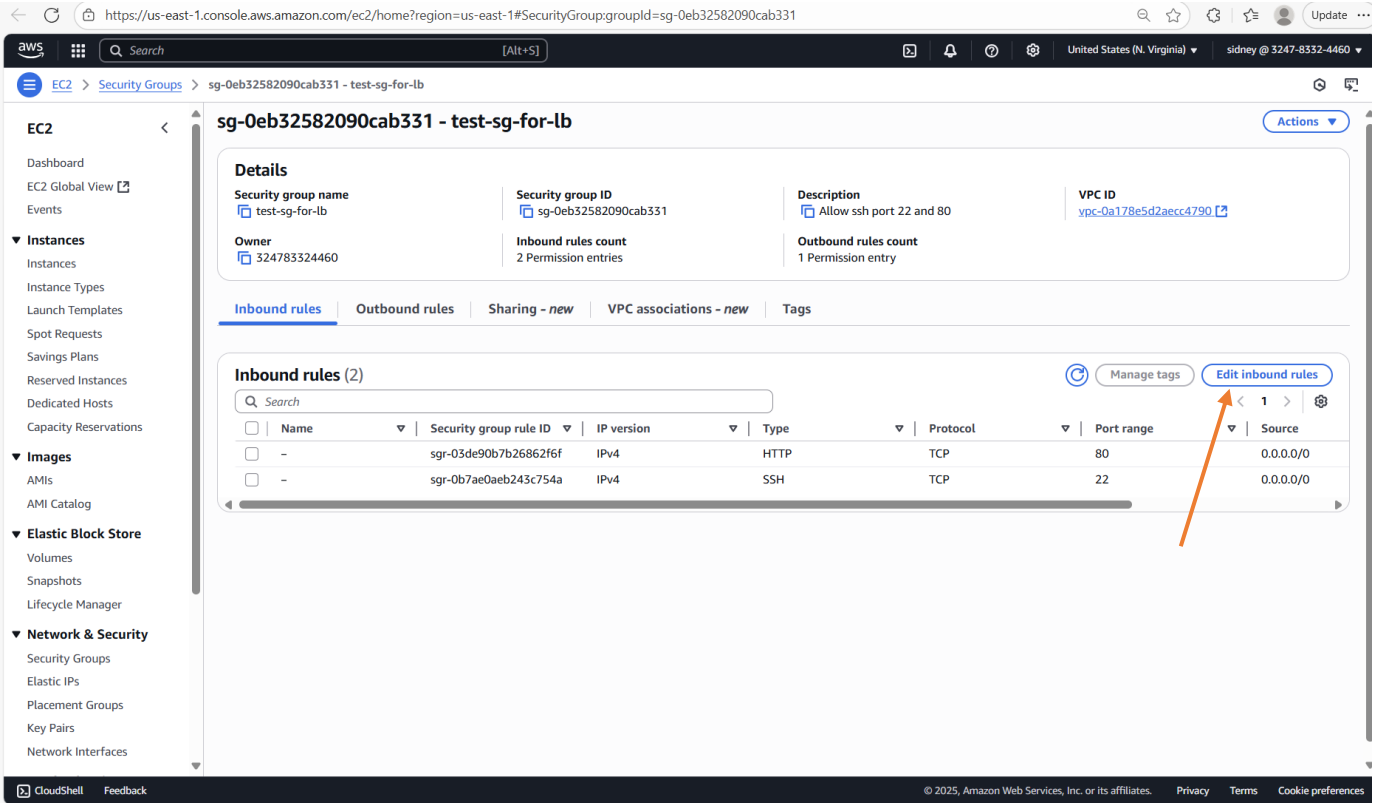
Scroll down



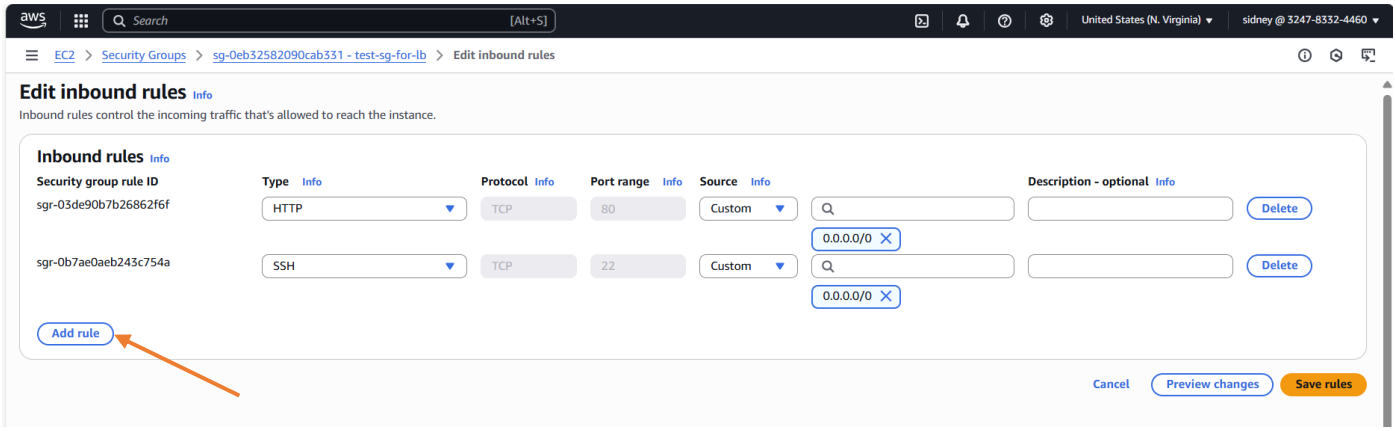
Now you can see two listeners and rules. Now, click on the “**Security Tab**”



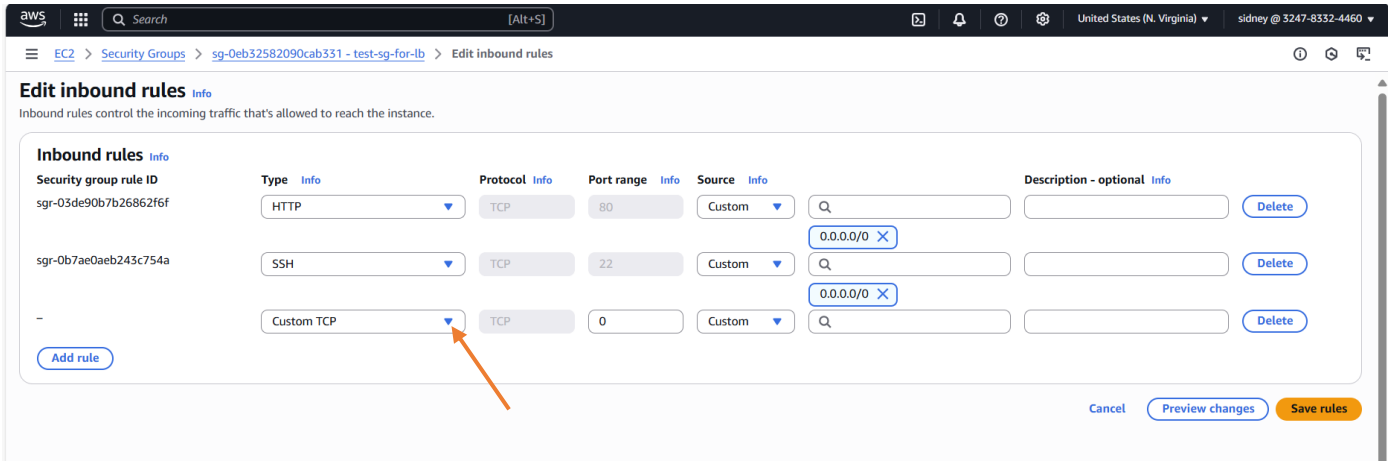
We have to add inbound rule for **HTTPS**. Click on the security group we created



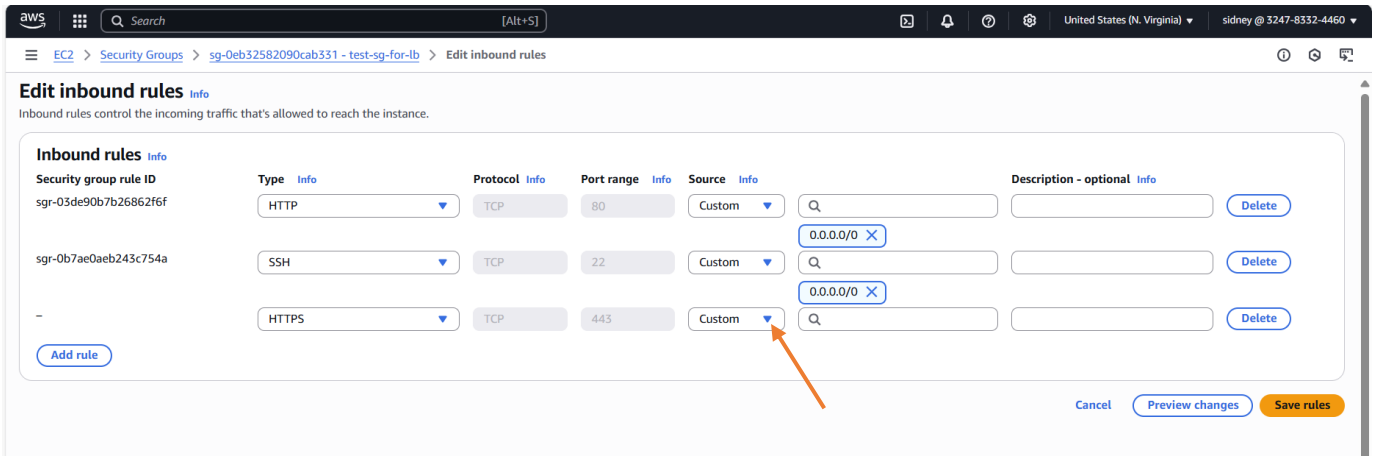
Click on "**Edit Inbound Rules**"



