Deploy Nginx Server on EC2 Instance:

In this topic you were going to learn about the Nginx Server and how to deploy Nginx Server on EC2 instance. What I am going to perform here is to deploy on Nginx server on port 80 on EC2 instance and this EC2 instance should be behind the Auto Scaling groups (ASG) and Elastic Load Balancer (ELB). For Auto Scaling Groups (ASG) I am going to set the minimum instance to 1, and desired instance to 2 and maximum instance to 3.

Table of Contents:

Introduction:

To learn about this completely you need to learn about different concepts here. Firstly, we need to learn about the EC2 instances and how you were going to launch an EC2 instances in AWS. I will show you how you were going to launch an EC2 instance in real time. Secondly, you need to know about Elastic Load Balancer (ELB) and Auto Scaling Groups (ASG). Lastly, you need to learn about the steps that are involved in deploying a Nginx Server on EC2 instance and how you were going to start the server and other operations of it.

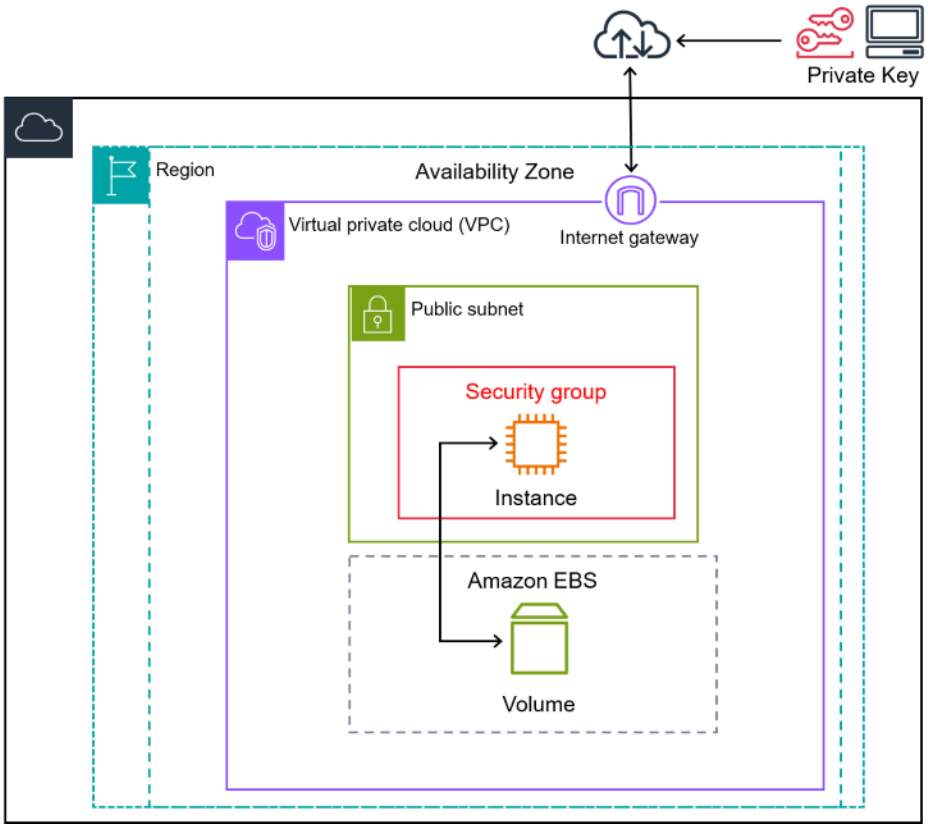
Elastic Cloud Compute (EC2):

Amazon Elastic Compute Cloud (Amazon EC2) offers the broadest and deepest compute platform, with over 750 instances and choice of the latest processor, storage, networking, operating system, and purchase model to help you best match the needs of your workload. They are the first major cloud provider that supports Intel, AMD, and Arm processors, the only cloud with on-demand EC2 Mac instances, and the only cloud with 400 Gbps Ethernet networking. They offer the best price performance for machine learning training, as well as the lowest cost per inference instances in the cloud. More SAP, high performance computing (HPC), ML, and Windows workloads run on AWS than any other cloud.

Amazon Elastic Compute Cloud (Amazon EC2) provides on-demand, scalable computing capacity in the Amazon Web Services (AWS) Cloud. Using Amazon EC2 reduces hardware costs so you can develop and deploy applications faster. You can use Amazon EC2 to launch as many or as few virtual servers as you need, configure security and networking, and manage storage. You can add capacity (scale up) to

handle compute heavy tasks, such as monthly or yearly processes, or spikes in website traffic. When usage decreases, you can reduce capacity (scale down) again.

The following diagram shows a basic architecture of an Amazon EC2 instance deployed within an Amazon Virtual Private Cloud (VPC). In this example, the EC2 instance is within an Availability Zone in the Region. The EC2 instance is secured with a security group, which is a virtual firewall that controls incoming and outgoing traffic. A private key is stored on the local computer and a public key is stored on the instance. Both keys are specified as a key pair to prove the identity of the user. In this scenario, the instance is backed by an Amazon EBS volume. The VPC communicates with the internet using an internet gateway.



Features of Amazon EC2:

Amazon EC2 provides the following high-level features:

Instances:

The EC2 instances, whatever we are going to create or created are virtual servers.

Amazon Machine Images (AMIs):

An Amazon Machine Image (AMI) is a supported and maintained image provided by AWS that provides the information required to launch an instance. You must specify an AMI when you launch an instance. You can launch multiple instances from a single AMI when you require multiple instances with the same configuration. You can use different AMIs to launch instances when you require instances with different configurations.

Instance types:

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instance types comprise varying combinations of CPU, memory, storage, and networking capacity and give you the flexibility to choose the appropriate mix of resources for your applications. Each instance type includes one or more instance sizes, allowing you to scale your resources to the requirements of your target workload.

Key pairs:

A key pair, consisting of a public key and a private key, is a set of security credentials that you use to prove your identity when connecting to an Amazon EC2 instance for security of instances. Amazon EC2 stores the public key on your instance, and you store the private key.

Instance store volumes:

An instance store provides temporary block-level storage for your instance. This storage is located on disks that are physically attached to the host computer. Instance store is ideal for temporary storage of information that changes frequently, such as buffers, caches, scratch data, and other temporary content. It can also be used to store temporary data that you replicate across a fleet of instances, such as a load-balanced pool of web servers.

An instance store consists of one or more instance store volumes exposed as block devices. The size of an instance store as well as the number of devices available varies by instance type and instance size.

The number, size, and type of instance store volumes are determined by the instance type and instance size. Some instance types, such as M6, C6, and R6, do not support instance store volumes, while other instance types, such as M5d, C6gd, and R6gd, do support instance store volumes. You can’t attach more instance store volumes to an instance than is supported by its instance type. For the instance types that do support instance store volumes, the number and size of the instance store volumes vary by instance size. For example, m5d.large supports 1 x 75 GB instance store volume, while m5d.24xlarge supports 4 x 900 GB instance store volumes.

Amazon EBS volumes:

Amazon Elastic Block Store (Amazon EBS) provides block level storage volumes for use with EC2 instances. EBS volumes behave like raw, unformatted block devices. You can mount these volumes as devices on your instances. EBS volumes that are attached to an instance are exposed as storage volumes

that persist independently from the life of the instance. You can create a file system on top of these volumes or use them in any way you would use a block device (such as a hard drive). You can dynamically change the configuration of a volume attached to an instance.

Regions, Availability Zones, Local Zones, AWS Outposts, and Wavelength Zones:

Multiple physical locations for your resources, such as instances and Amazon EBS volumes.

Security groups:

A security group acts as a virtual firewall for your EC2 instances to control incoming and outgoing traffic. Inbound rules control the incoming traffic to your instance, and outbound rules control the outgoing traffic from your instance. When you launch an instance, you can specify one or more security groups. If you don't specify a security group, Amazon EC2 uses the default security group for the VPC. You can add rules to each security group that allow traffic to or from its associated instances. You can modify the rules for a security group at any time. New and modified rules are automatically applied to all instances that are associated with the security group. When Amazon EC2 decides whether to allow traffic to reach an instance, it evaluates all the rules from all the security groups that are associated with the instance.

Elastic Load Balancer (ELB):

Elastic Load Balancing automatically distributes incoming application traffic across multiple targets, such as Amazon EC2 instances, Docker containers, IP addresses, and Lambda functions. It can handle the varying load of your application traffic in a single Availability Zone or across multiple Availability Zones. It monitors the health of its registered targets, and routes traffic only to the healthy targets. Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.

A load balancer accepts incoming traffic from clients and routes requests to its registered targets (such as EC2 instances) in one or more Availability Zones. The load balancer also monitors the health of its registered targets and ensures that it routes traffic only to healthy targets. When the load balancer detects an unhealthy target, it stops routing traffic to that target. It then resumes routing traffic to that target when it detects that the target is healthy again.

You configure your load balancer to accept incoming traffic by specifying one or more listeners. A listener is a process that checks for connection requests. It is configured with a protocol and port number for connections from clients to the load balancer. Likewise, it is configured with a protocol and port number for connections from the load balancer to the targets.

Elastic Load Balancing supports the following types of load balancers:

- Application Load Balancers
- Network Load Balancers

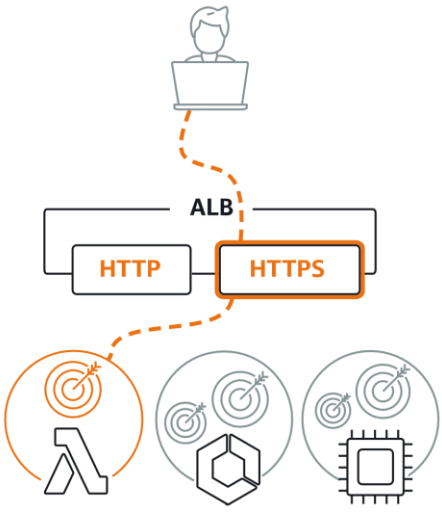
- Gateway Load Balancers
- Classic Load Balancers

There is a key difference in how the load balancer types are configured. With Application Load Balancers, Network Load Balancers, and Gateway Load Balancers, you register targets in target groups, and route traffic to the target groups. With Classic Load Balancers, you register instances with the load balancer.

Application Load Balancer:

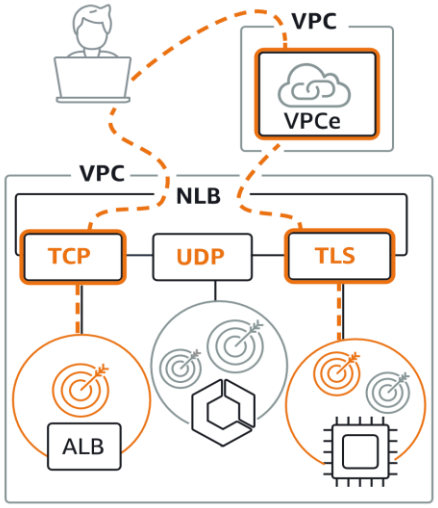
Choose an Application Load Balancer when you need a flexible feature set for your applications with HTTP and HTTPS traffic. Operating at the request level, Application Load Balancers provide advanced routing and visibility features targeted at application architectures, including microservices and containers.

For an example, if you change your web browser's language into French, an Application LB has visibility of the metadata it receives from your browser which contains details about the language you use. To optimize your browsing experience, it will then route you to the French-language servers on the backend behind the LB. You can also create advanced request routing, moving traffic into specific servers based on rules that you set yourself for specific cases.



Network Load Balancer:

Choose a Network Load Balancer when you need ultra-high performance, TLS offloading at scale, centralized certificate deployment, support for UDP, and static IP addresses for your applications. Operating at the connection level, Network Load Balancers are capable of handling millions of requests per second securely while maintaining ultra-low latencies.

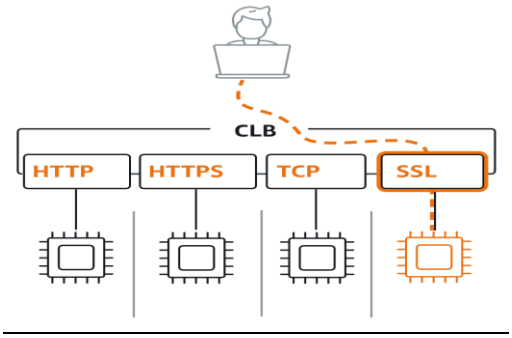


Gateway Load Balancer:

Choose a Gateway Load Balancer when you need to deploy and manage a fleet of third-party virtual appliances that support GENEVE. These appliances enable you to improve security, compliance, and policy controls.

Classic Load Balancer:

The Classic Load Balancer distributes incoming application traffic across multiple EC2 instance targets in multiple Availability Zones. This increases the fault tolerance of your applications.

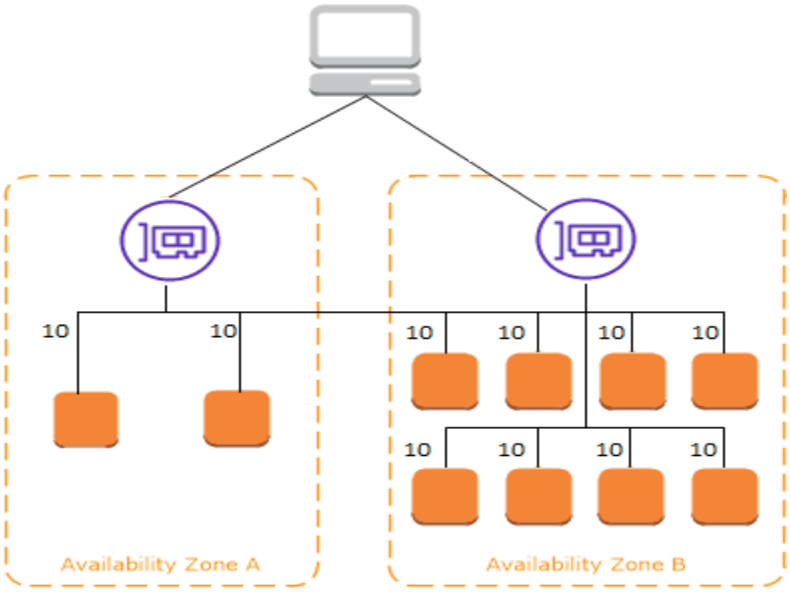


Cross-zone load balancing:

The nodes for your load balancer distribute requests from clients to registered targets. When cross-zone load balancing is enabled, each load balancer node distributes traffic across the registered targets in all enabled Availability Zones. When cross-zone load balancing is disabled, each load balancer node distributes traffic only across the registered targets in its Availability Zone.

The following diagrams demonstrate the effect of cross-zone load balancing with round robin as the default routing algorithm. There are two enabled Availability Zones, with two targets in Availability Zone A and eight targets in Availability Zone B. Clients send requests, and Amazon Route 53 responds to each request with the IP address of one of the load balancer nodes. Based on the round robin routing algorithm, traffic is distributed such that each load balancer node receives 50% of the traffic from the clients. Each load balancer node distributes its share of the traffic across the registered targets in its scope.

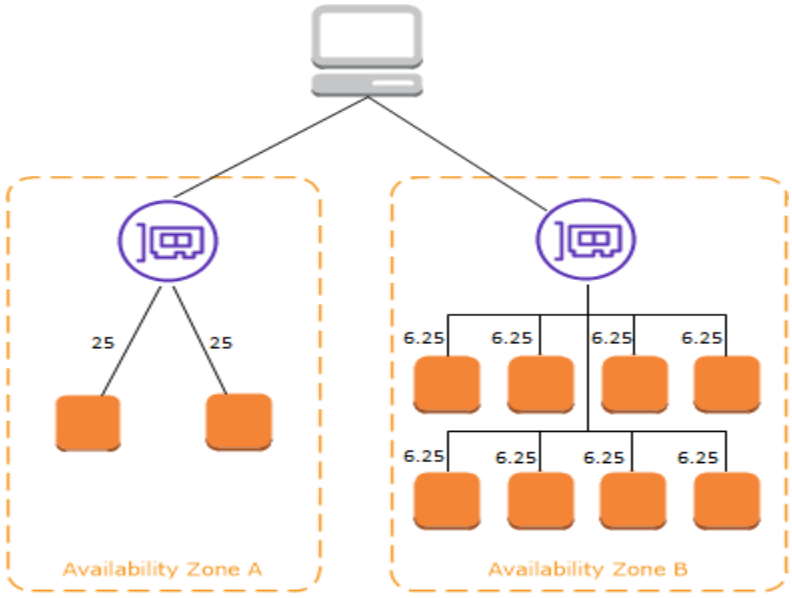
If cross-zone load balancing is enabled, each of the 10 targets receives 10% of the traffic. This is because each load balancer node can route its 50% of the client traffic to all 10 targets.