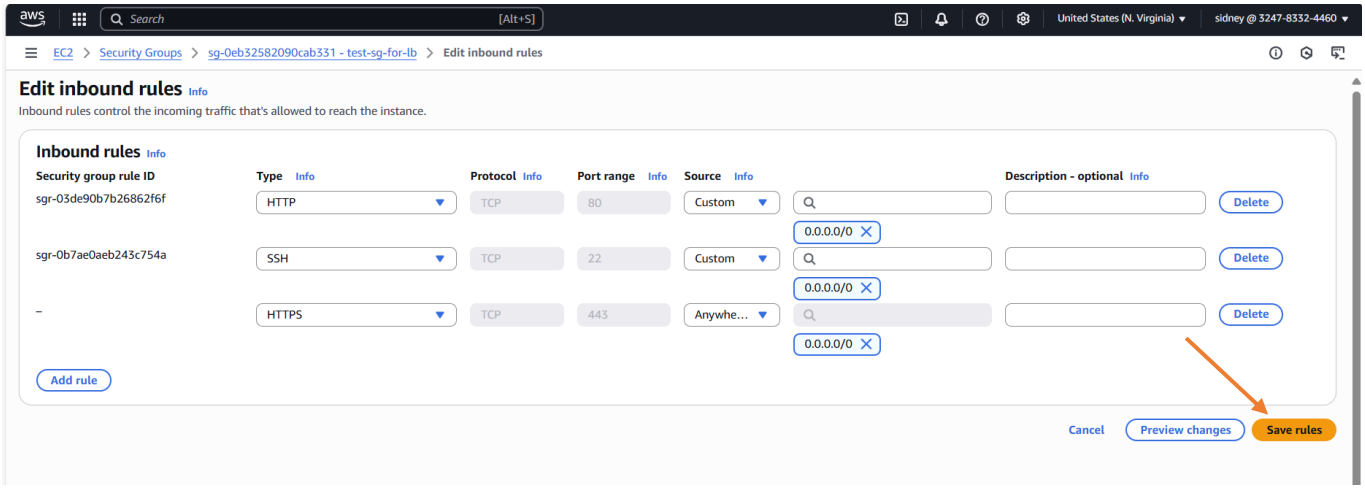
Click on “**Add Rule**”



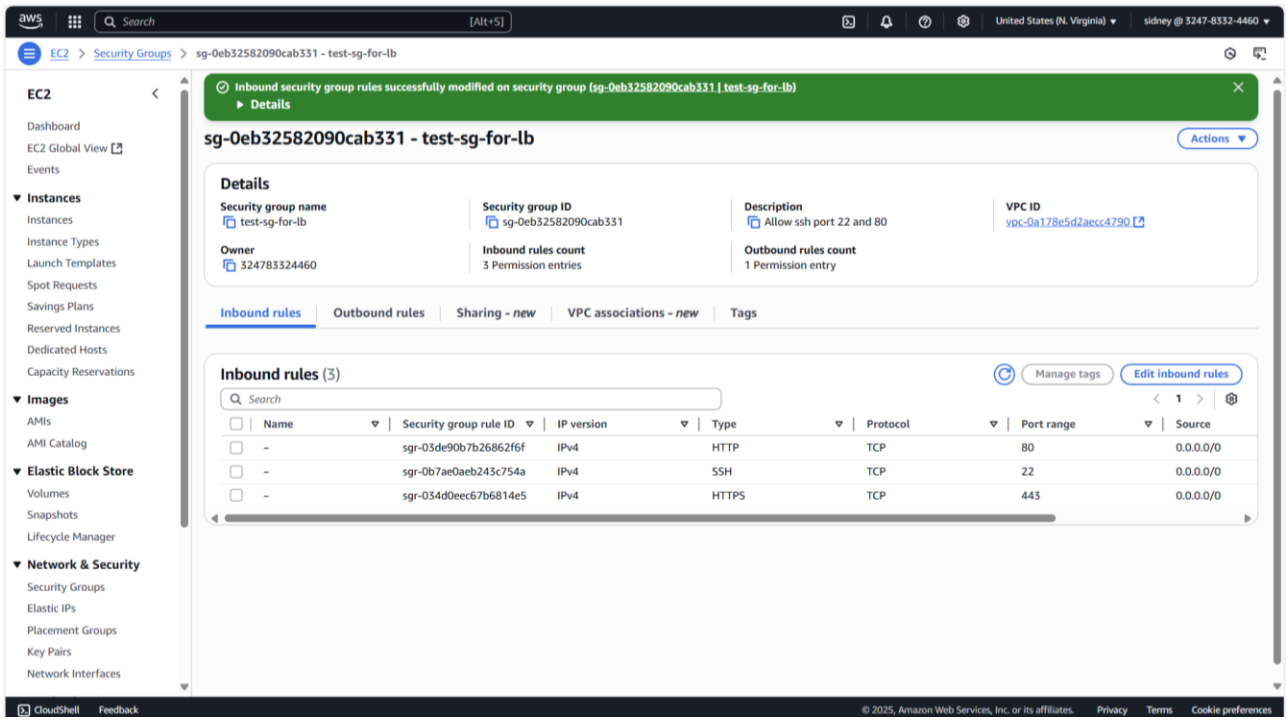
Click on the drop down and select “**HTTPS**”



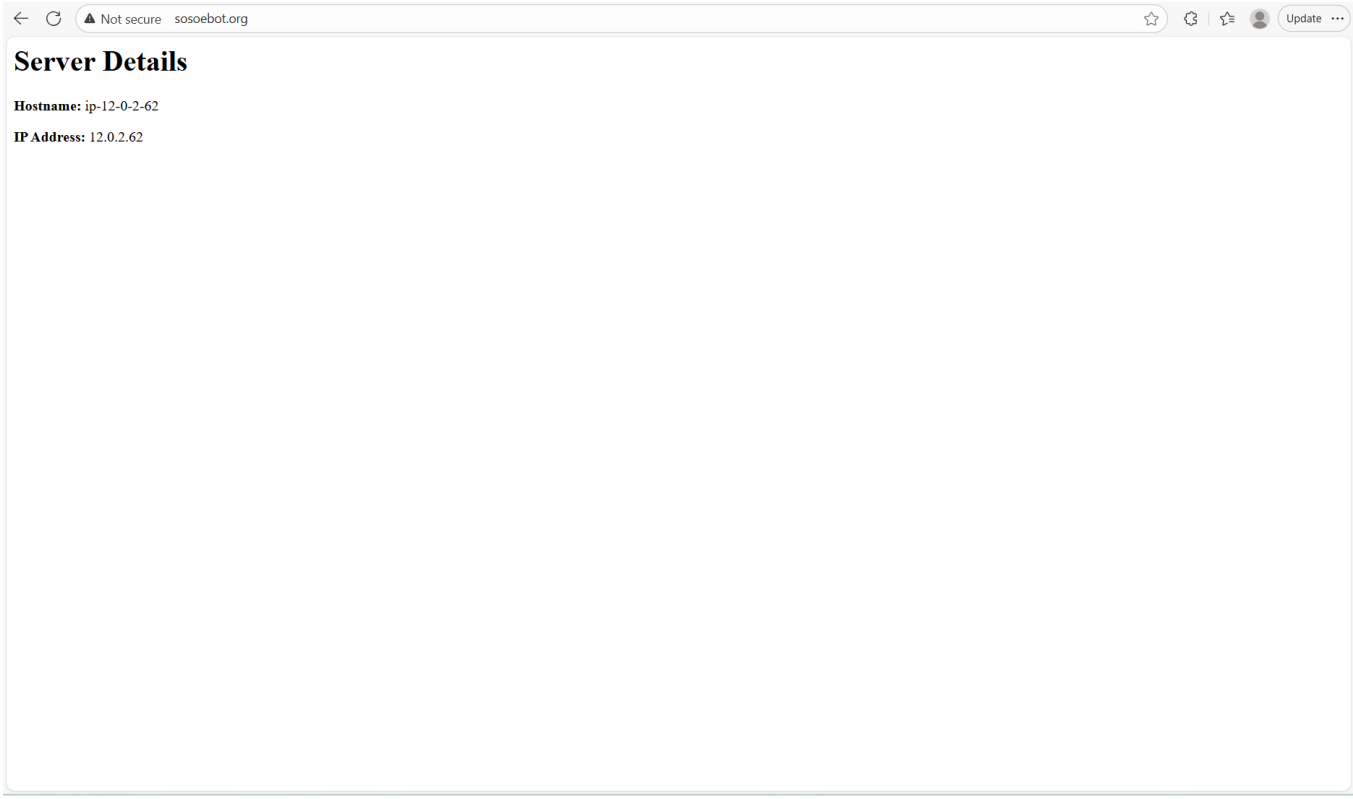
Click on the drop down on “**Source**” and select “**Anywhere-IPv4**”



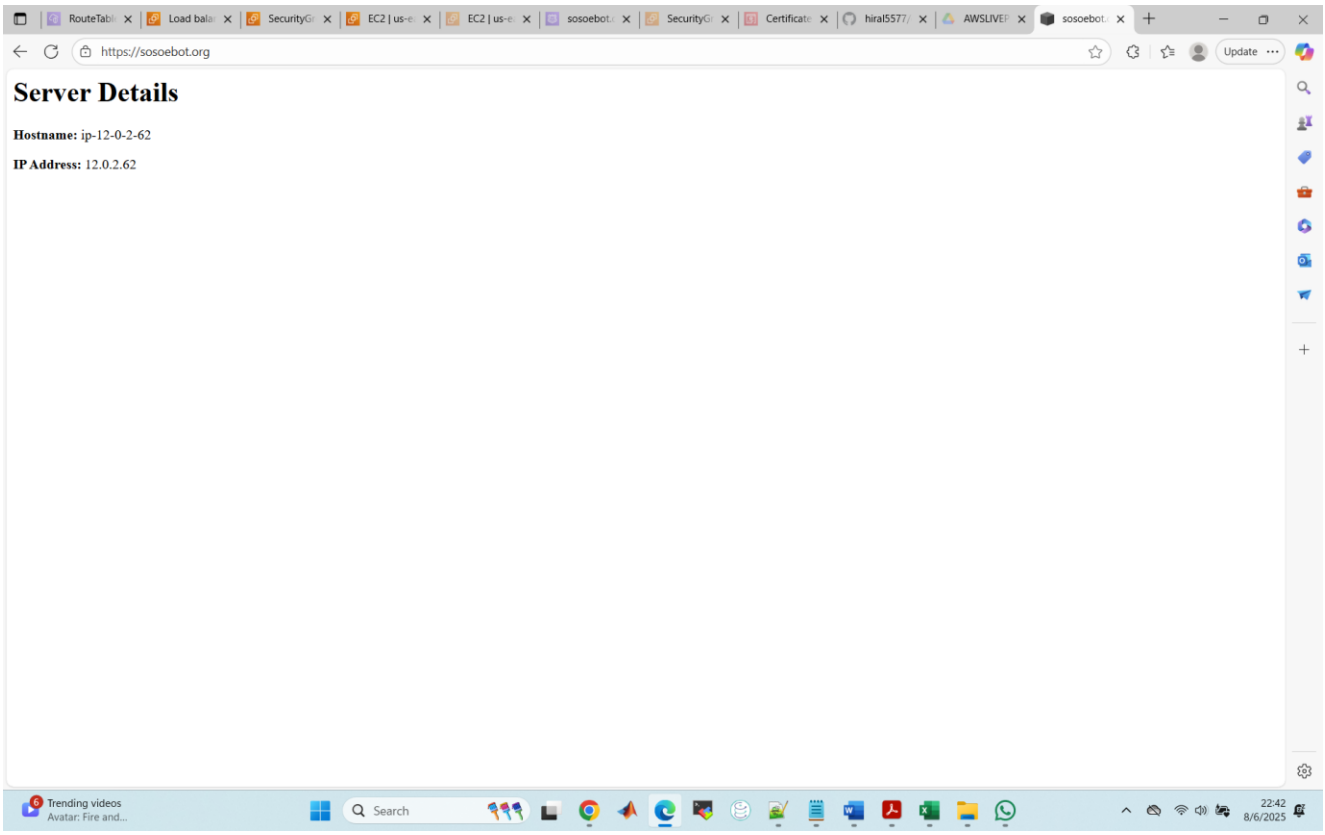
Click on "**Save Rules**"



Let us first verify by using http://sosoebot.org on our browser



Now, let us verify by using **https://sosoebot.org** on our browser

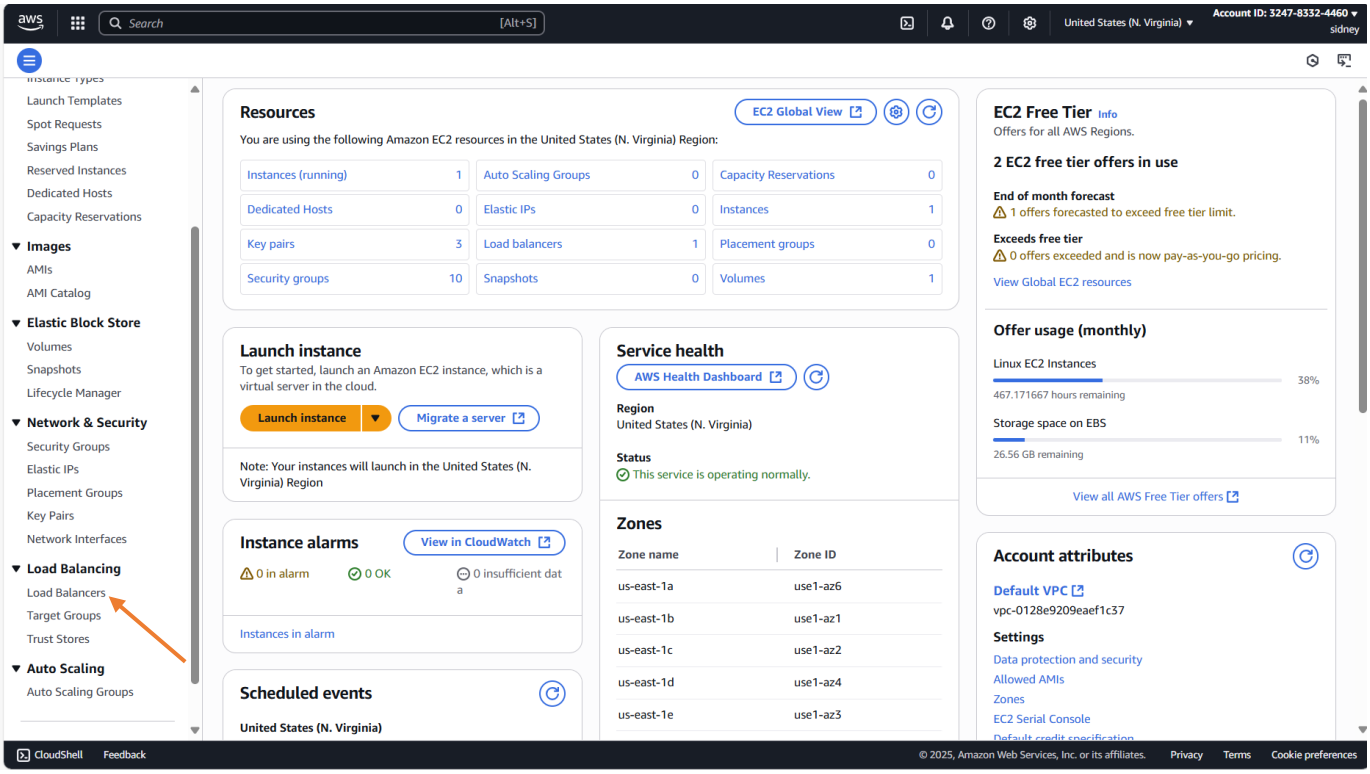


It works. We are able to use HTTPS inside our URL. Our both URL patterns are working but the thing which we need to notice here is we need to stop the HTTP request or at least we need to redirect this particular request which is coming without HTTPS to our HTTPS part.