If cross-zone load balancing is disabled:

- Each of the two targets in Availability Zone A receives 25% of the traffic.
- Each of the eight targets in Availability Zone B receives 6.25% of the traffic.

This is because each load balancer node can route its 50% of the client traffic only to targets in its Availability Zone.



With Application Load Balancers, cross-zone load balancing is always enabled at the load balancer level. At the target group level, cross-zone load balancing can be disabled.

With Network Load Balancers and Gateway Load Balancers, cross-zone load balancing is disabled by default. After you create the load balancer, you can enable or disable cross-zone load balancing at any time.

When you create a Classic Load Balancer, the default for cross-zone load balancing depends on how you create the load balancer.

Zonal shift:

Zonal shift is a capability in Amazon Route 53 Application Recovery Controller (Route 53 ARC). With zonal shift, you can shift a load balancer resource away from an impaired Availability Zone with a single action. This way, you can continue operating from other healthy Availability Zones in an AWS Region.

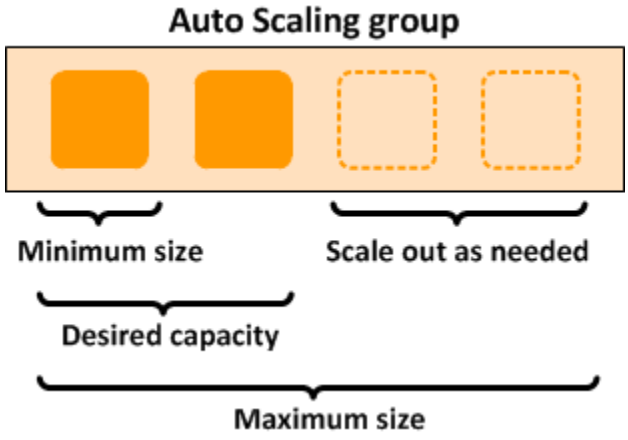
Zonal shifts are only supported on Application Load Balancers and Network Load Balancers with cross-zone load balancing turned off. If you turn on cross-zone load balancing, you can't start a zonal shift.

Auto Scaling Groups (ASG):

Amazon EC2 Auto Scaling:

Amazon EC2 Auto Scaling helps you ensure that you have the correct number of Amazon EC2 instances available to handle the load for your application. You create collections of EC2 instances, called Auto Scaling groups. You can specify the minimum number of instances in each Auto Scaling group, and Amazon EC2 Auto Scaling ensures that your group never goes below this size. You can specify the maximum number of instances in each Auto Scaling group, and Amazon EC2 Auto Scaling ensures that your group never goes above this size. If you specify the desired capacity, either when you create the group or at any time thereafter, Amazon EC2 Auto Scaling ensures that your group has this many instances. If you specify scaling policies, then Amazon EC2 Auto Scaling can launch or terminate instances as demand on your application increases or decreases.

For example, the following Auto Scaling group has a minimum size of one instance, a desired capacity of two instances, and a maximum size of four instances. The scaling policies that you define adjust the number of instances, within your minimum and maximum number of instances, based on the criteria that you specify.



Auto Scaling groups:

An Auto Scaling group contains a collection of EC2 instances that are treated as a logical grouping for the purposes of automatic scaling and management. An Auto Scaling group also lets you use Amazon EC2 Auto Scaling features such as health check replacements and scaling policies. Both maintaining the number of instances in an Auto Scaling group and automatic scaling are the core functionality of the Amazon EC2 Auto Scaling service.

The size of an Auto Scaling group depends on the number of instances that you set as the desired capacity. You can adjust its size to meet demand, either manually or by using automatic scaling.

An Auto Scaling group starts by launching enough instances to meet its desired capacity. It maintains this number of instances by performing periodic health checks on the instances in the group. The Auto Scaling group continues to maintain a fixed number of instances even if an instance becomes unhealthy. If an instance becomes unhealthy, the group terminates the unhealthy instance and launches another instance to replace it.

You can use scaling policies to increase or decrease the number of instances in your group dynamically to meet changing conditions. When the scaling policy is in effect, the Auto Scaling group adjusts the desired capacity of the group, between the minimum and maximum capacity values that you specify and launches or terminates the instances as needed. You can also scale on a schedule.

When creating an Auto Scaling group, you can choose whether to launch On-Demand Instances, Spot Instances, or both. You can specify multiple purchase options for your Auto Scaling group only when you use a launch template.

Spot Instances provide you with access to unused EC2 capacity at steep discounts relative to On-Demand prices There are key differences between Spot Instances and On-Demand Instances:

- The price for Spot Instances varies based on demand.
- Amazon EC2 can terminate an individual Spot Instance as the availability of, or price for, Spot Instances changes.

When a Spot Instance is terminated, the Auto Scaling group attempts to launch a replacement instance to maintain the desired capacity for the group.

When instances are launched, if you specify multiple Availability Zones, the desired capacity is distributed across these Availability Zones. If a scaling action occurs, Amazon EC2 Auto Scaling automatically maintains balance across all the Availability Zones that you specify.

Nginx Server:

Nginx [engine x] is an HTTP and reverse proxy server, a mail proxy server, and a generic TCP/UDP proxy server, originally written by Igor Sysoev.

Nginx provides high performance for Web servers with massive scaling. Nginx can run at high speeds under heavier loads. The reverse proxy feature allows a single site to present aggregated information sources as if they all come from one page. Its load balancer allows loads to be split among different resources such as servers.

Many prominent companies use Nginx to manage high-traffic pages, including Autodesk, Facebook, Atlassian, LinkedIn, Twitter, Apple, Citrix Systems, Intuit, T-Mobile, GitLab, DuckDuckGo, Target, Intel, Microsoft, IBM, Google, and Cisco.

Part of the reason Nginx scales so effectively and runs faster than other Web server software -- such as the standard Apache build -- is its more efficient use of processes. Unlike Apache builds, Nginx does not create a process per user. Nginx instead uses a master and worker process structure. The master process controls the worker processes which perform the calculations.

Nginx is important because it was purposely built for extreme loads and efficiency. The Web server software helps with several aspects of hosting Web site applications and content delivery services. Nginx is the second-most popular Web server software after Apache.

Installation steps for Nginx Server:

Step-1: Setup Yum repo for RHEL/CENTOS.

Command → sudo vi /etc/yum.repos.d/nginx.repo

Step-2: Add the following to the nginx.repo

```
[nginx]
name=nginx repo
baseurl=https://nginx.org/packages/centos/8/$basearch/
gpgcheck=0
enabled=1
```

Step-3: Update Yum repo

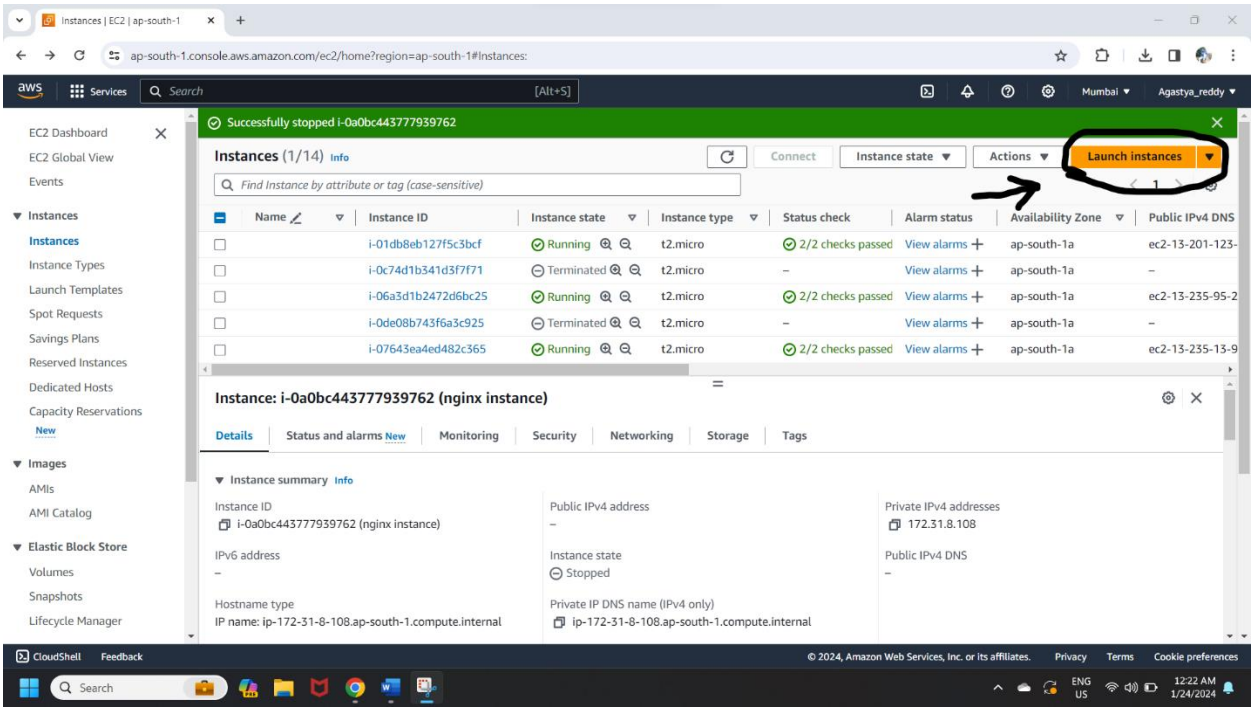
Command → sudo yum update

Step-4: Install NGINX Open source package:

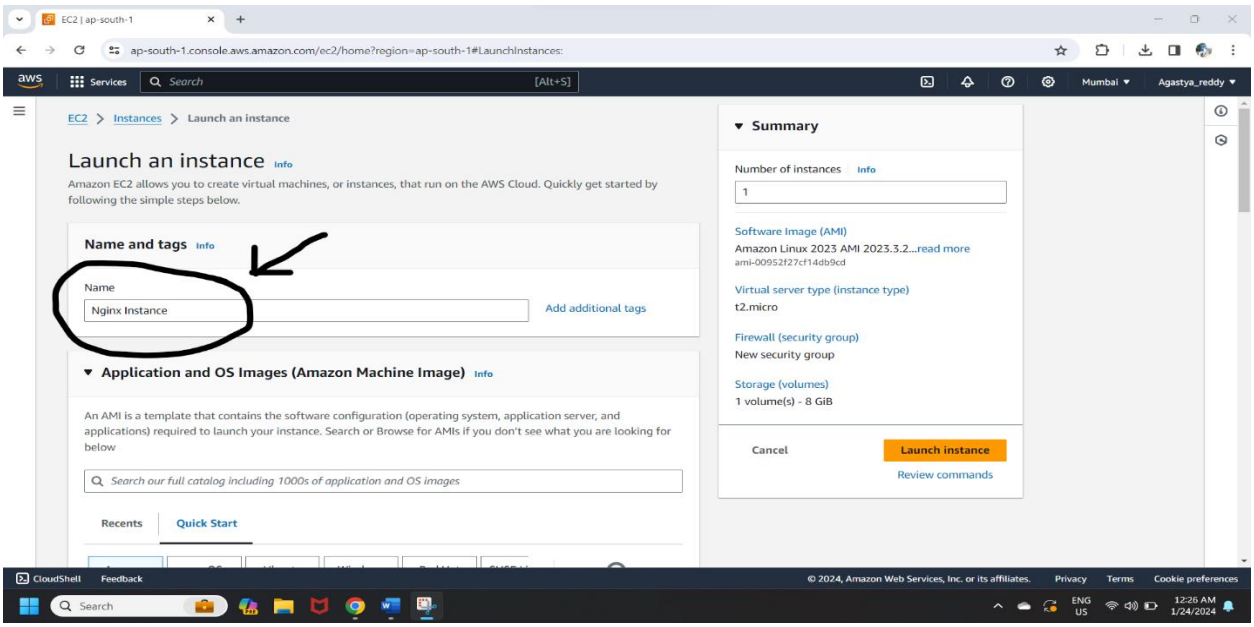
Command → sudo yum install nginx

Steps to deploy Nginx Server on EC2 instances:

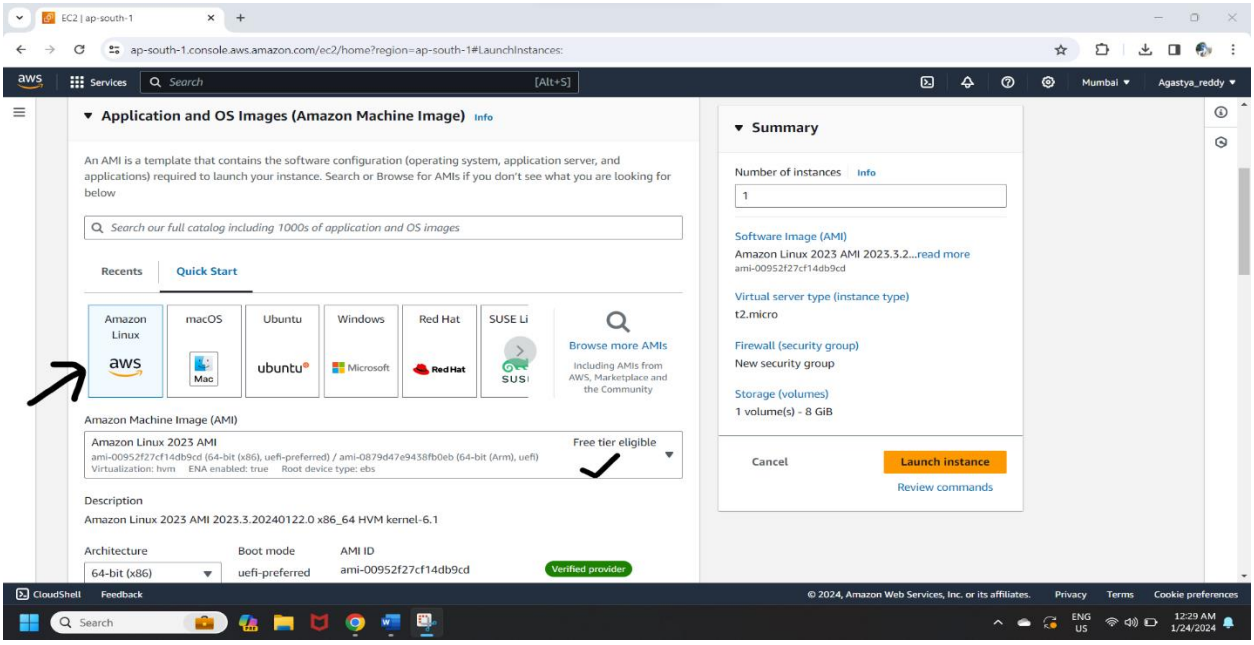
Step-1: Launch an EC2 instance by clicking on the “Launch instance” option.



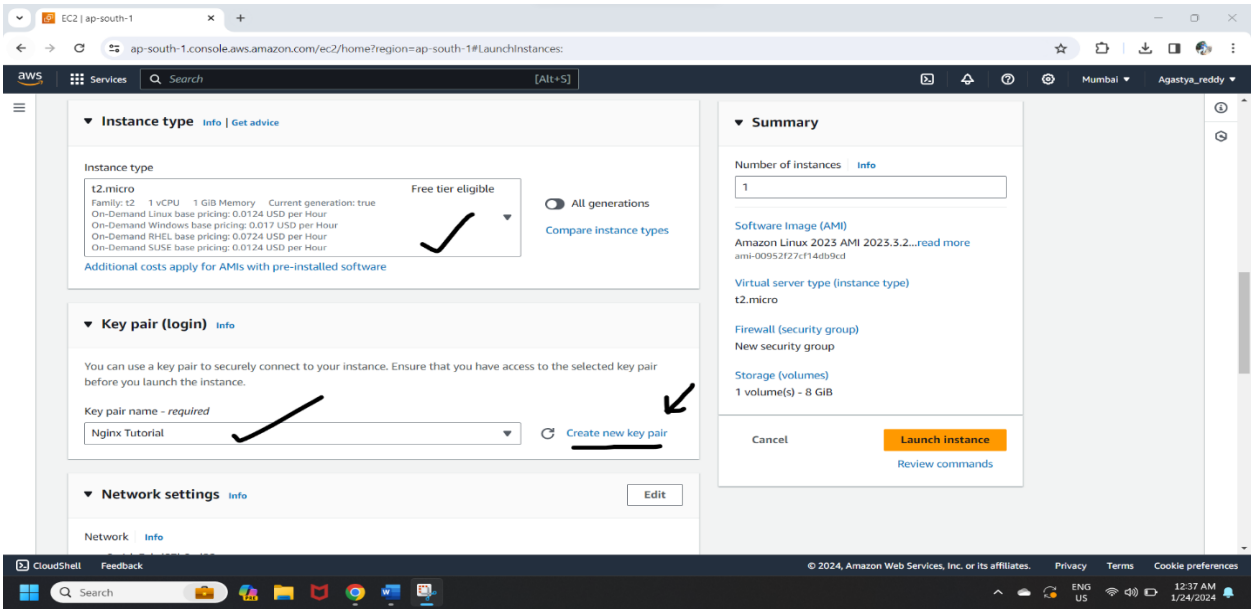
Step-2: Give a name of your wish for your EC2 instance.



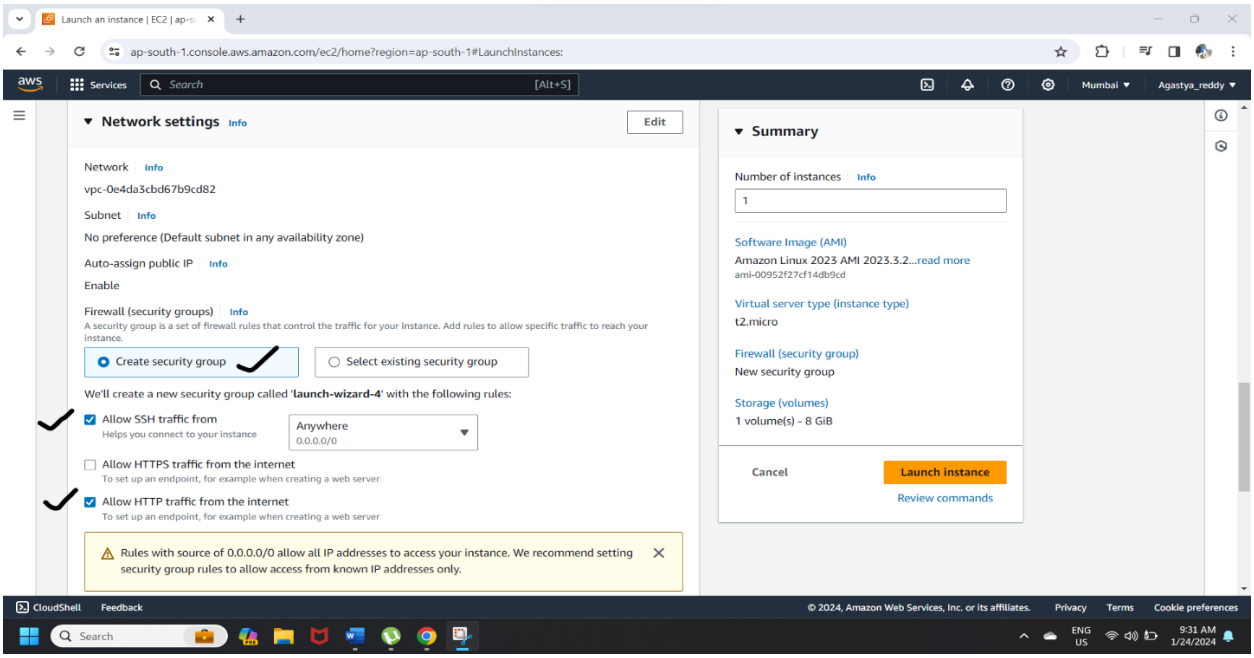
Step-3: After giving name for an EC2 instance, then we need to select the Amazon Machine Image (AMI). There are many AMIs available in Amazon EC2. But here I am going to select the "Amazon Linux 2023 AMI" which is a free tier eligible. And you also have many other options, you can select whatever AMI you need for your instance. Red Hat AMI is the popularly used AMI.



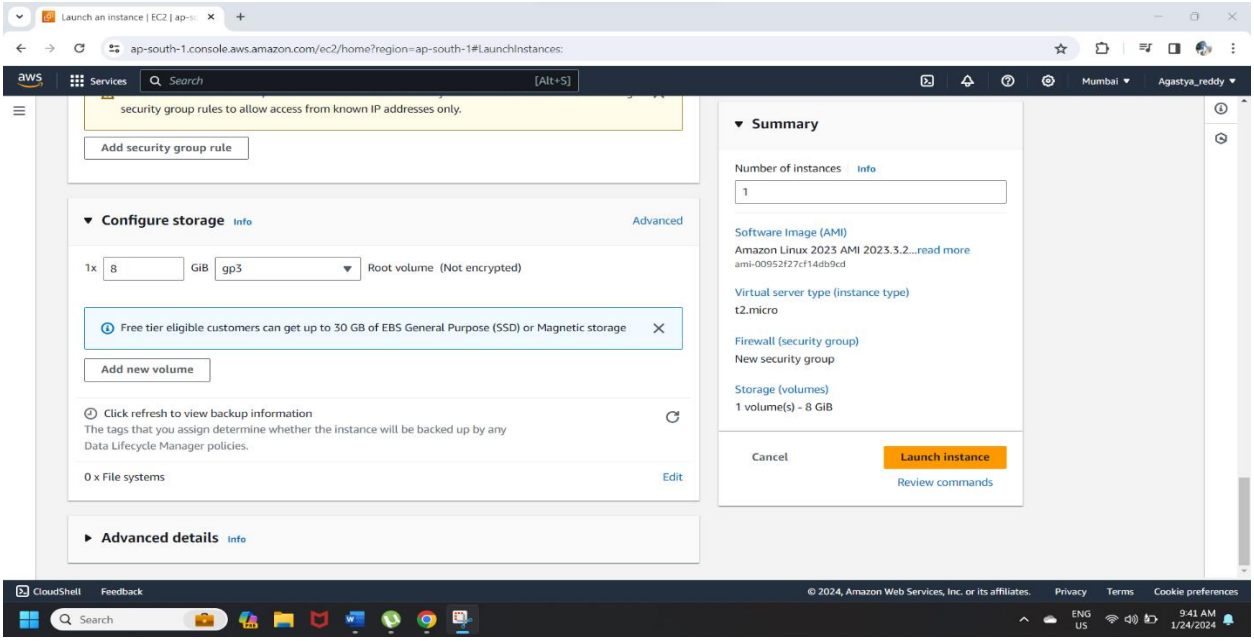
Step-4: Next you need to select the Instance type for your instance. There will be many options available for you, but I am going to select the "t2.micro" which is a free tier eligible Instance type. Next you need to give the Key Pair to instance by clicking on "create a new key pair" option. Neglect if you have created a key pair. And select the key pair which you created.



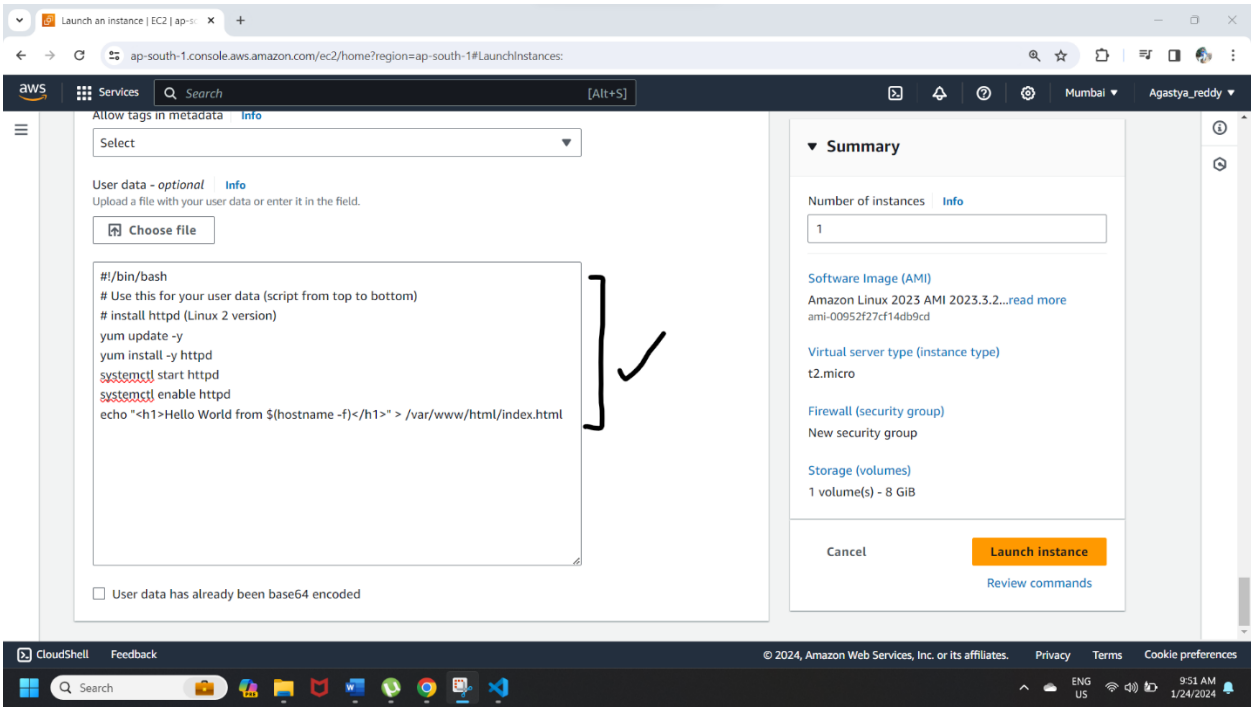
Step-5: In the Network Settings, you need to select the create security group option which helps you to create security group automatically called “launch-wizard-1” or any other name you could get if you created many instances before this. Next you need to select “Allow SSH traffic from” and “Allow HTTP traffic from the internet” options that will create SSH in port range 22 and for HTTP in port range 80. And leave the remaining as default.



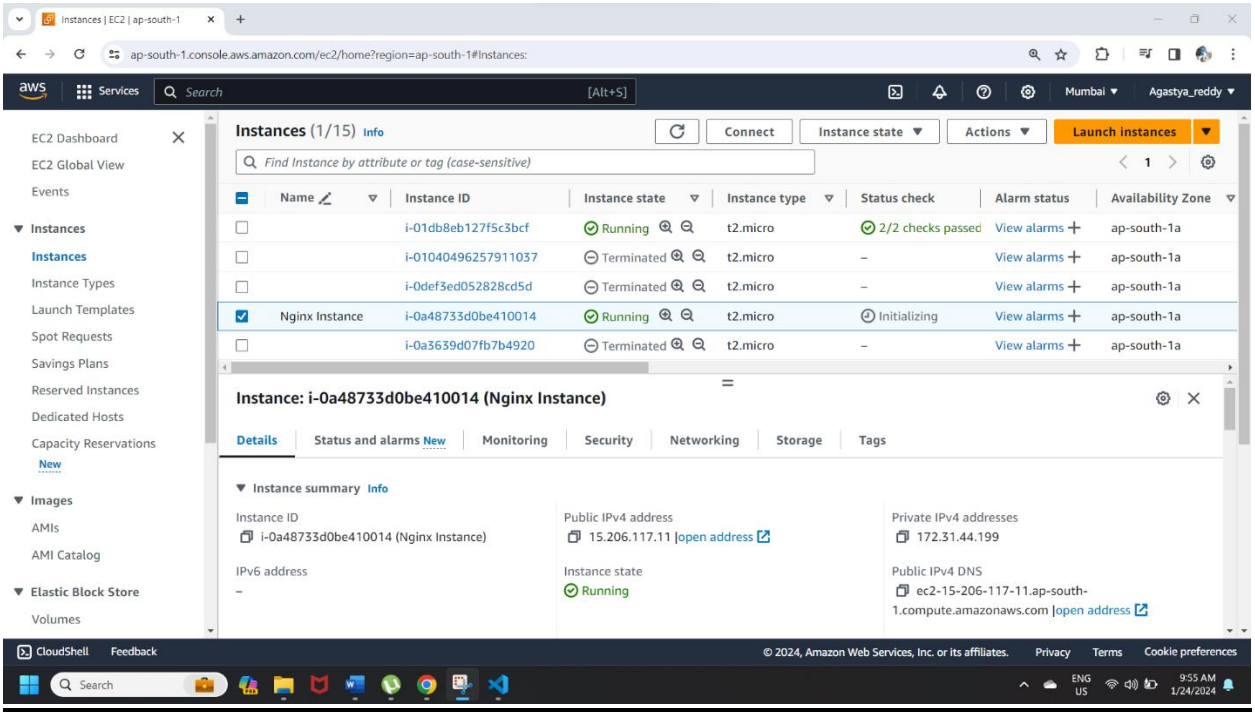
Step-6: Leave the configure settings as default. Or if you like to change the settings you can select whatever size you need and the root volume type. If you would like to give the file system, then select the file system option which is below. But here in this step I am going to leave it as default configure settings.



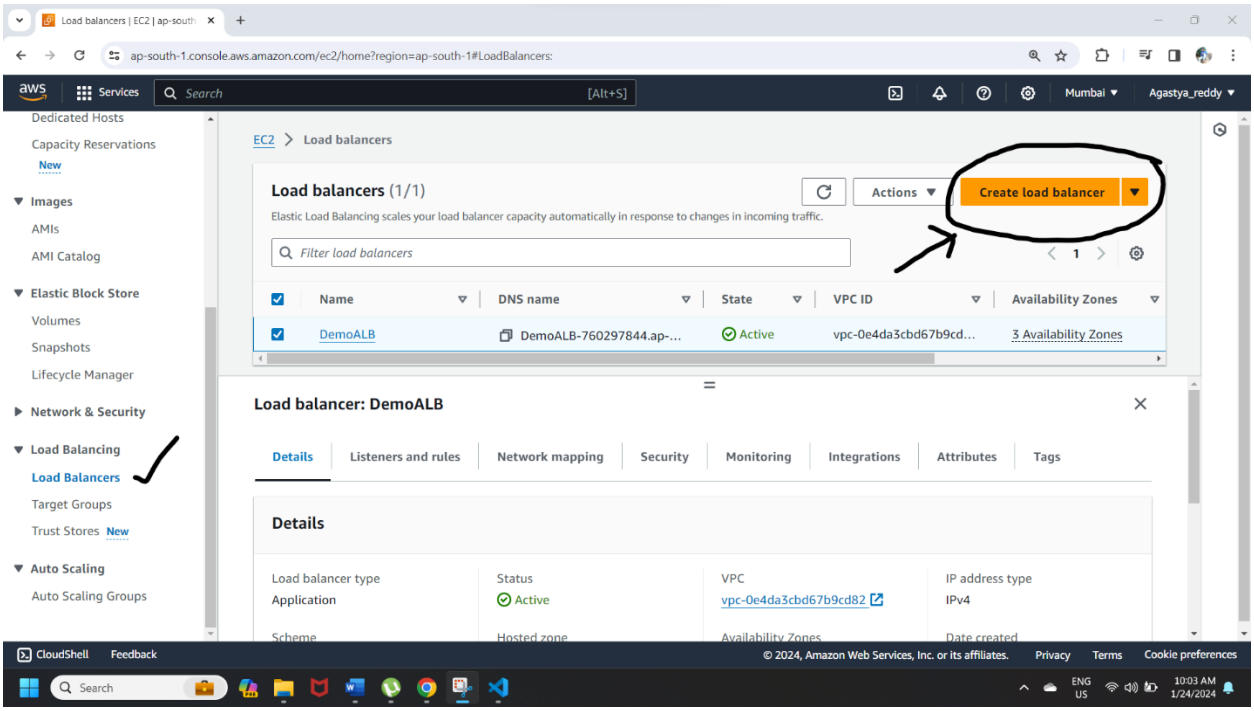
Step-7: In advanced details leave all the options but in user data you need to give some commands to it. After select the launch instance option that will create your EC2 instance within seconds. You can the code in the below image.