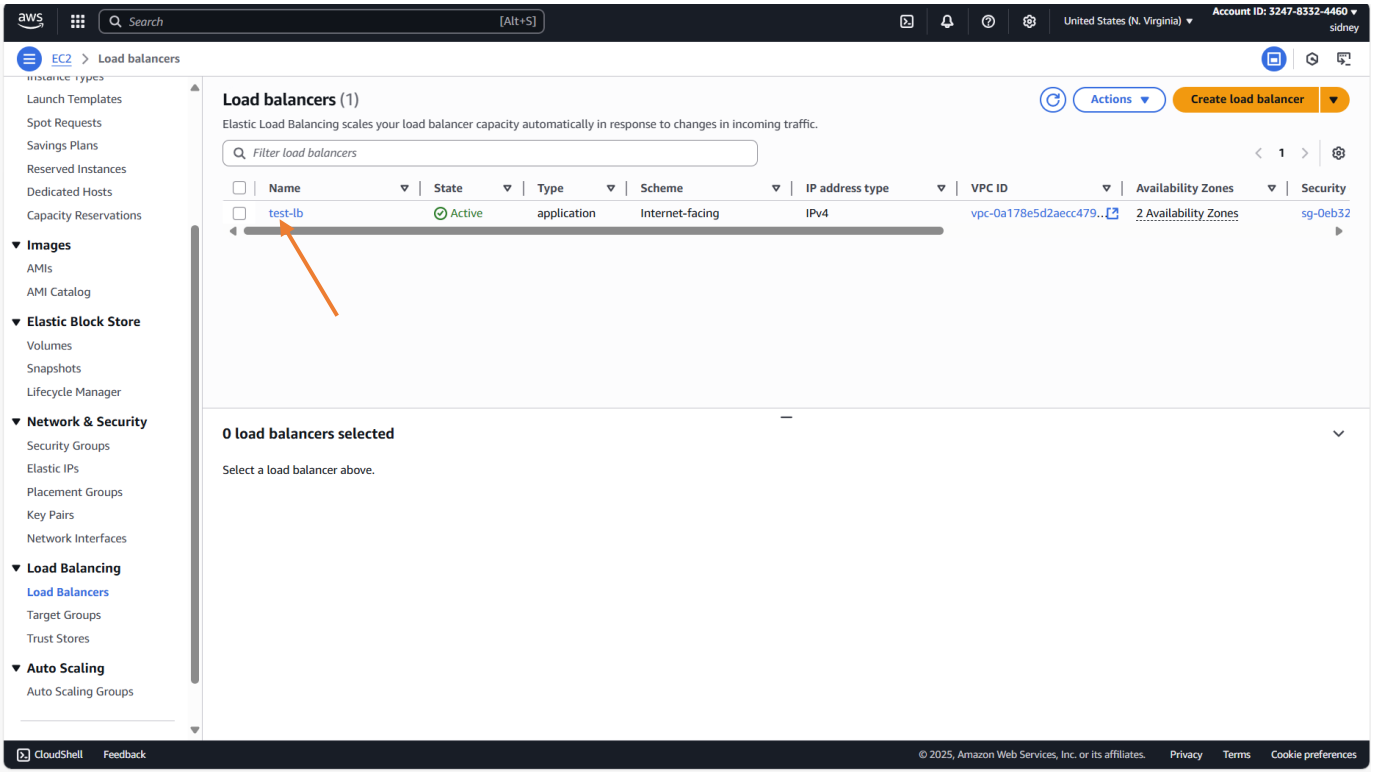
The next thing we are going to fix is to redirect this particular request where we don't put HTTPS and we should redirect this request to land on this HTTP URL.

How do we fix the redirection issue?

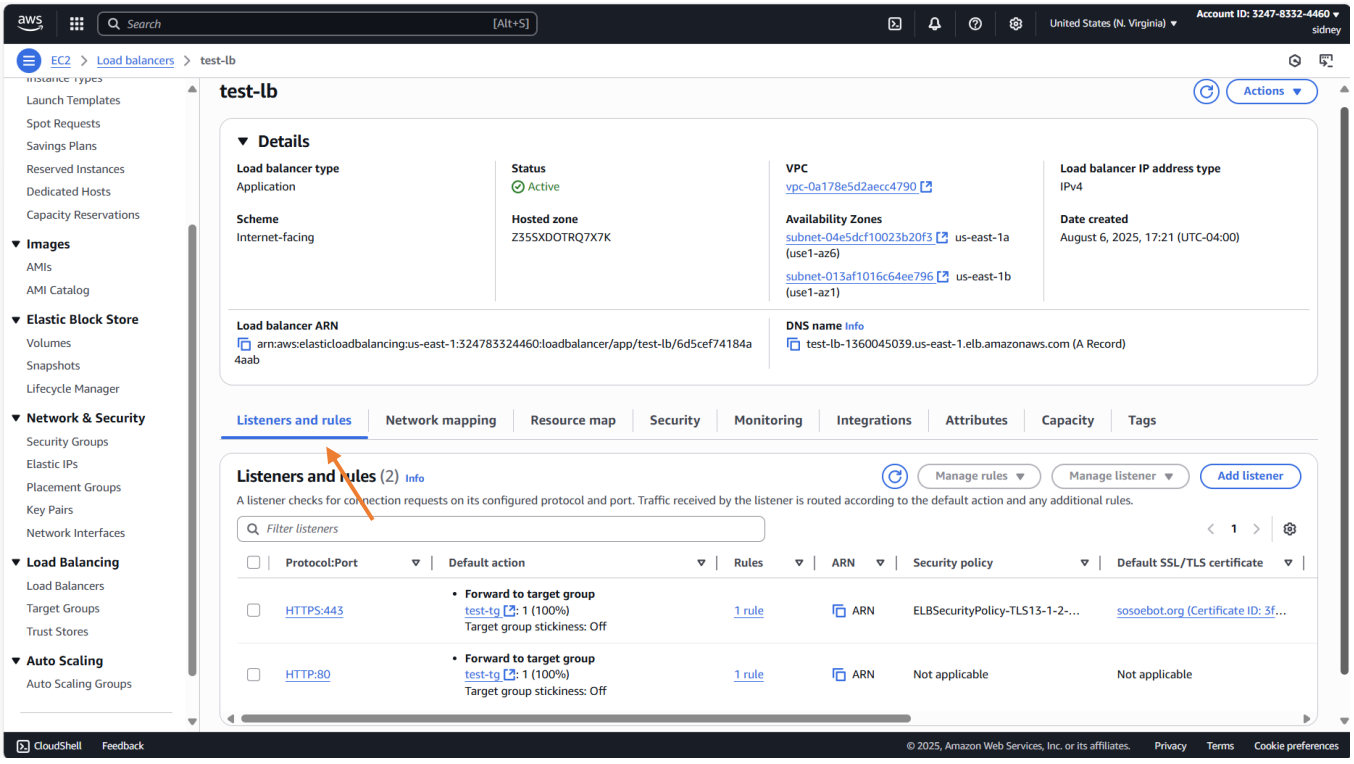
Go to your EC2 dashboard



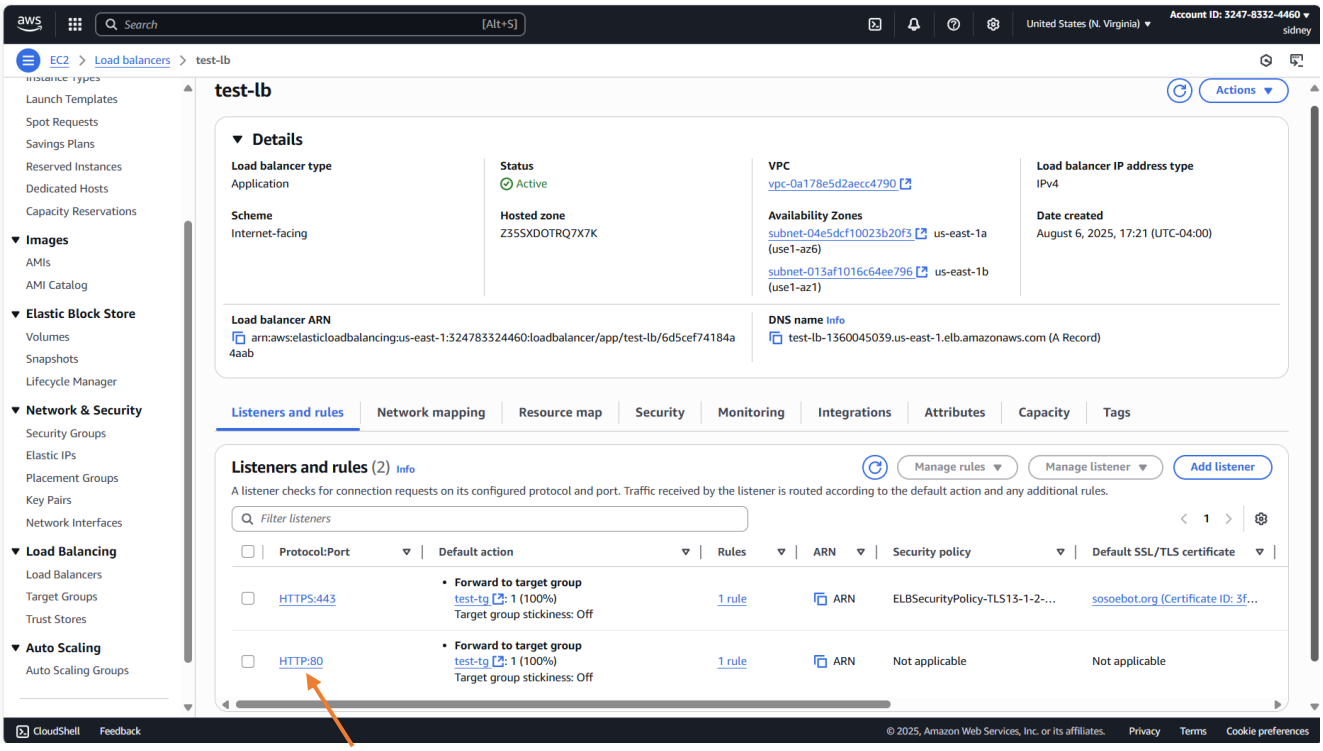
Click on "**Load Balancer**"