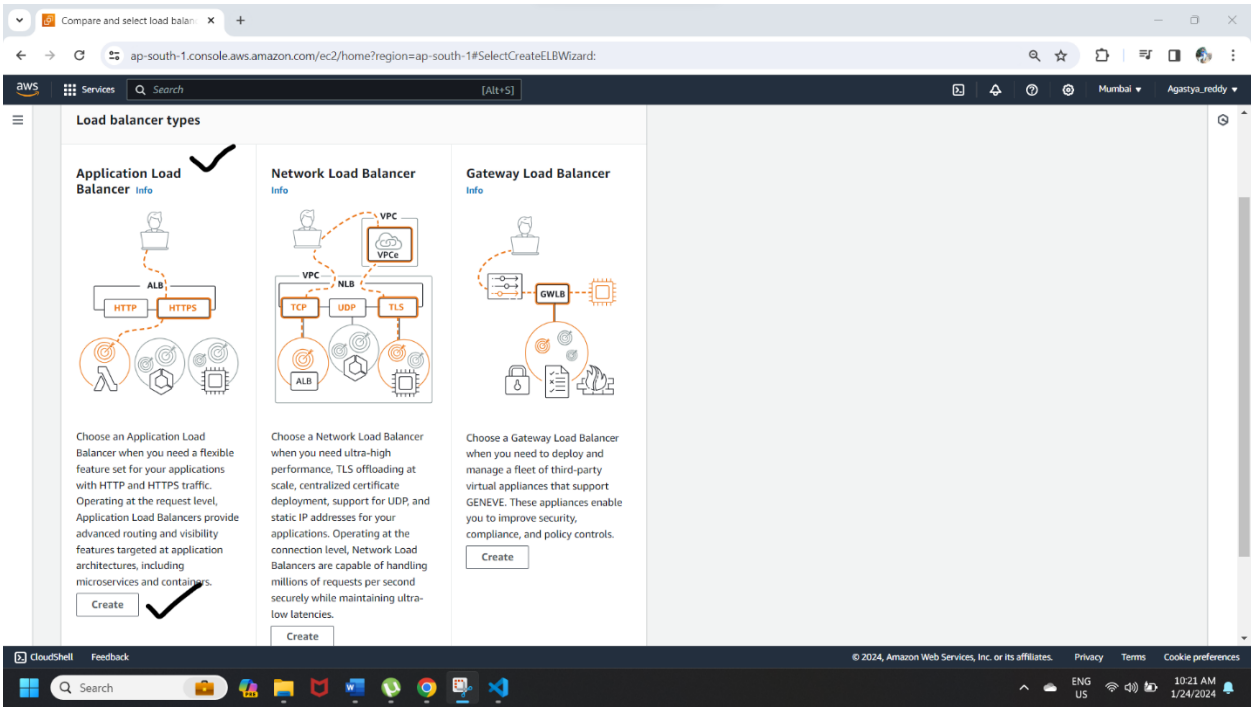
You can see that your EC2 instance state is “Running”.



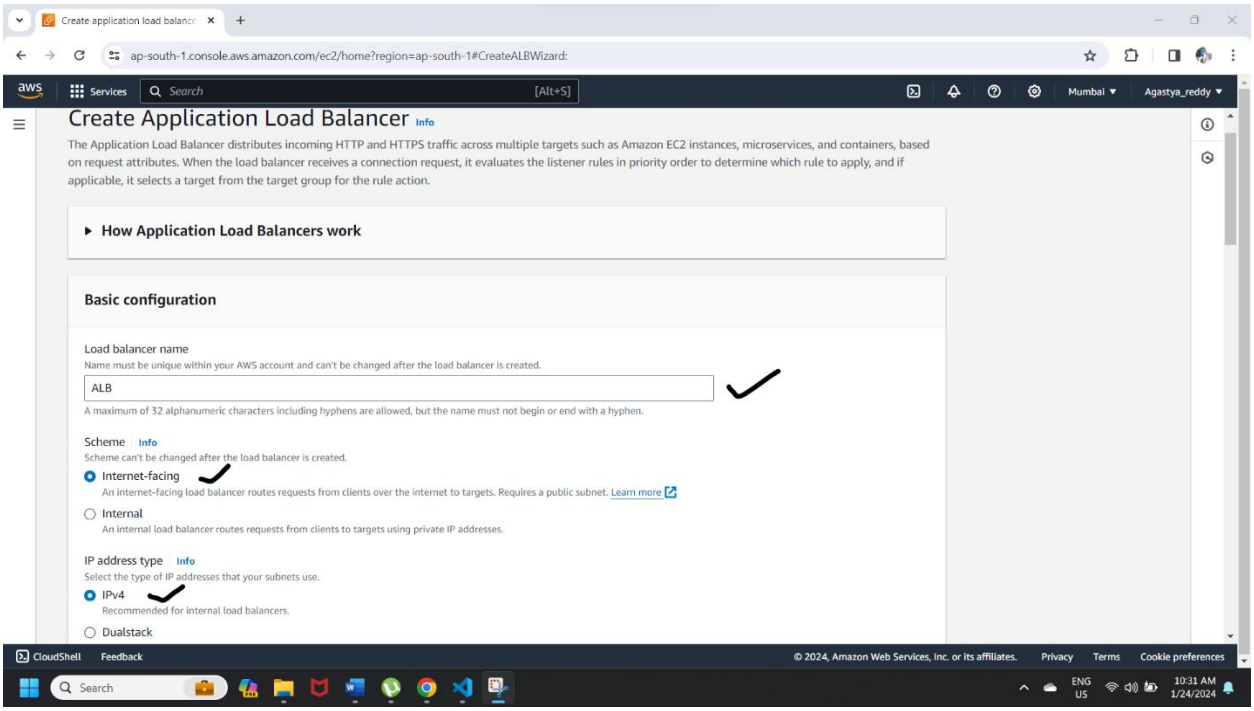
Step-8: After creating an EC2 instance, now you need to create a load balancer. Firstly, you need to select "Load Balancers" option which is on the left side and you can see the load balancers content box. There you need to select the "Create load balancer" option.



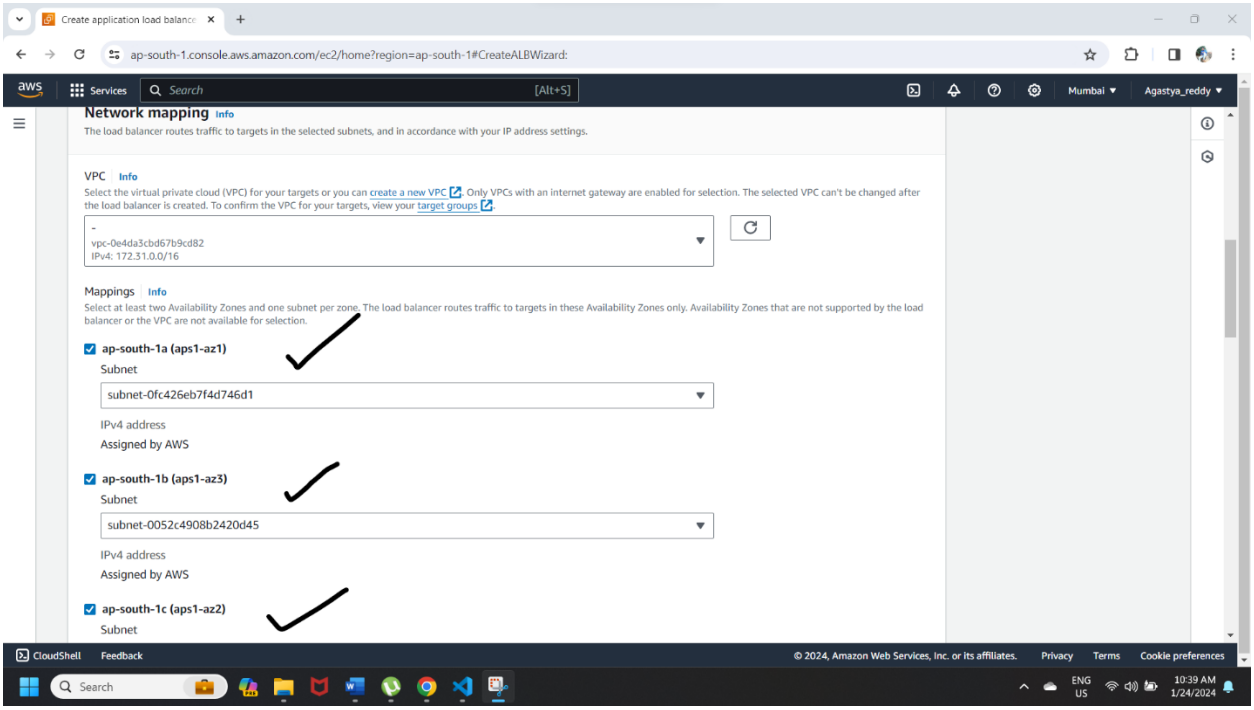
Step-9: Now you need to select the Application Load Balancer (ALB) because here you will deal with the HTTP and HTTPS traffic, and it is the best option also. Then select the "create" option.



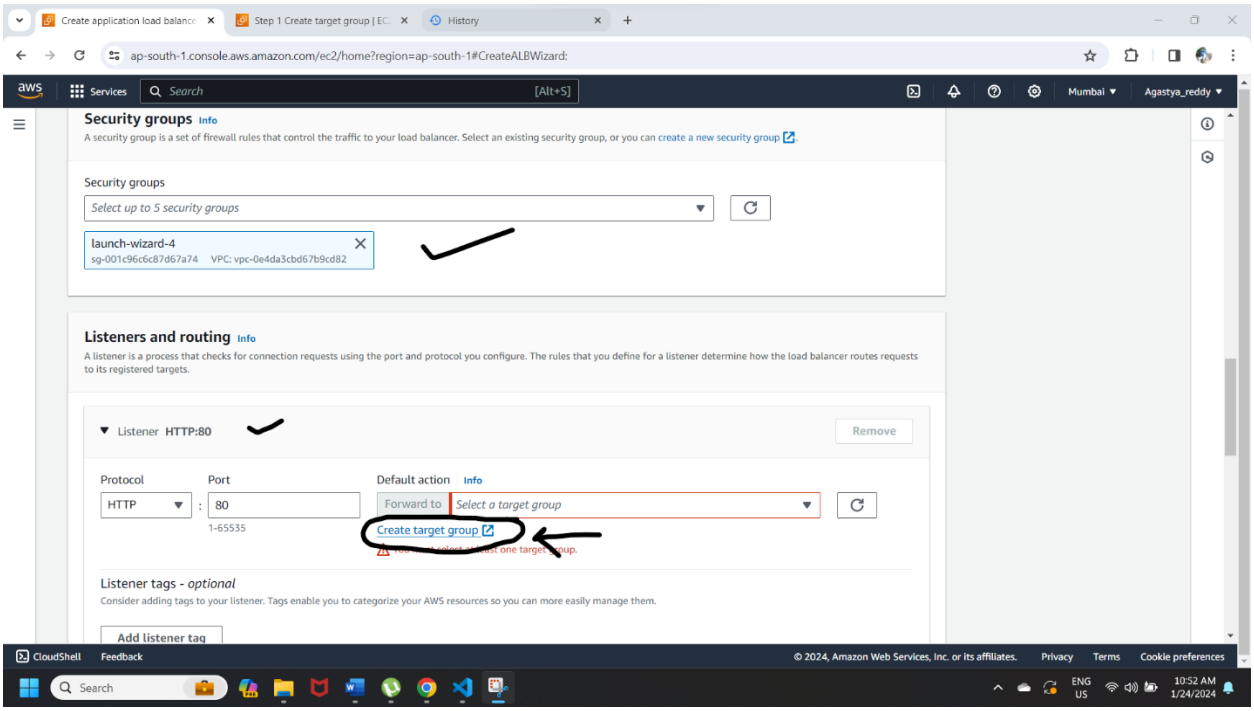
Step-10: In Basic Configuration, you need to give your load balancer name. You need to select the scheme type as “Internet-facing” and the IP address type as “IPv4”.



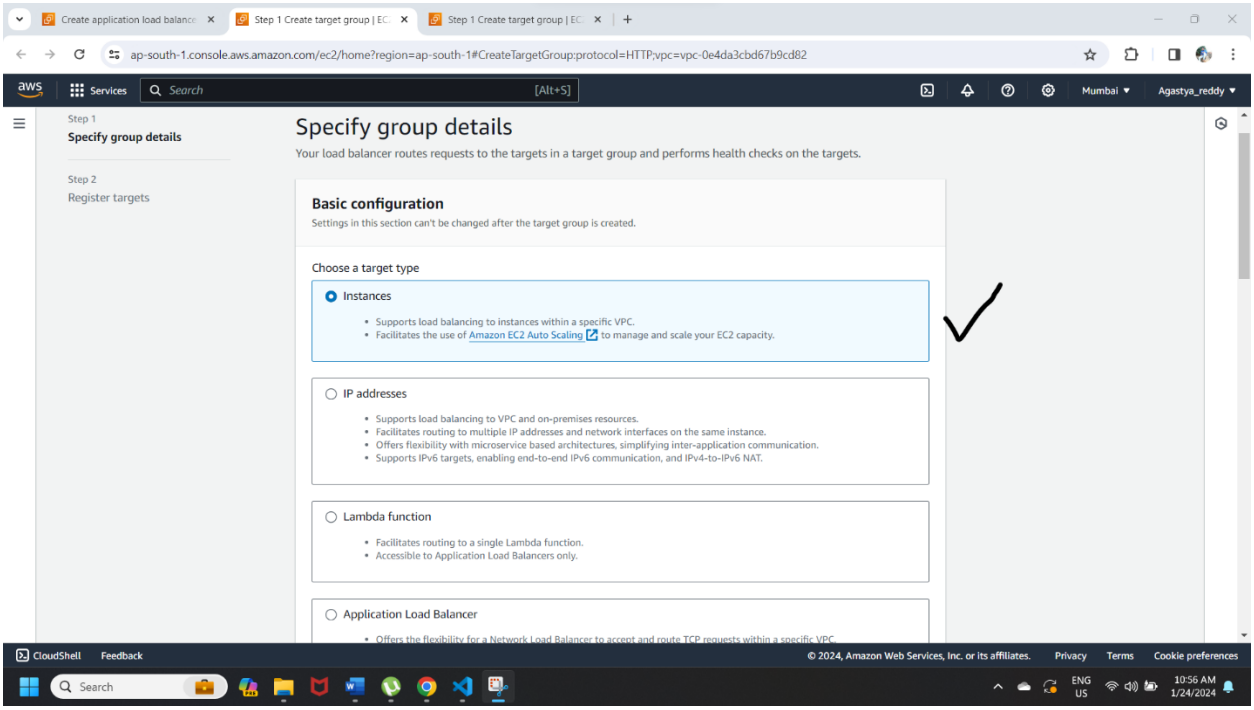
Step-11: In network mapping leave the VPC as default and select the Availability Zones at least 2 but here you need to select all three AZs.



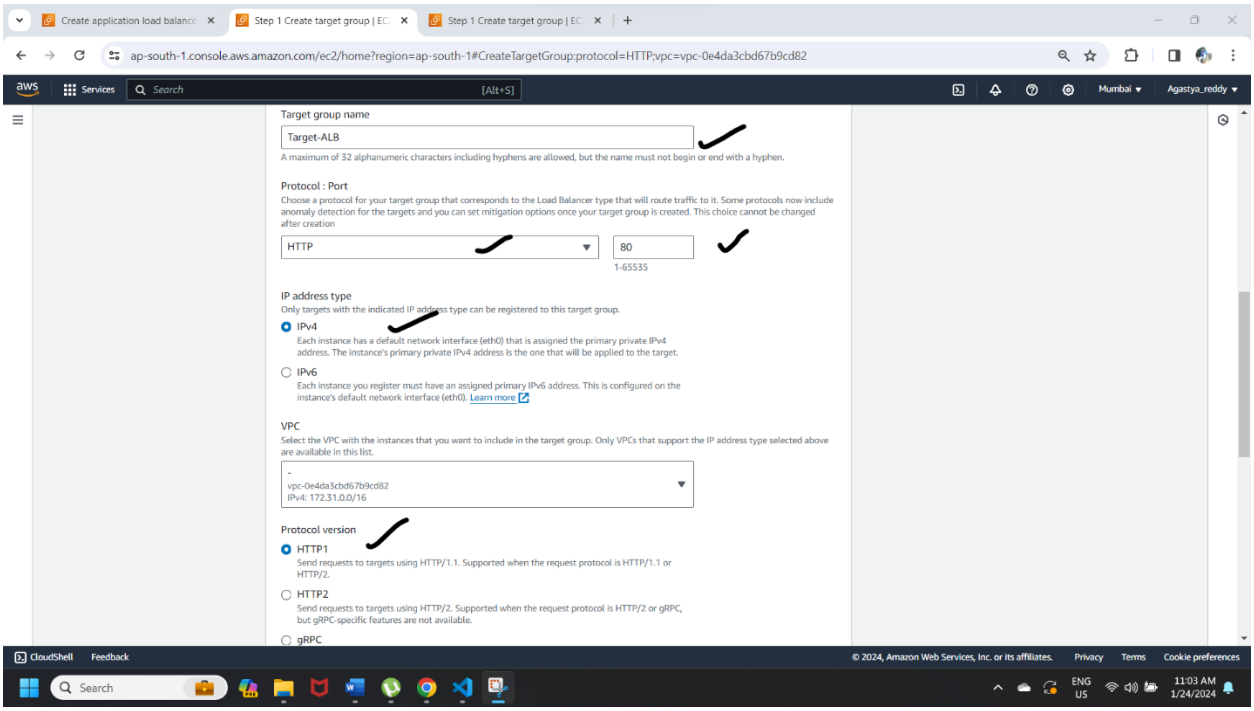
Step-12: In security groups, select the security group which you have given to your EC2 instance. And in listeners and routing select the listener as HTTP:80 where HTTP is a protocol and 80 is a port. And also you need to create a new target group for your load balancer.