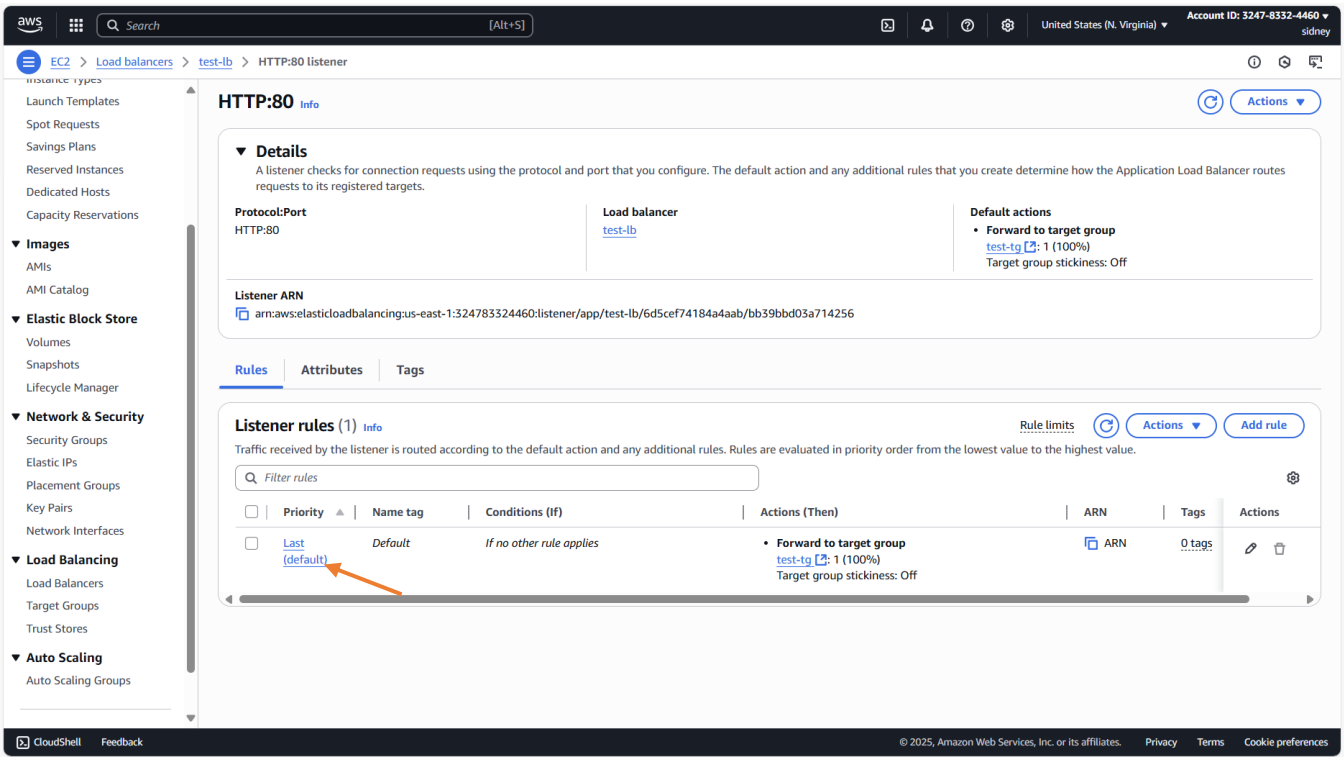
Click on the Load Balancer name, that is "**test-lb**"



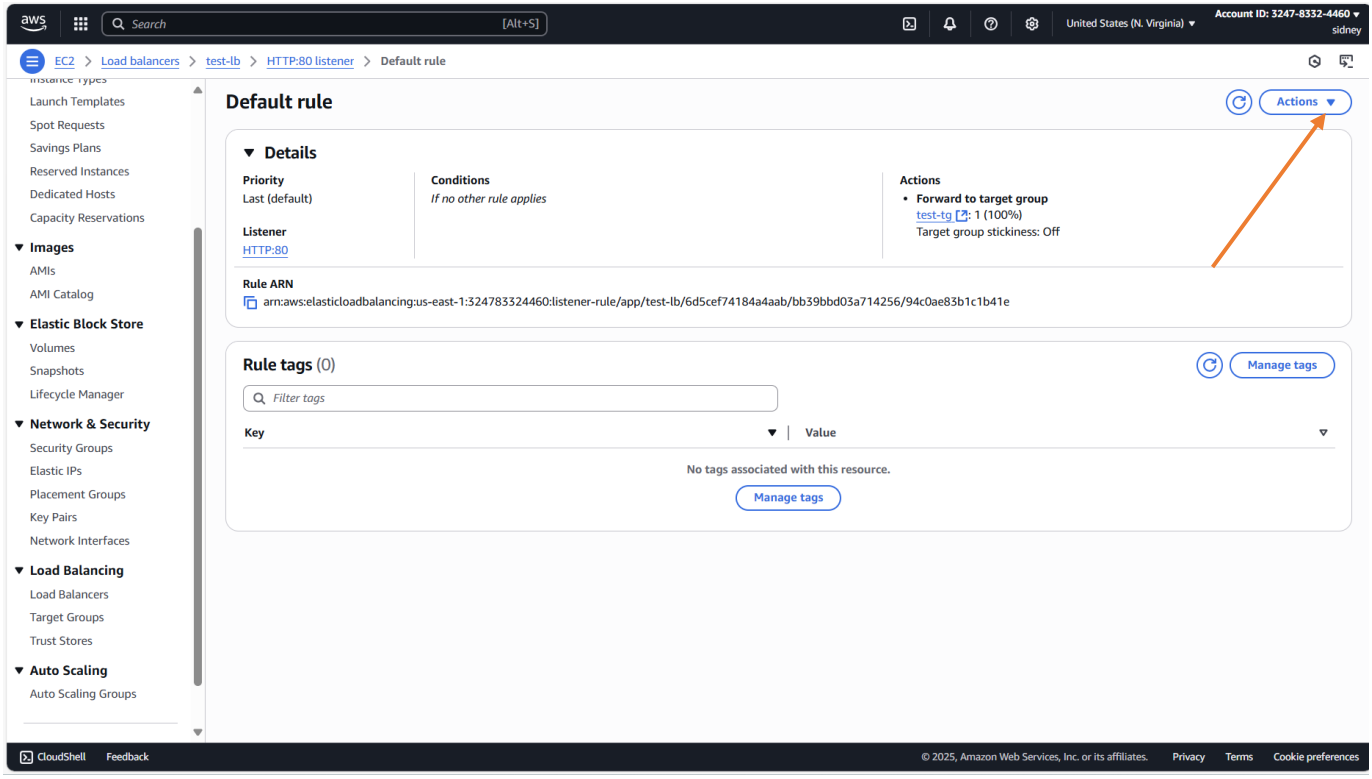
Go to "**Listener and Rules**"