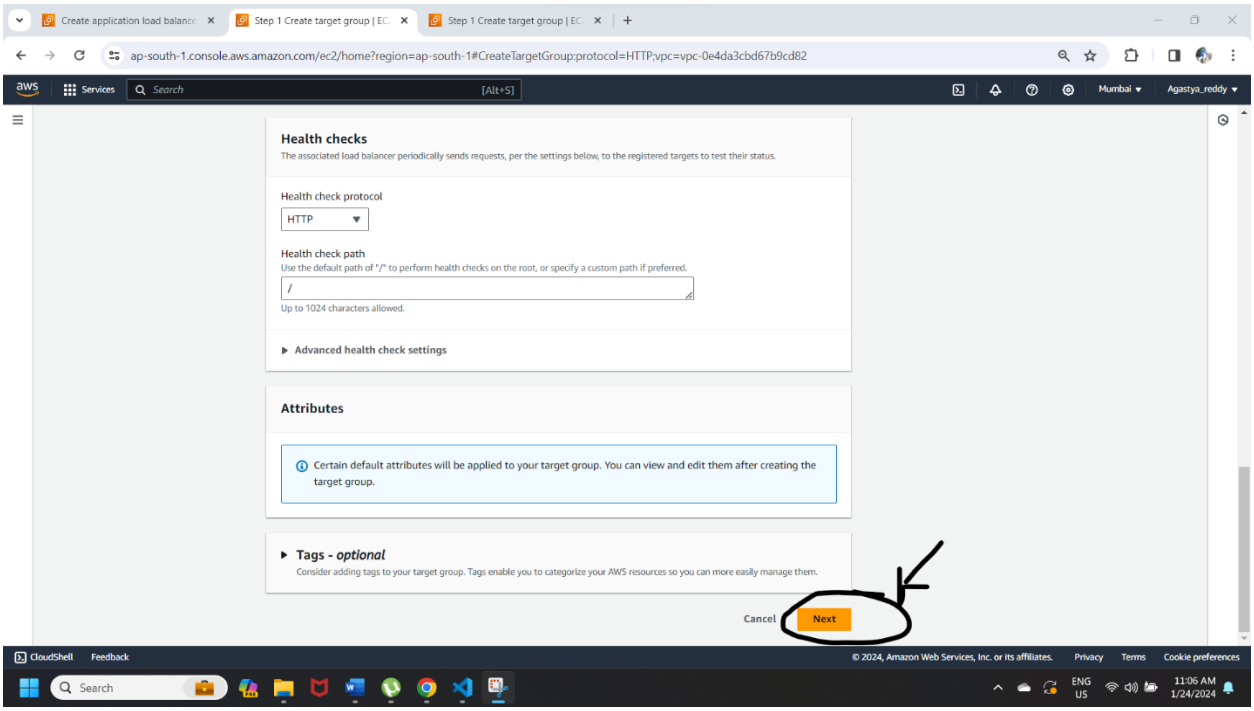
Step-13: In basic configuration, choose the target type as "Instance".



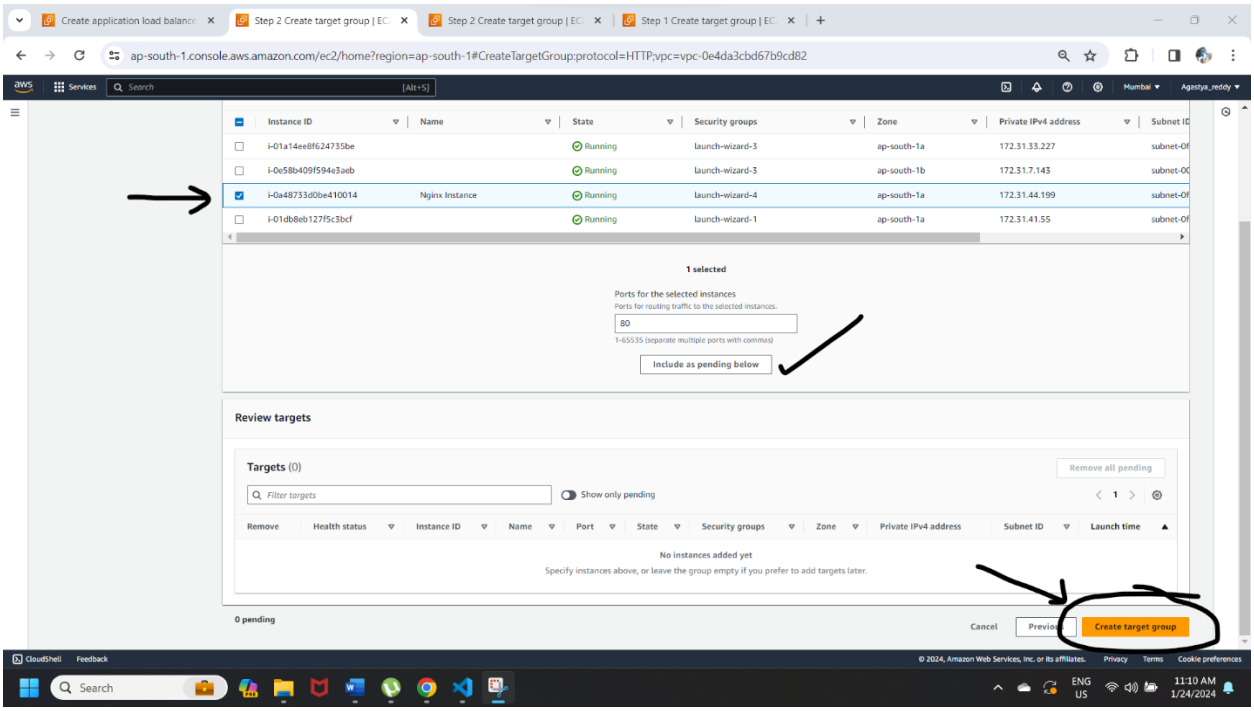
Step-14: Give the target group name and select protocol as HTTP and Port number as 80. Select IP address type as IPv4 and protocol version as HTTP1. Leave VPC as default.



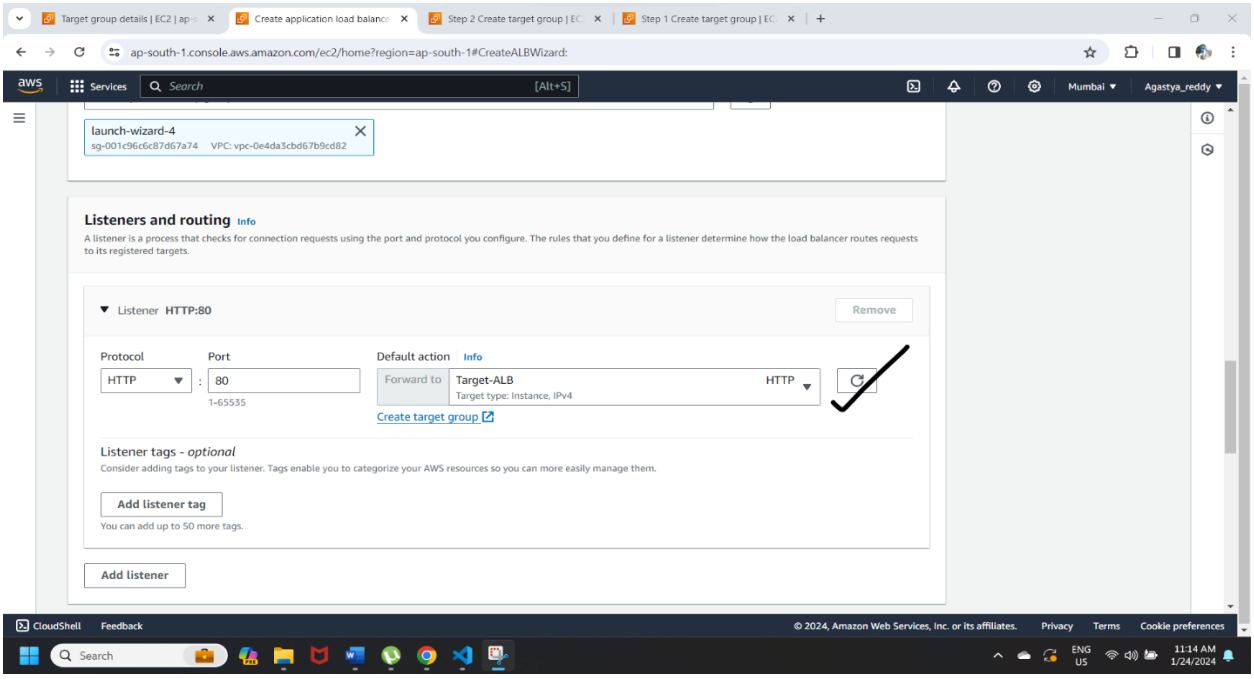
Step-15: Leave the health check as default and click on "Next".



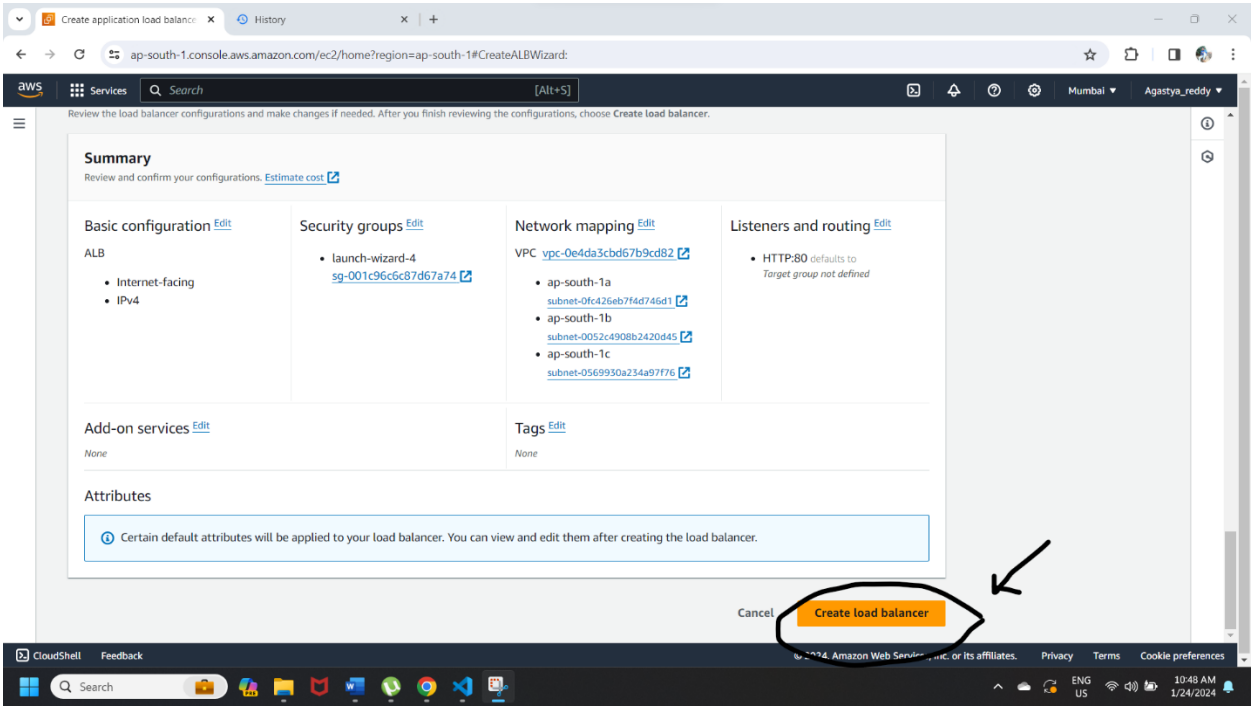
Step-16: Select the instance that you have created and select the “include as pending below” option. Then select the “create target group” option.



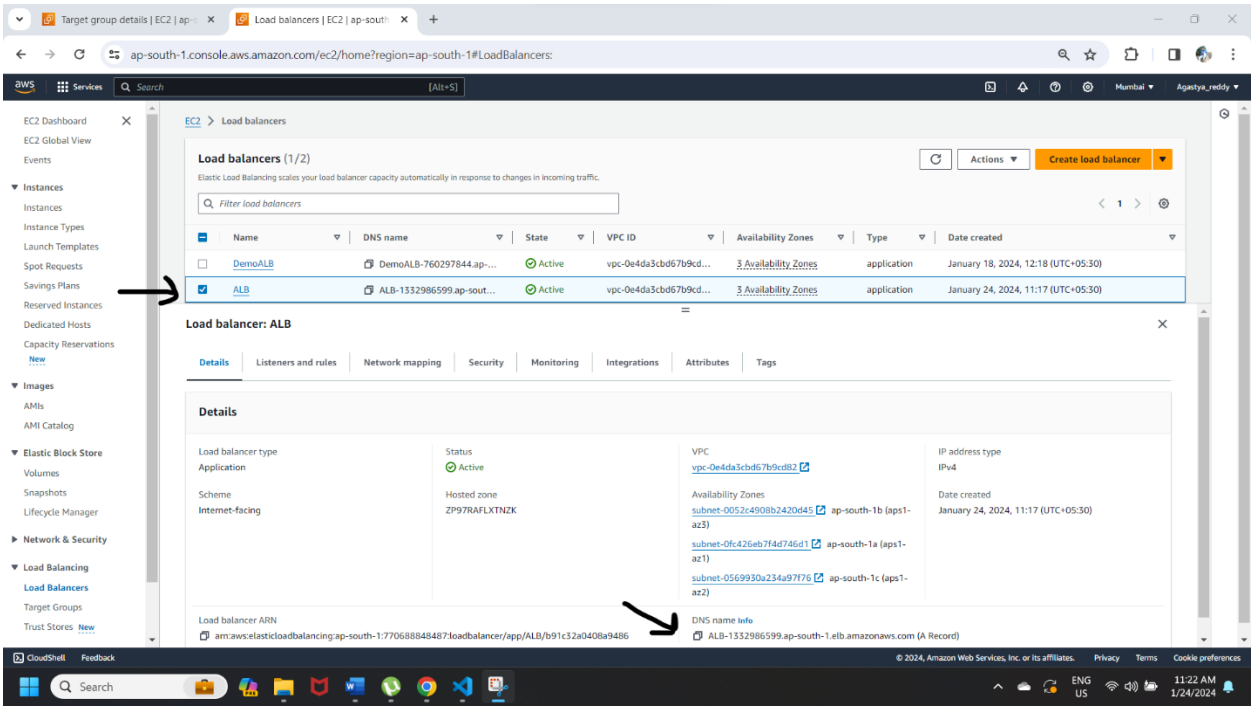
Step-17: Now come back to load balancer creation and select the target group that which you have created.



Step-18: And leave remaining options and select the create “load balancer option”.

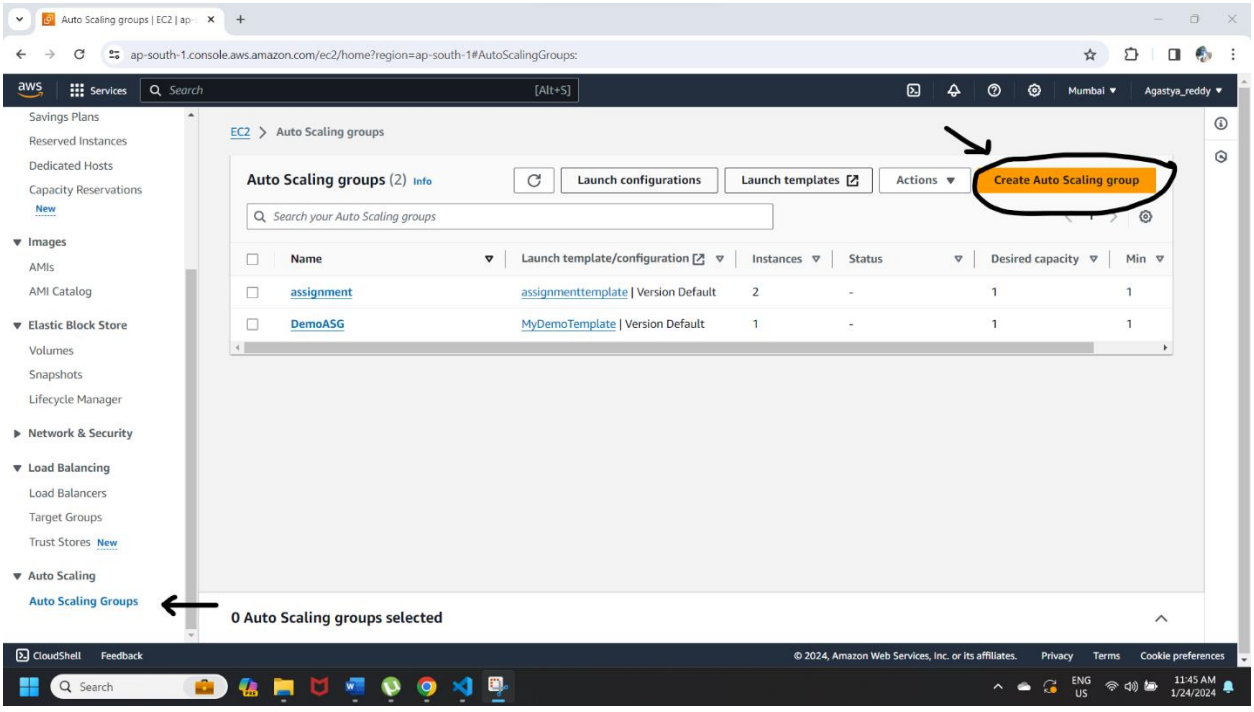


Step-19: Now you can see your load balancer is in an active state. By clicking on the details of load balancer, you need to copy the DNS-name.

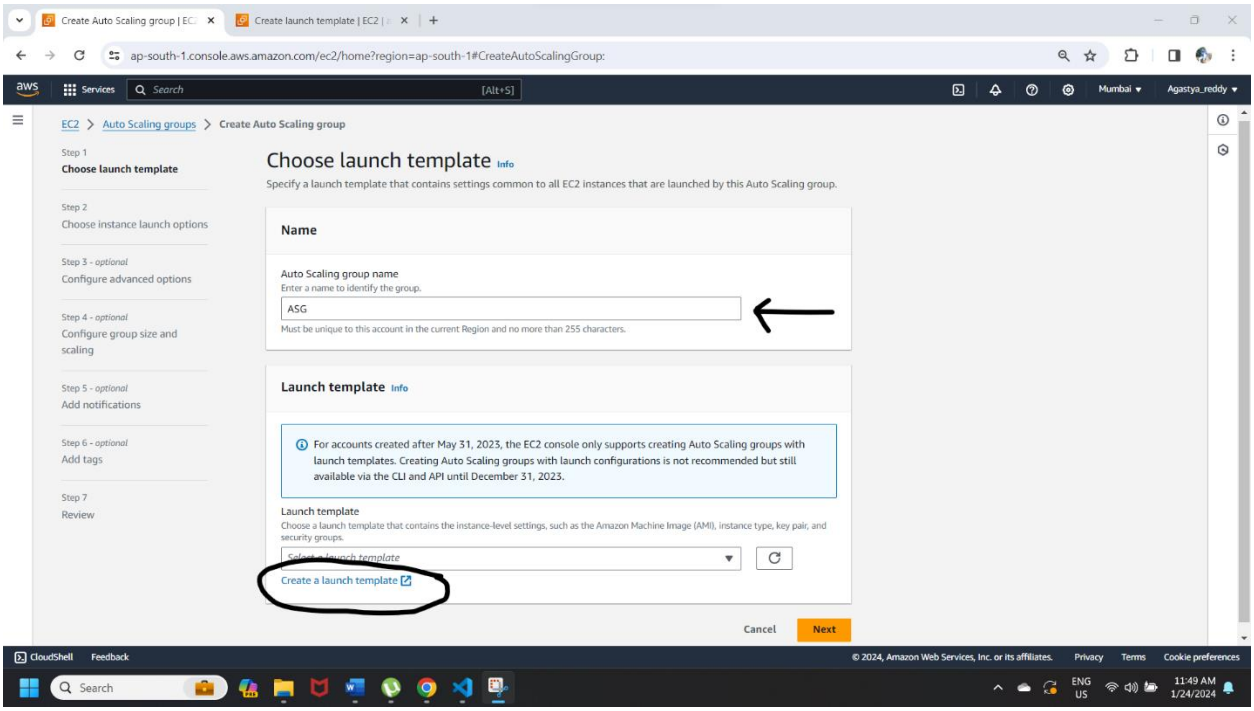


After copying the DNS-name, paste it on the new tab in your browser and search that link. Then you need to get output as below figure. If not, please very the EC2 instance you have created.

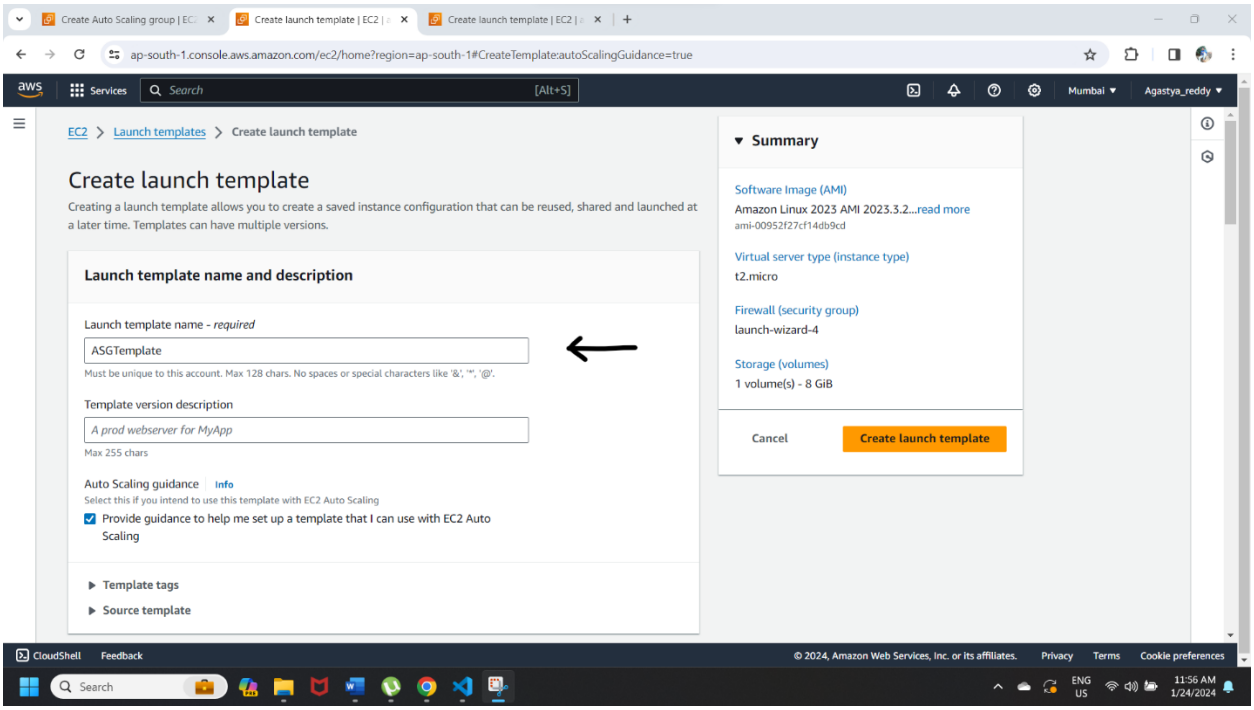
Step-20: Now select the Auto Scaling Group on your EC2 instance. And select the "Create Auto Scaling Group" option.



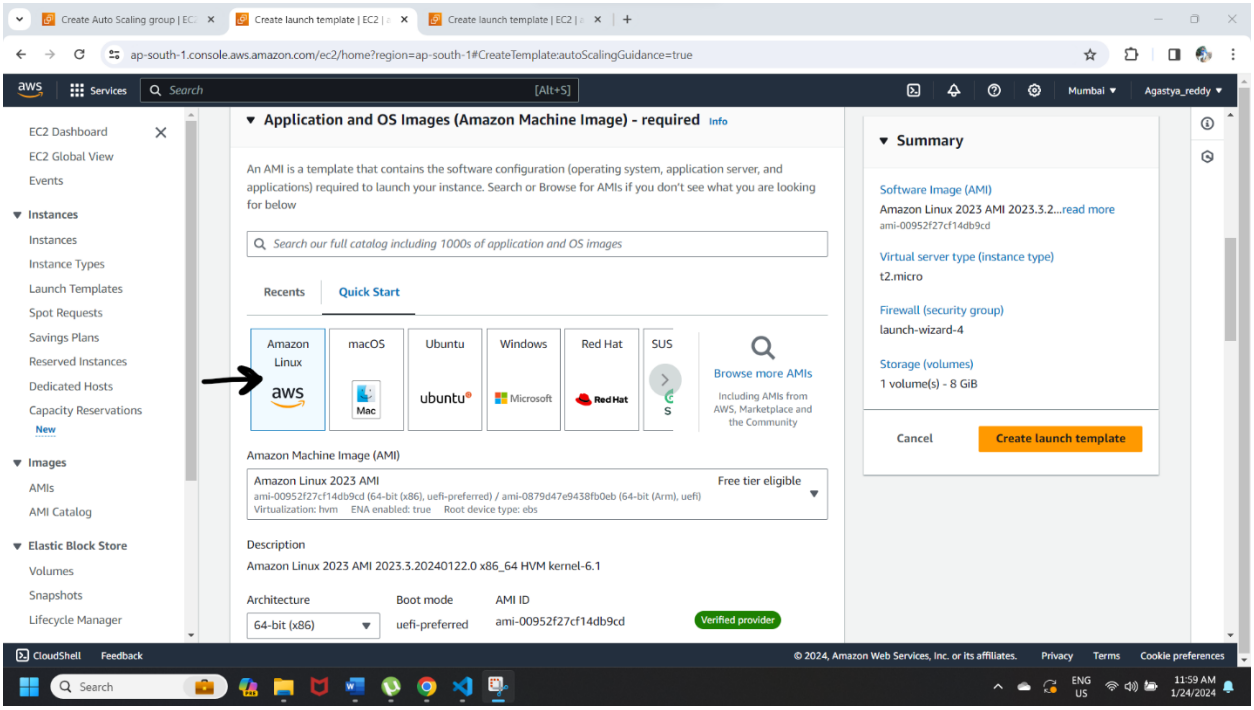
Step-21: In “Choose launch template” enter the name for auto scaling group. And select the “Create a launch template”.



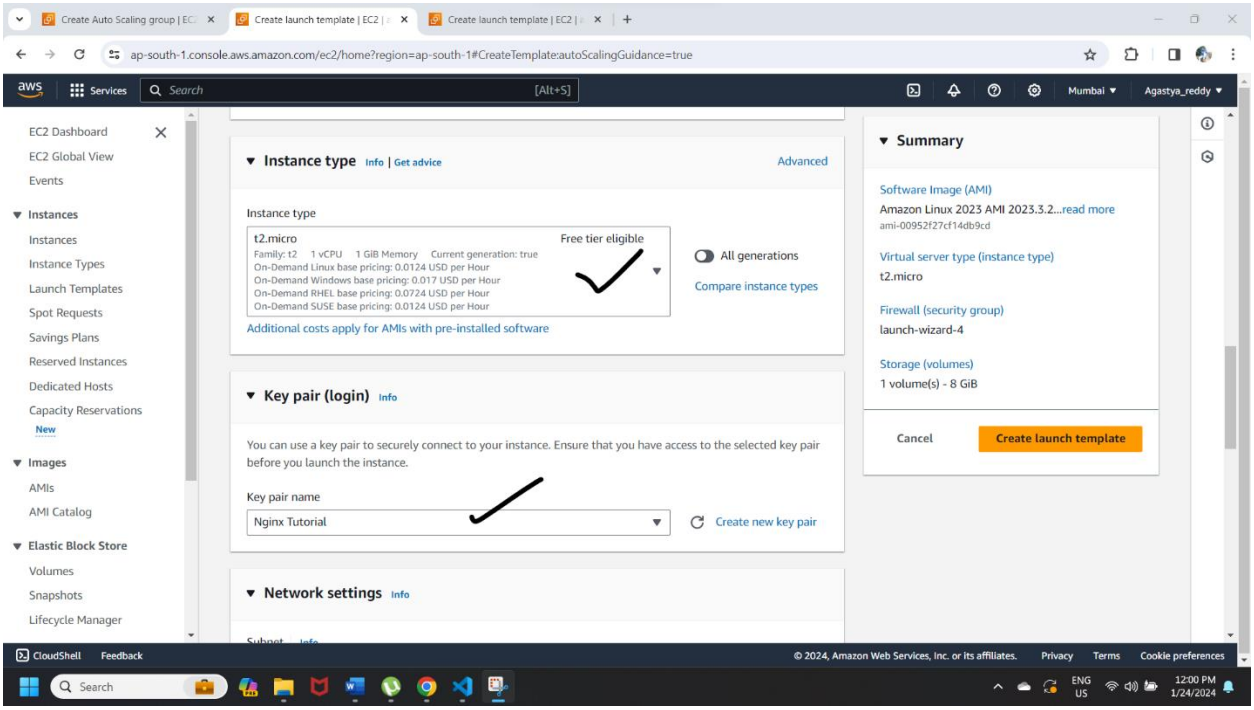
Step-22: In Launch template name and description, you need to give the launch template name.



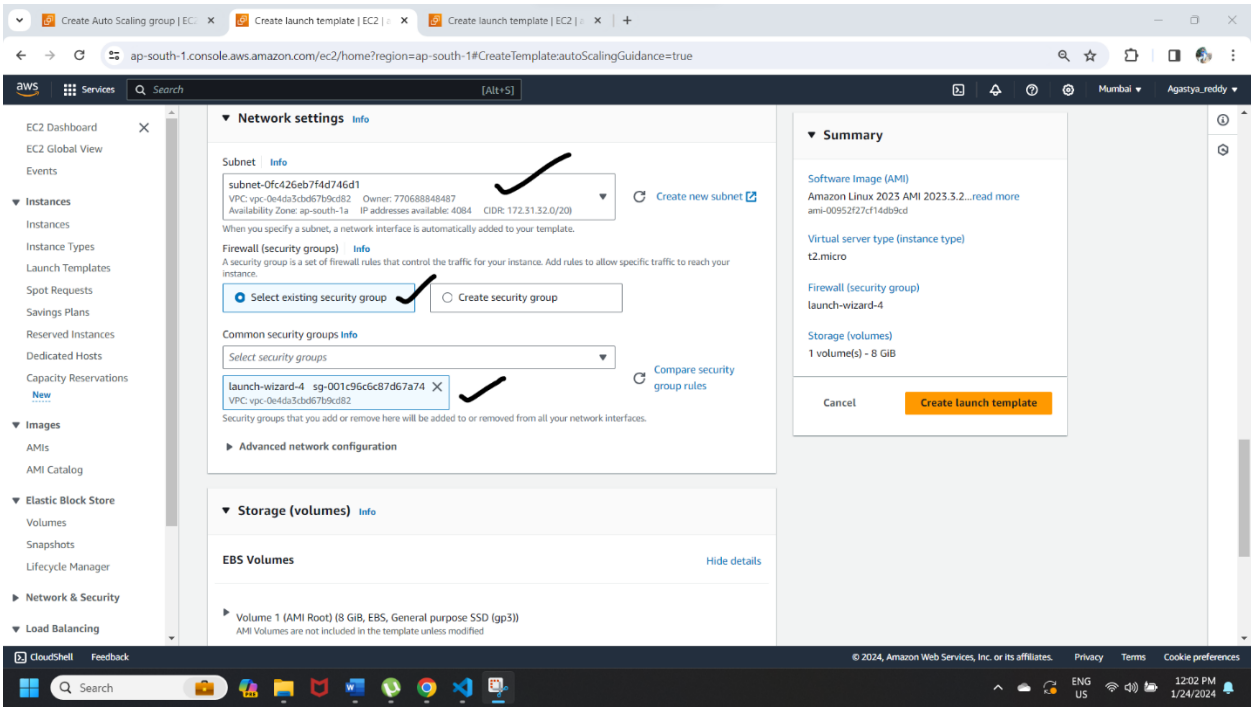
Step-23: Here you need to select Amazon Machine Image as Amazon Linux 2023 AMI.