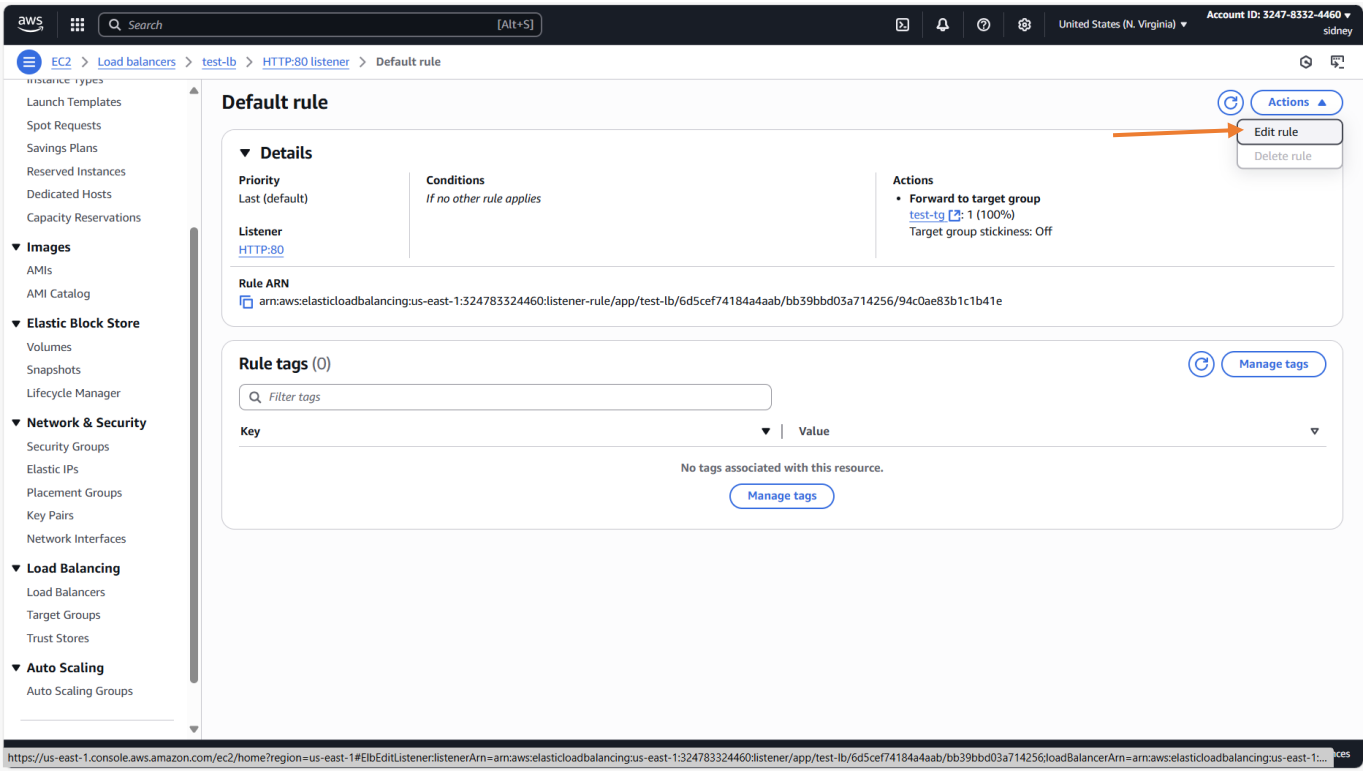
Now, we are interested into the request only for HTTP. We want to let go that request and redirect it to the HTTPS. Click on "**HTTP**"



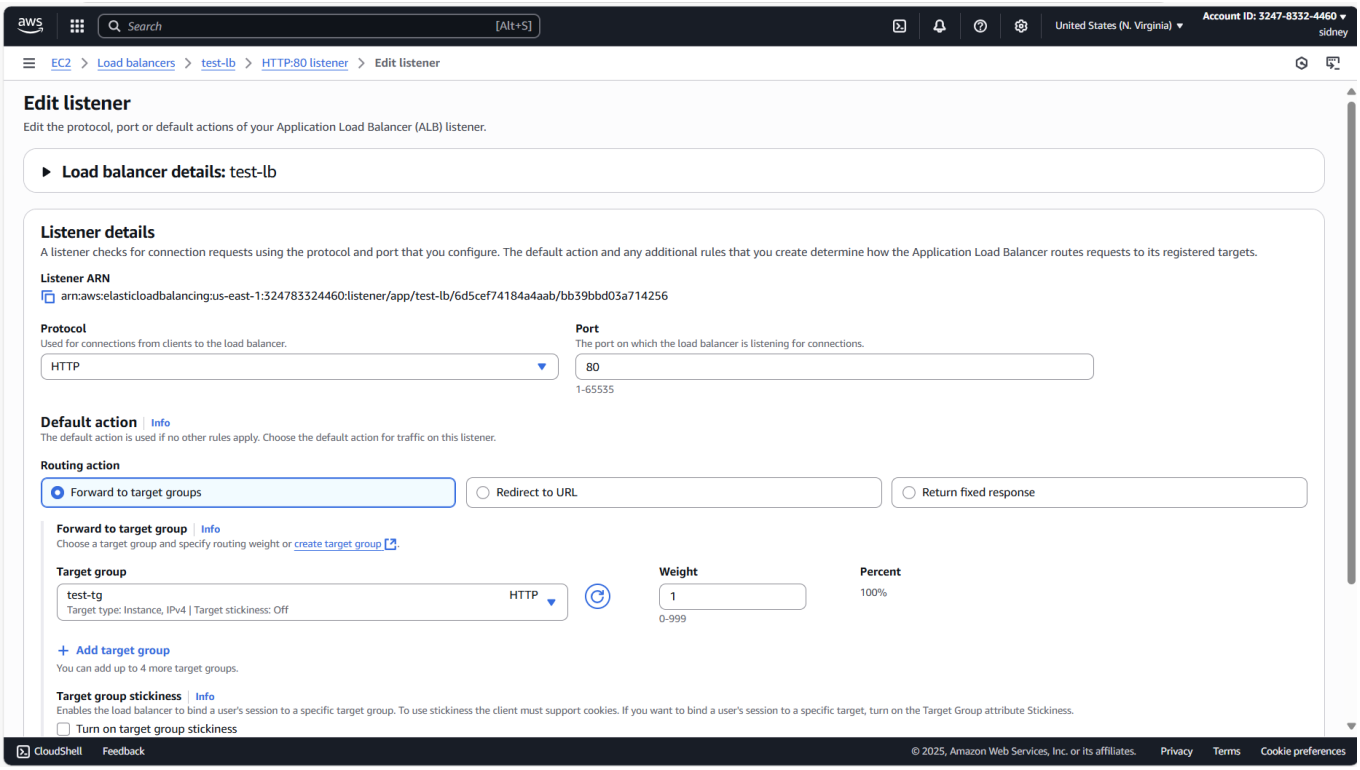
Click on "**Default**" rule



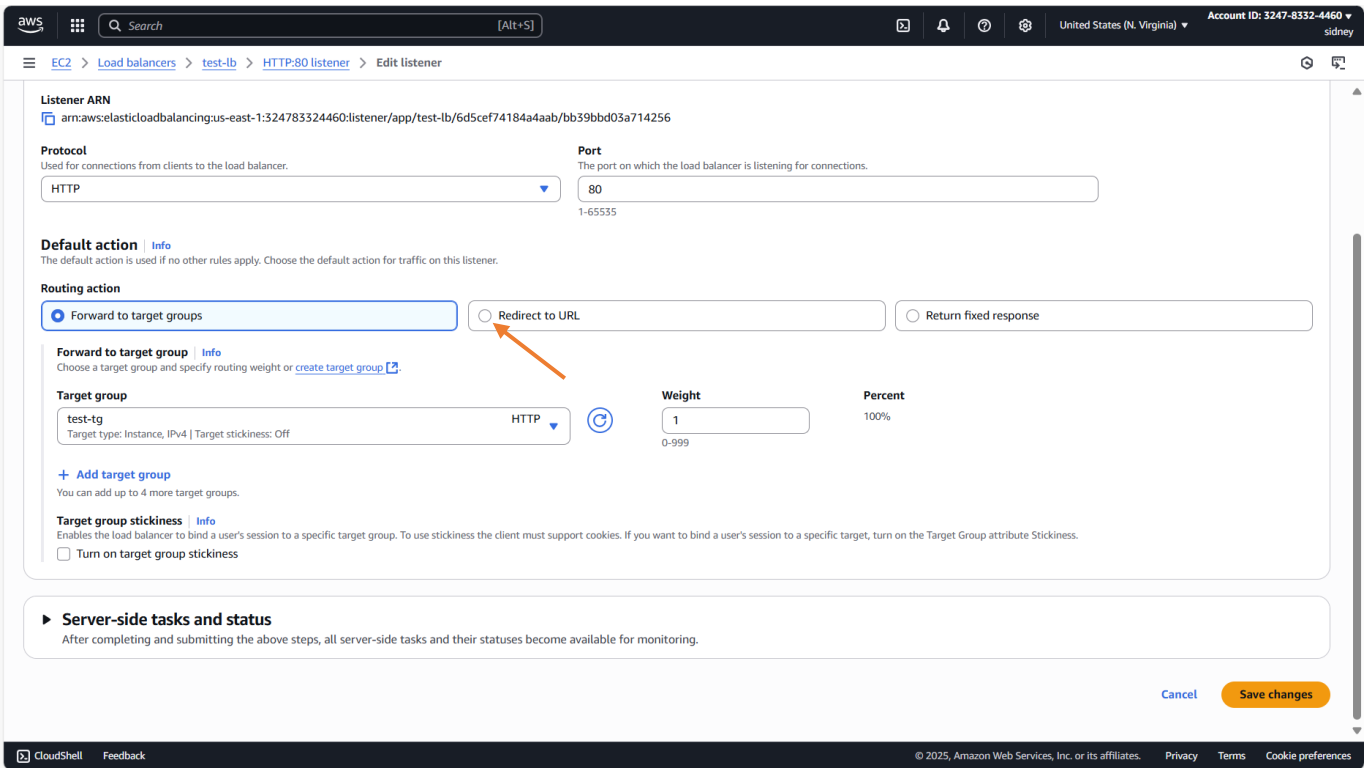
Then click on the drop down on "**Action**"