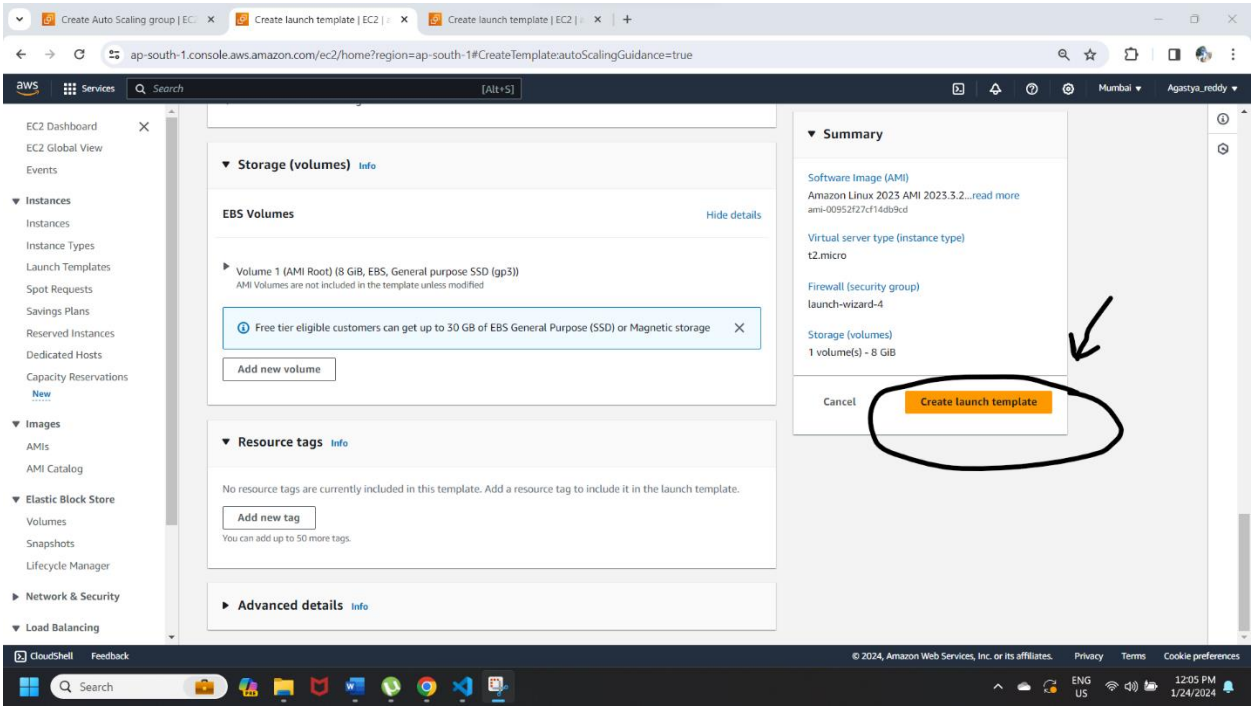
Step-24: Select the instance type as t2.micro and key pair as Nginx Tutorial.



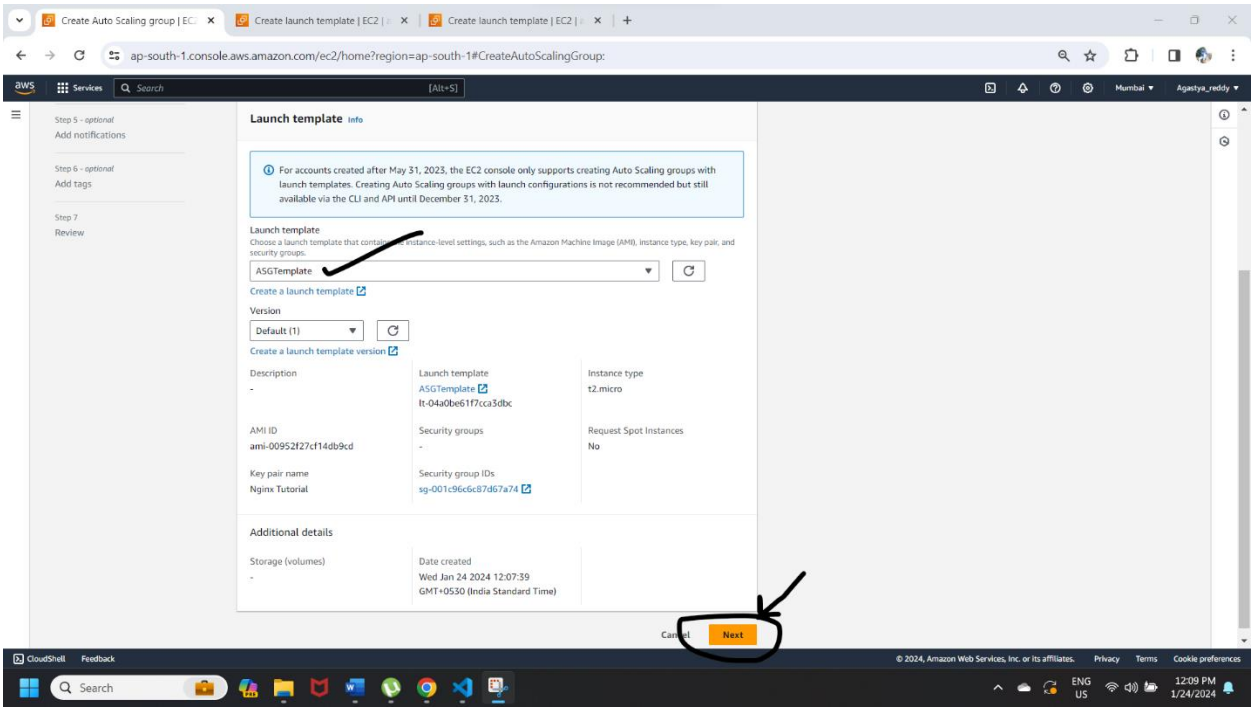
Step-25: Do not select the subnet leave it as empty (ignore in the fig). Then select the "existing security group" Launch-wizard-4 which is also same for the EC2 you have created.



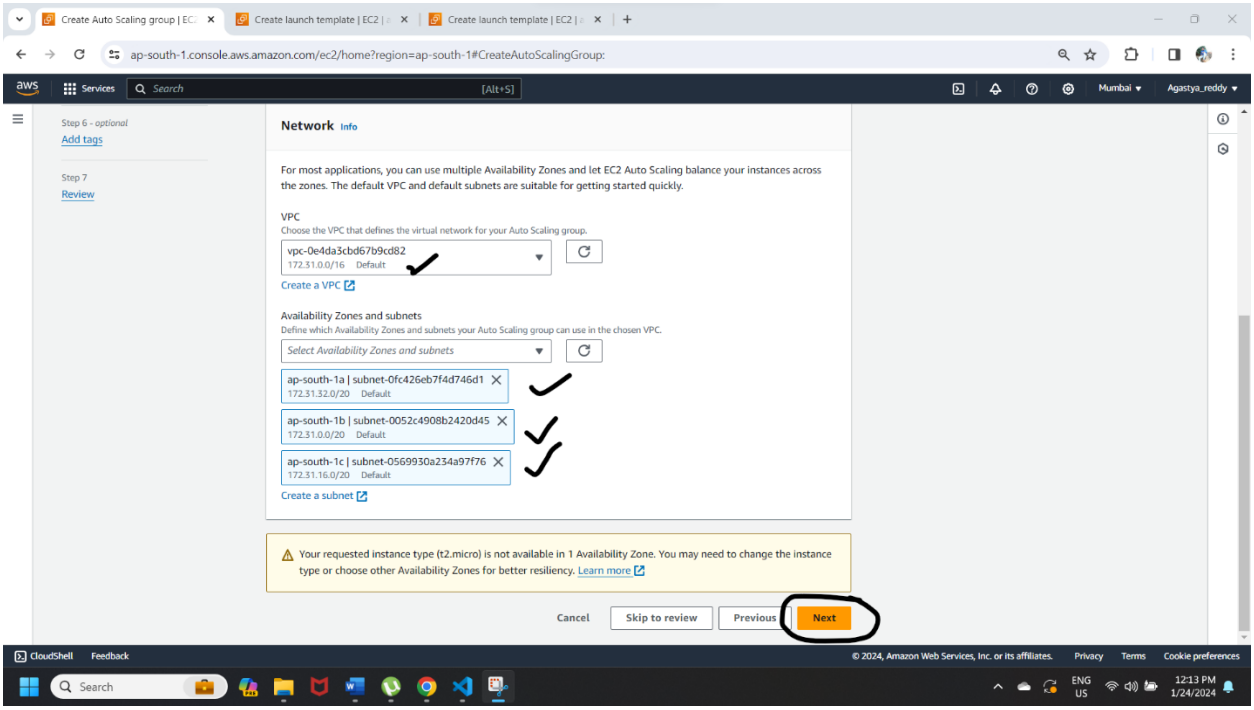
Step-26: Leave the remaining all as default and click on the create launch template. Then the launch template will be created.



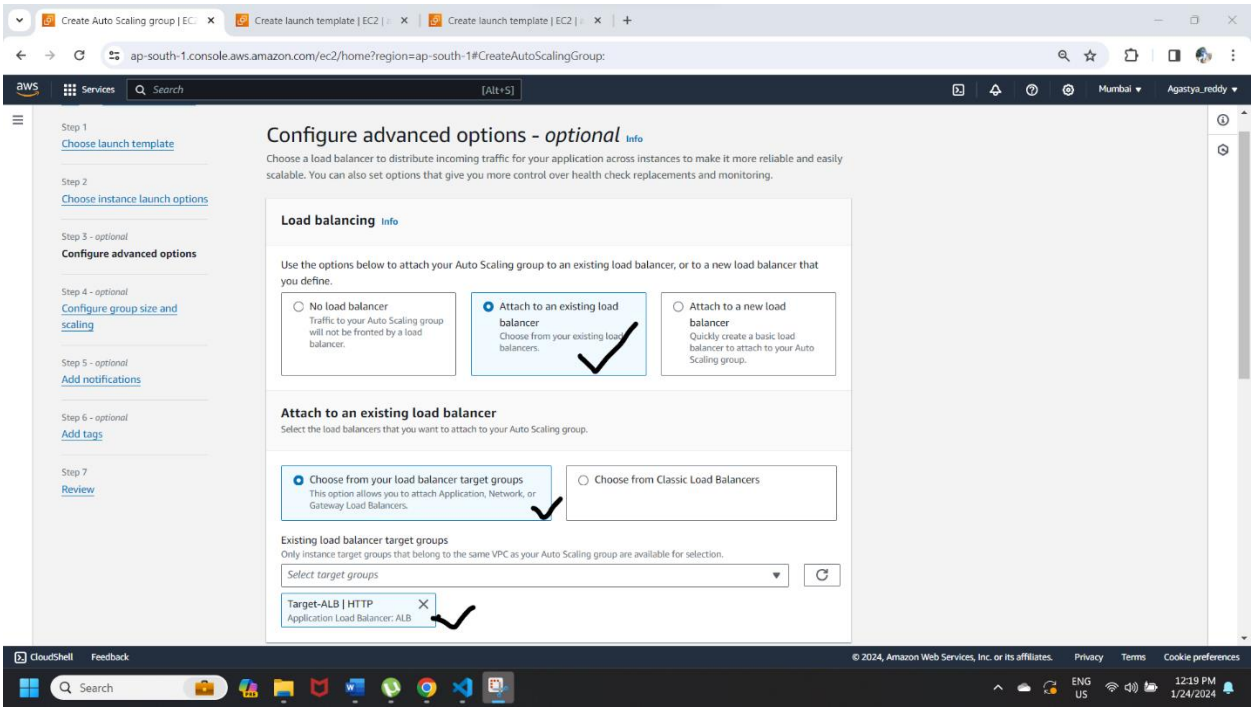
Step-27: Now come back to the launch template and select the launch template that you have created. And leave the version as default (1). And click on the next button.



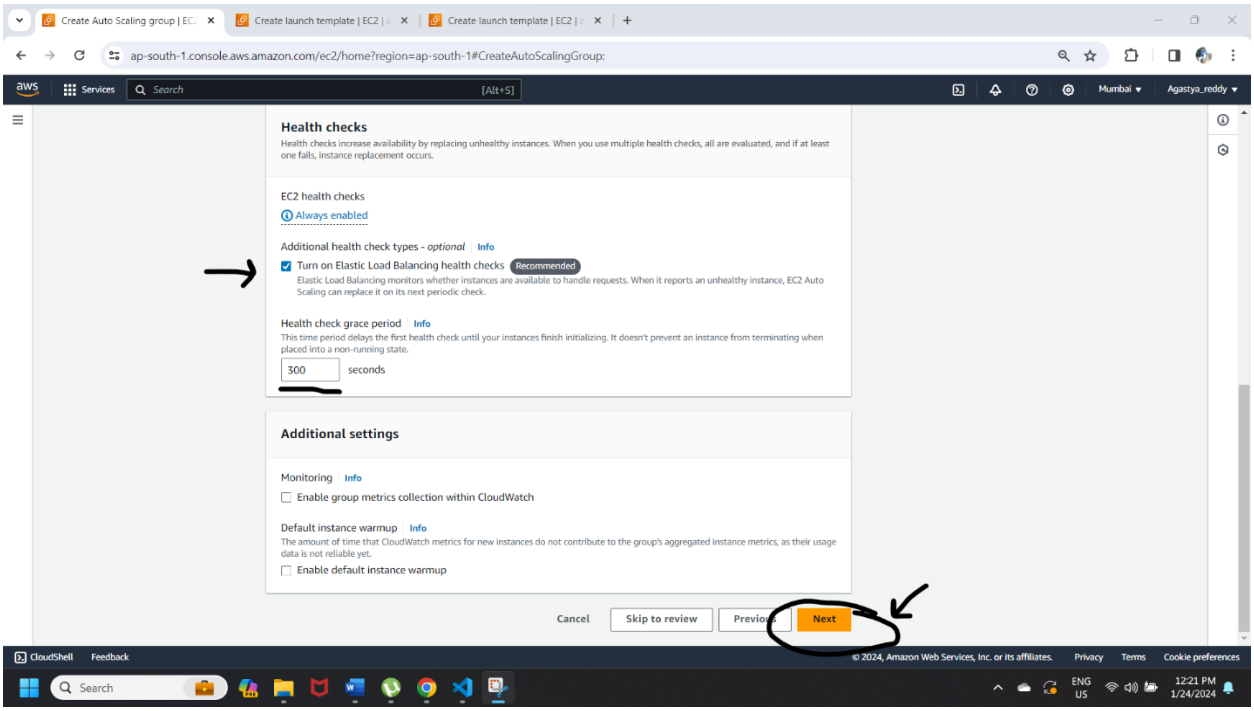
Step-28: In choose instance launch options, in network select all the three availability zones and leave VPS as default. Then click on the next button.