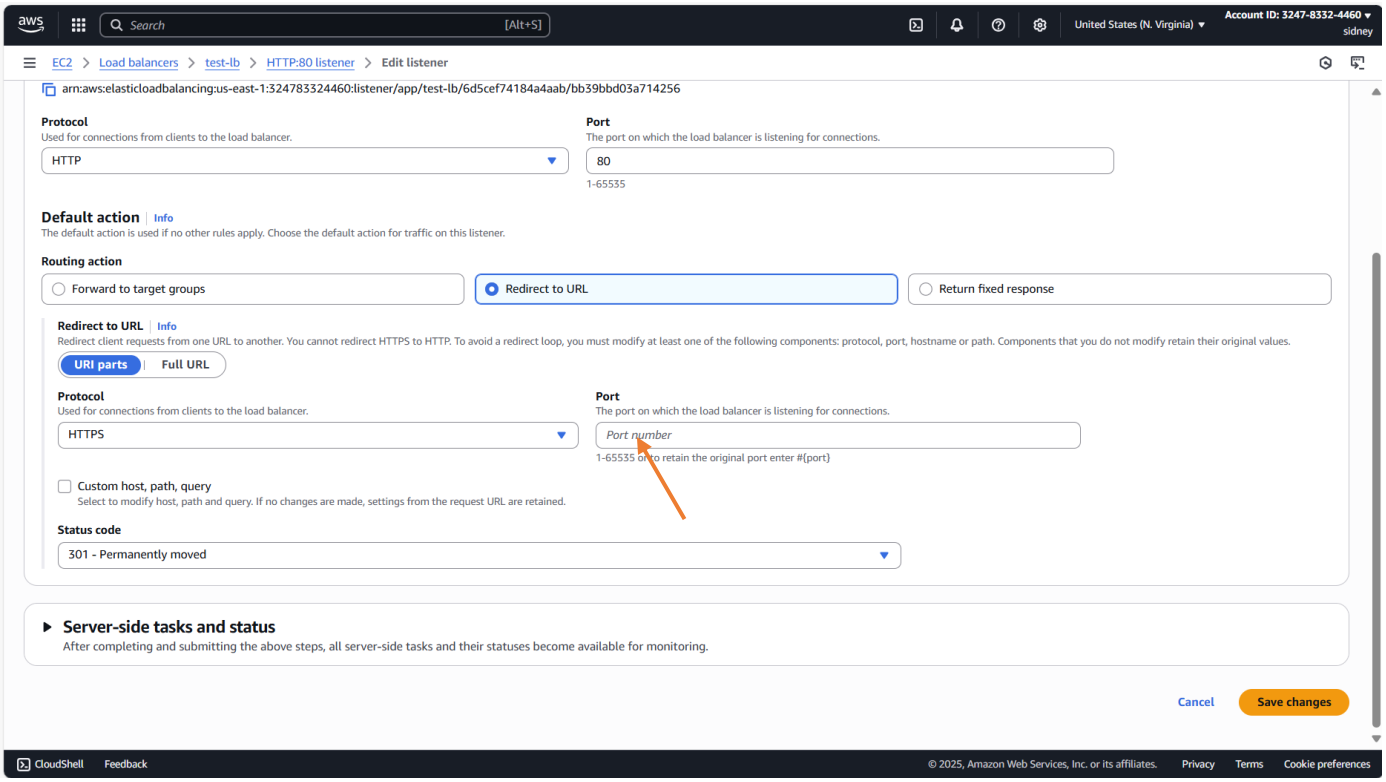
Select "**Edit Rule**"



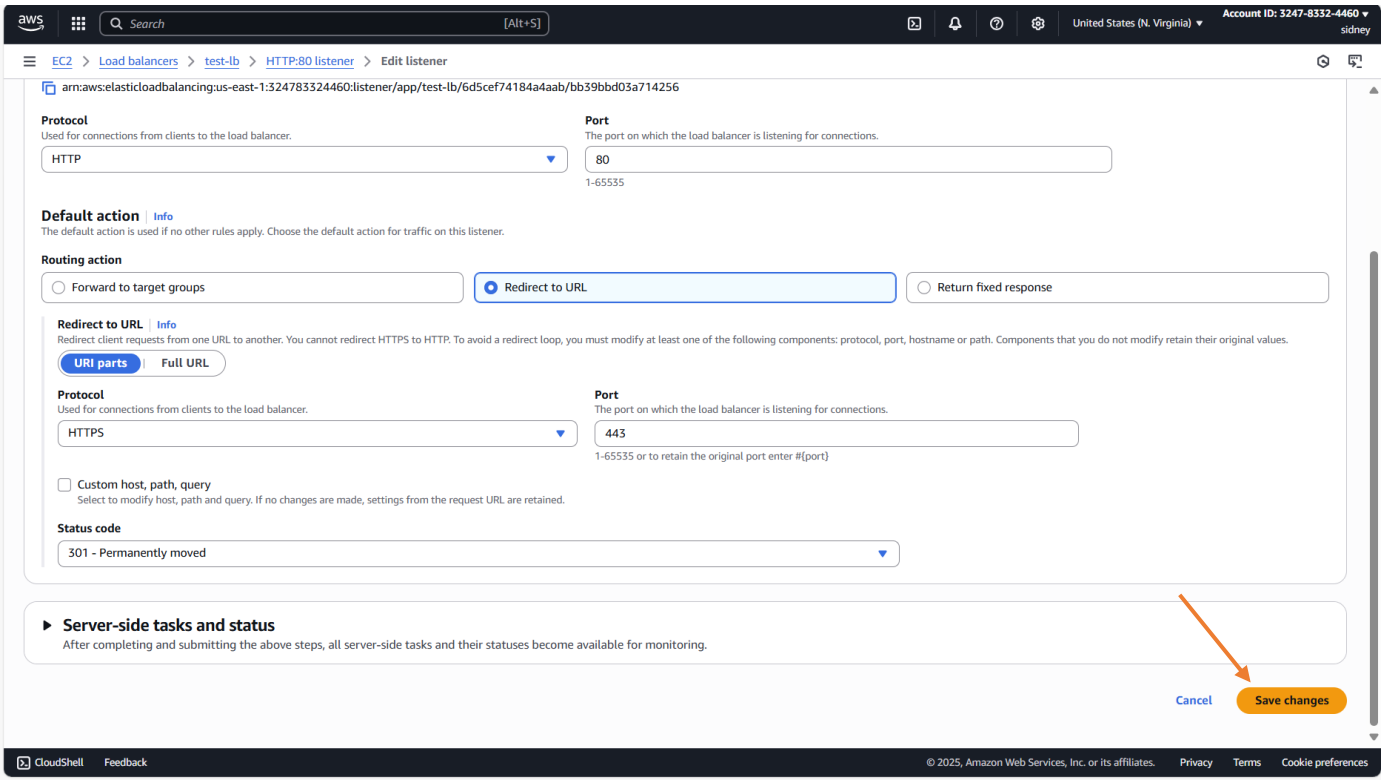
Scroll down to “**Default Actions**”



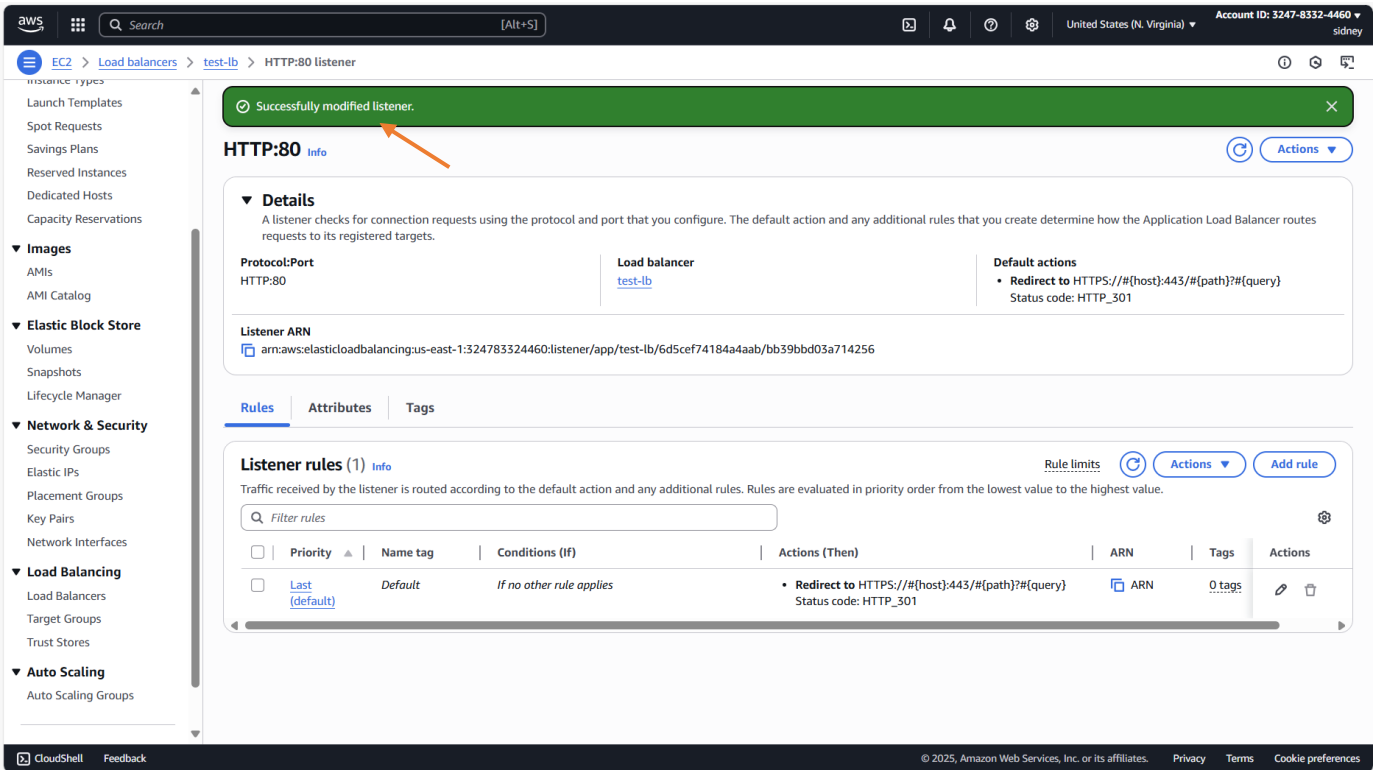
Select “**Redirect to URL**”



On PORT enter "**443**" and make sure "**Custom host, path, query**" is unchecked.

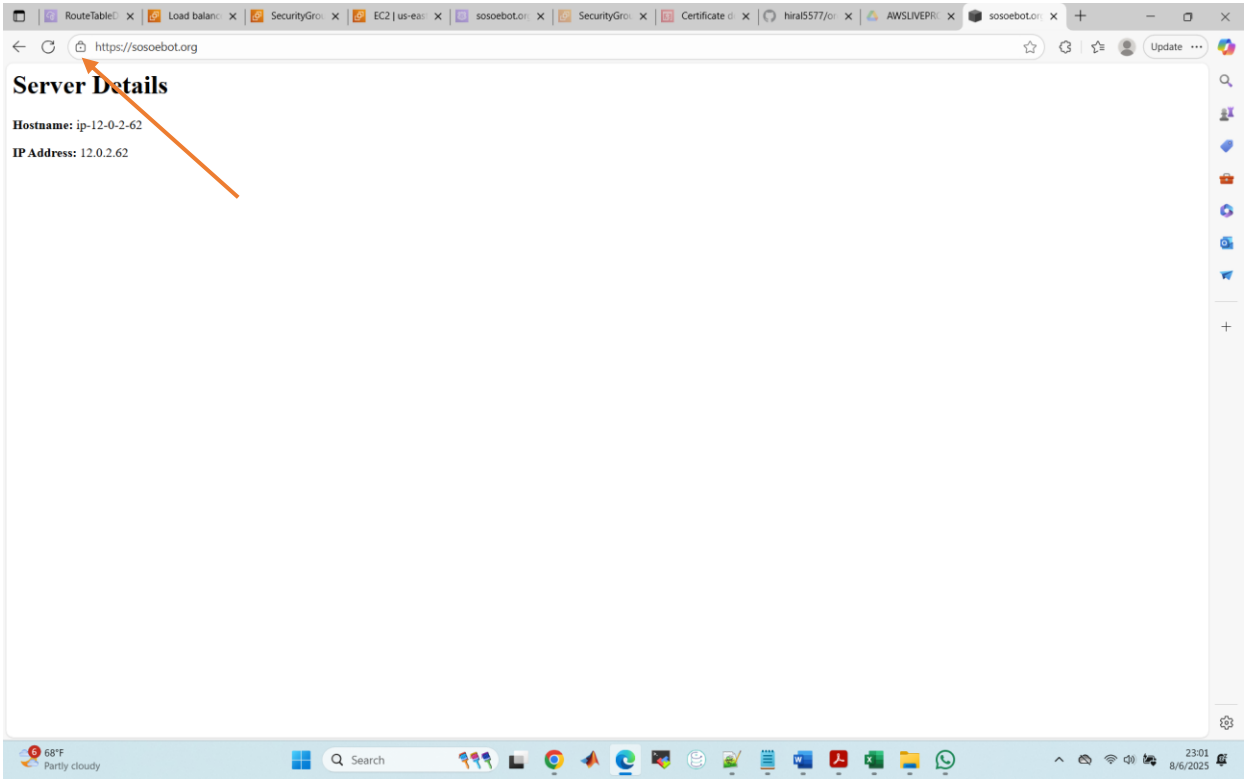


Click on "**Save Changes**"



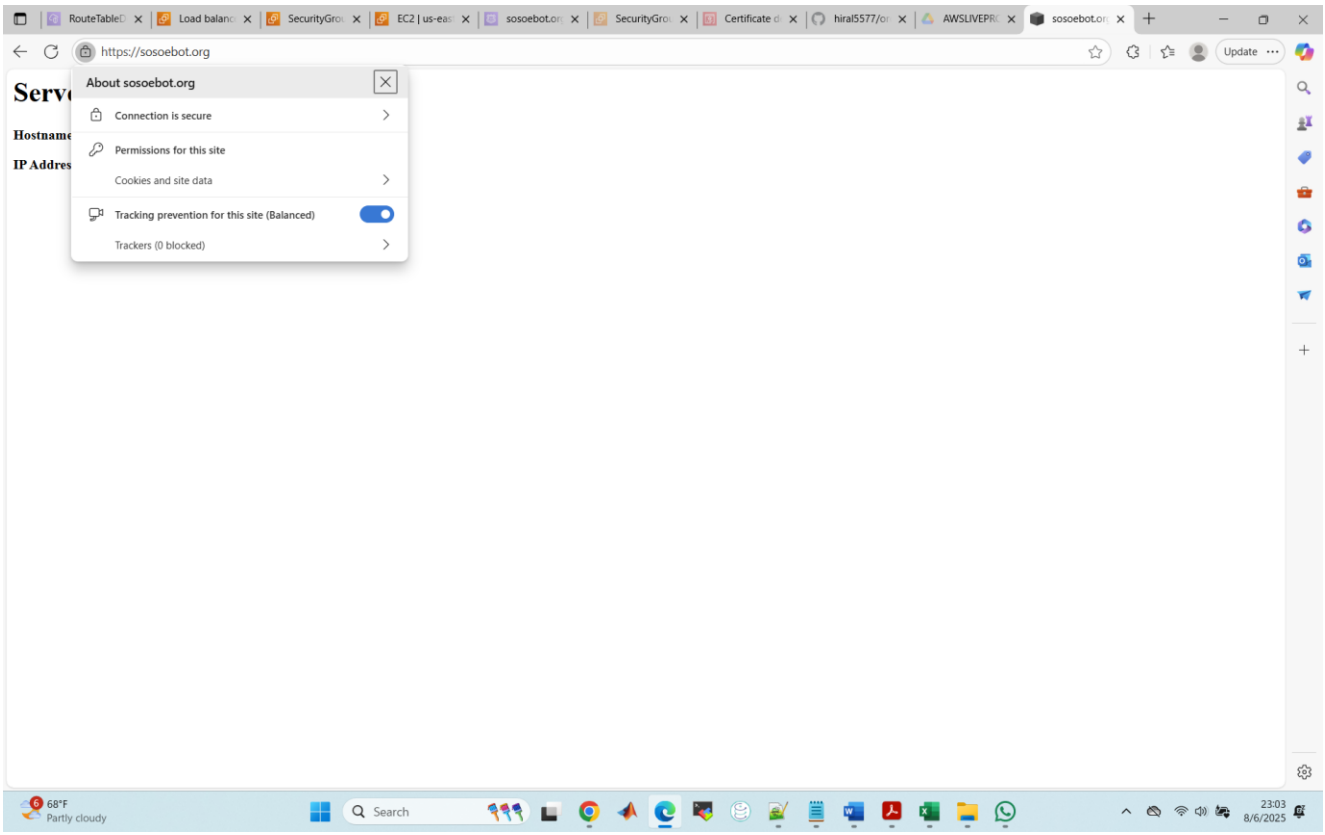
You can see "**successfully modified the listener**". You can wait for a couple of minutes and then it will start redirecting the normal HTTP request to the HTTPS.

Let us test it now. We will enter http://sosoebot.org on our browser and hit Enter. It will redirect us to https://sosoebot.org.



You can see it has redirected me to https://sosoebot.org

You can verify



You can see that the connection is secured.

So, this is how you are going to set up your SSL and you are going to redirect the normal http request to https.

In case you are using some other domain, let us say for example www.sosoebot.org. Then this is a different name, then you need to request a certificate for this particular domain name as well.

So, you need to create a new certificate and there you need to enter the root name as www.sosoebot.org to request a new certificate for this domain name. Then push the records again to our Route 53, after that you will be able to access the URL or home page using the www URL pattern as well.

I give credit to Rahul Wagn for using images from his tutorial to put up this document