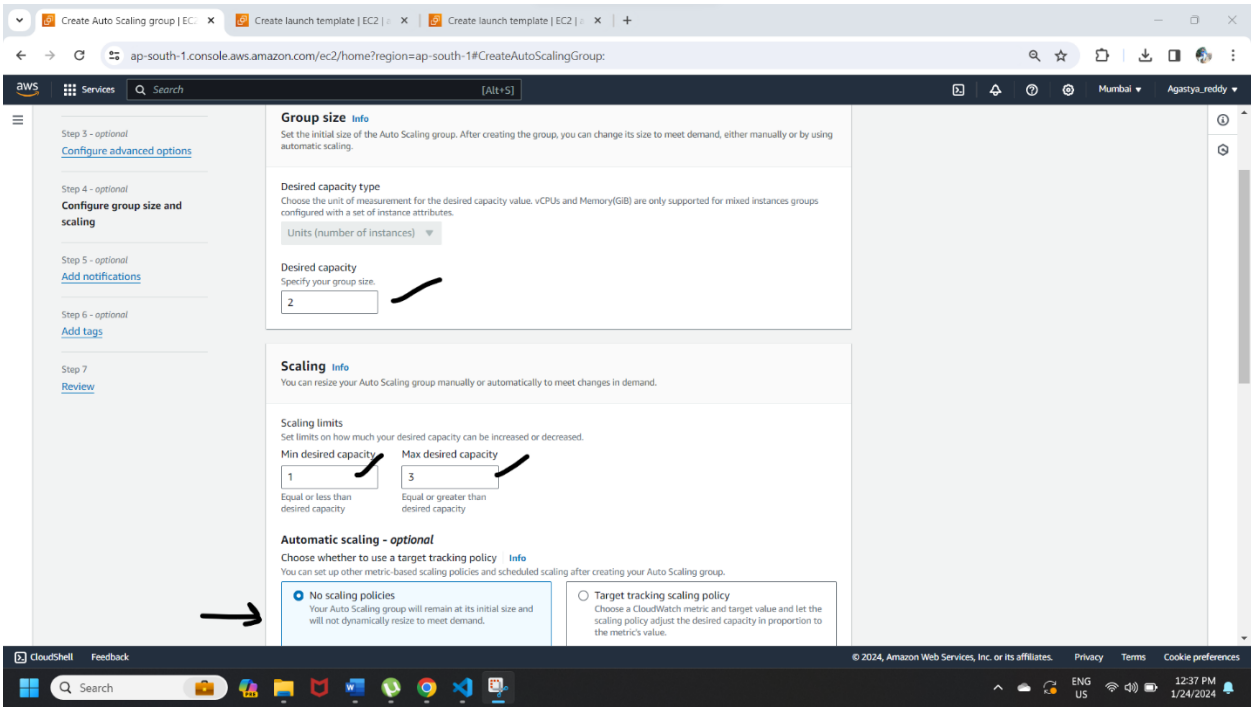
Step-29: In configure advanced option step, select load balancing option as “attach to an existing load balancer’ and choose the target group that you have created by selecting the “choose from your load balancers target group” option.



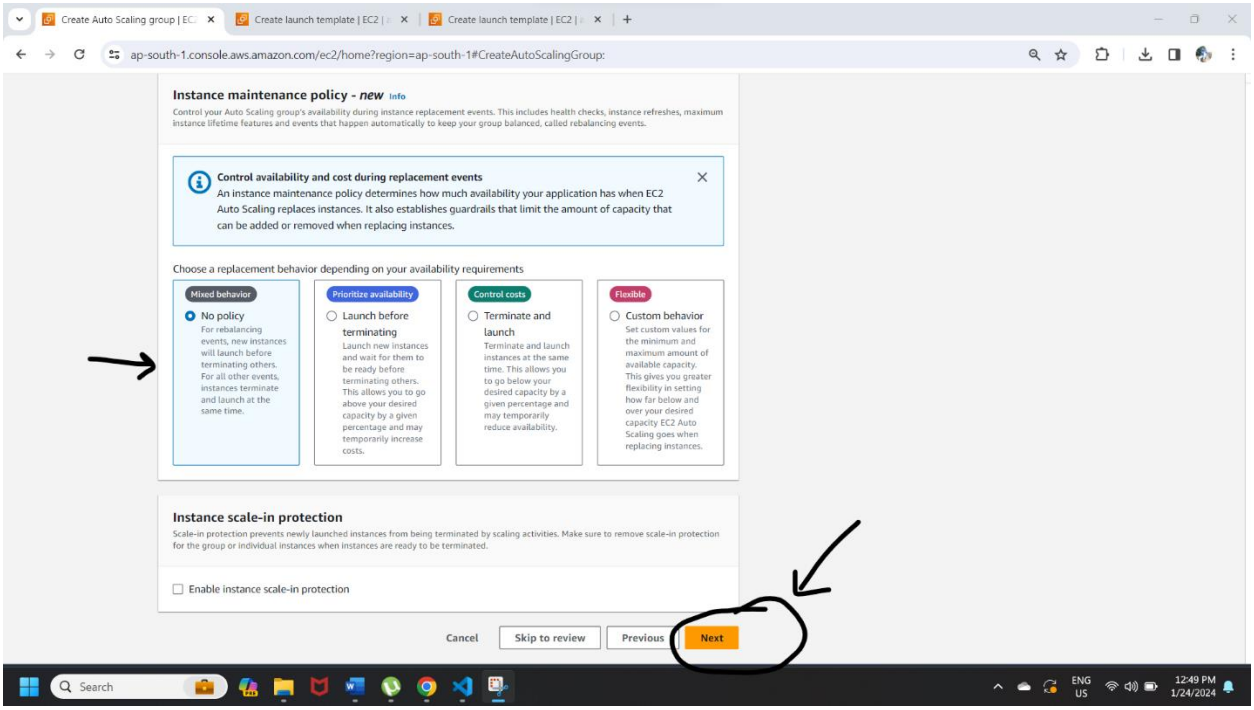
Step-30: Now turn on Elastic Load Balancing health checks and set health check grace period to 300 seconds. Next leave other options and click on the next button.



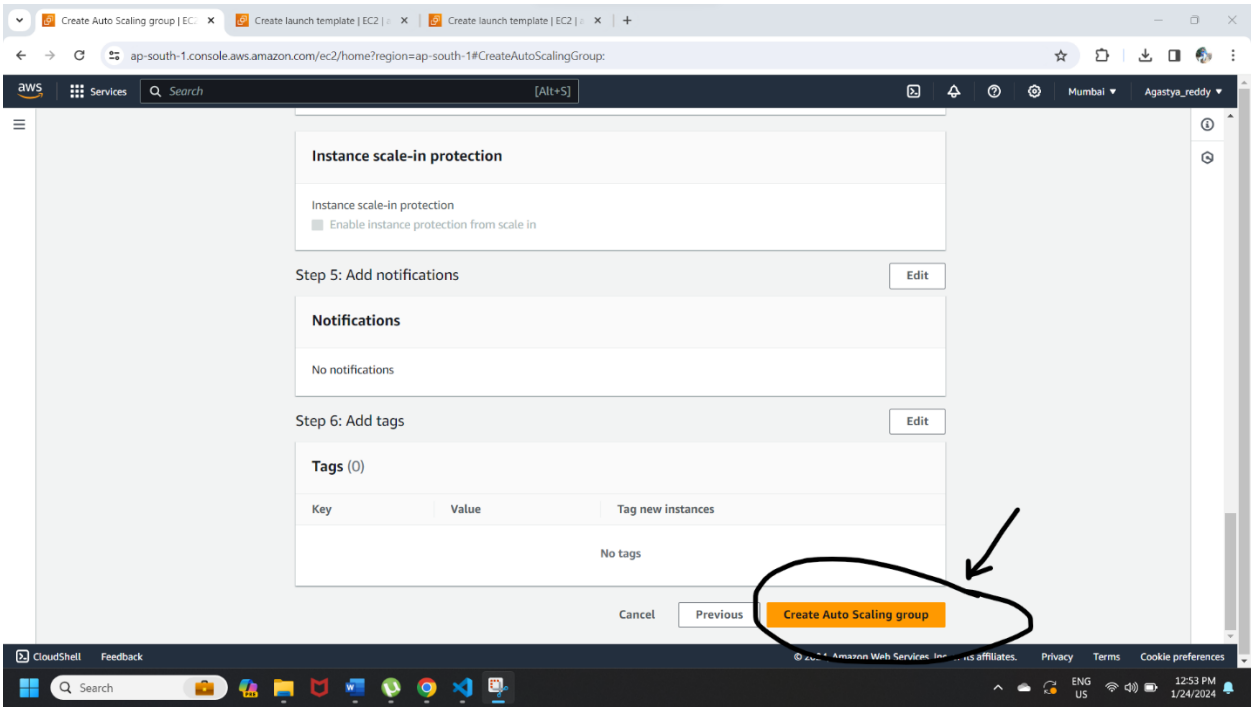
Step-31: Now you need to mention group size as how many instances need to be created. The Minimum desired capacity is 1, desired capacity is 2 and maximum desired capacity is 3. And select the automatic scaling to “No scaling policies”.



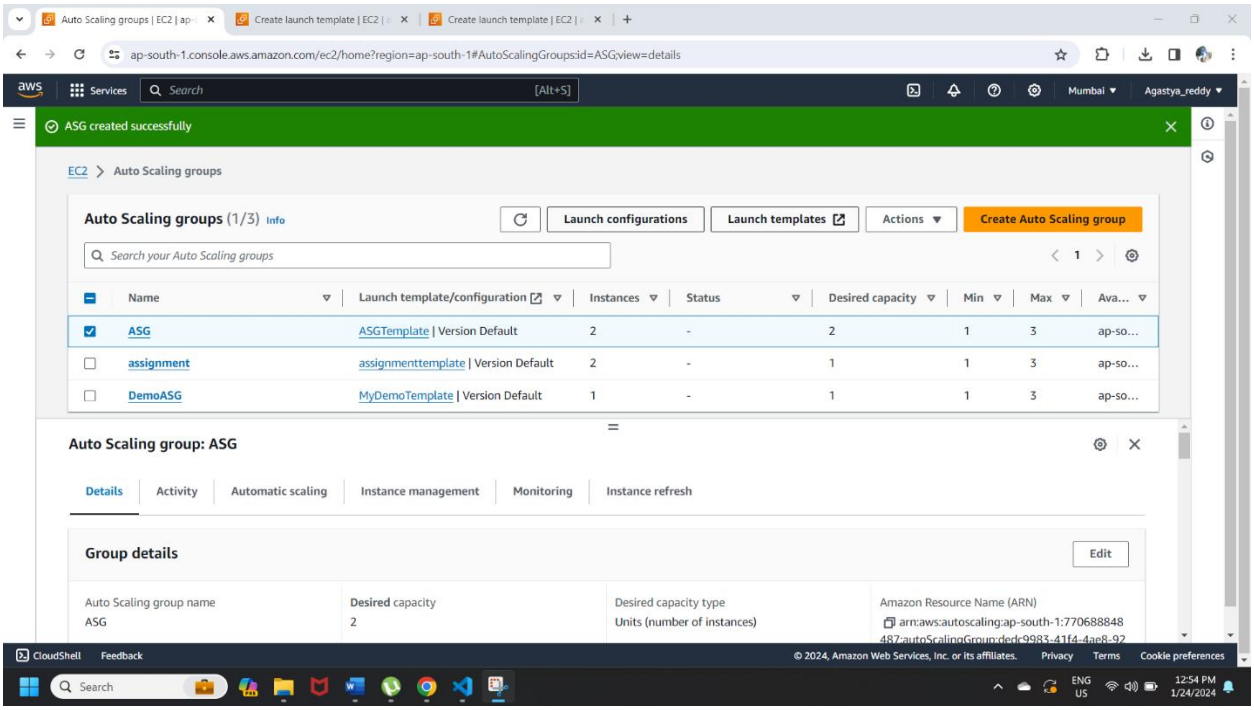
Step-32: Next in Instance maintenance policy select the “No policy” option.



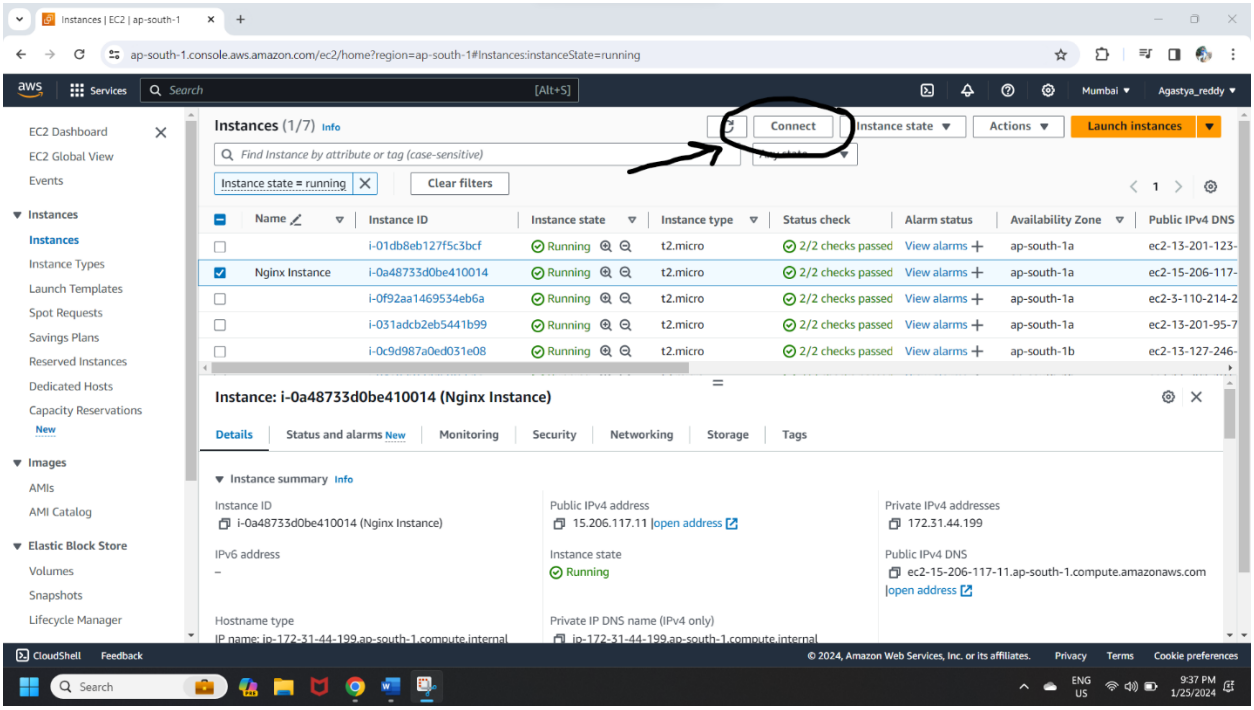
Step-33: Next steps notifications and tags leave it as empty and in the review, step click on “Create Auto Scale Group”.



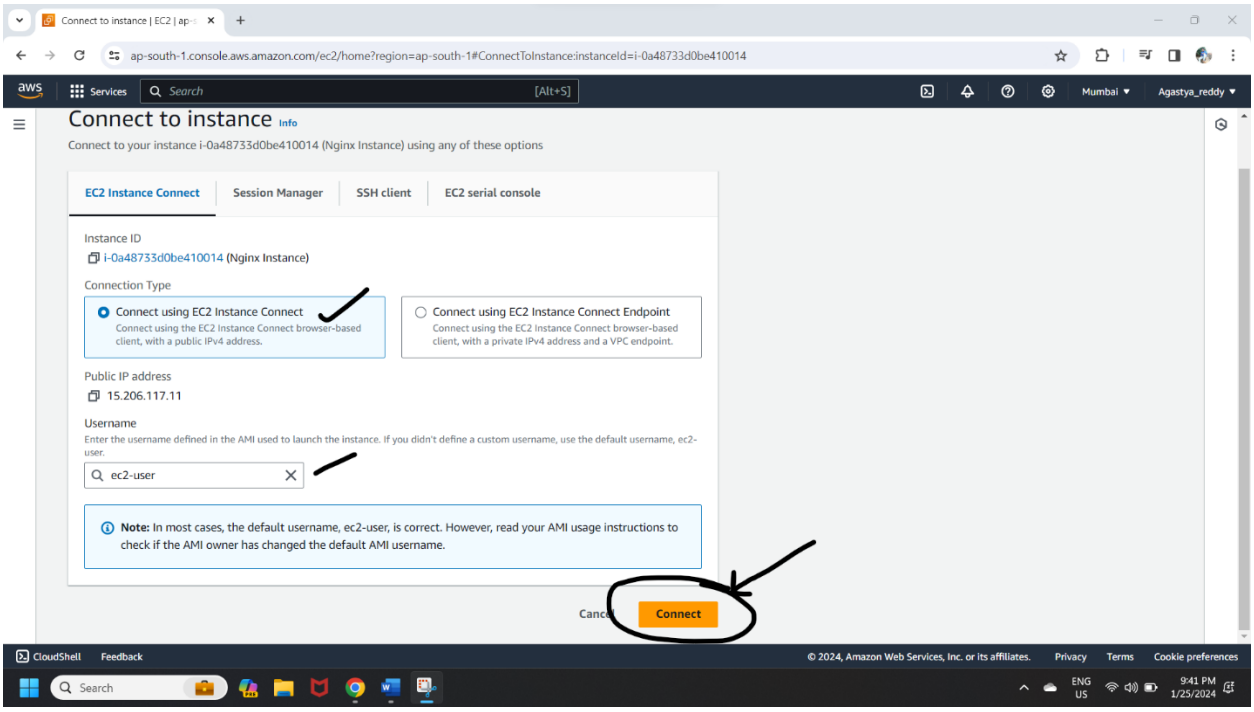
Step-34: Now you can see that you have successfully created an Auto Scaling Group (ASG).



Step-35: Now go to instances and select Nginx Instance. After clicking on the connect button.

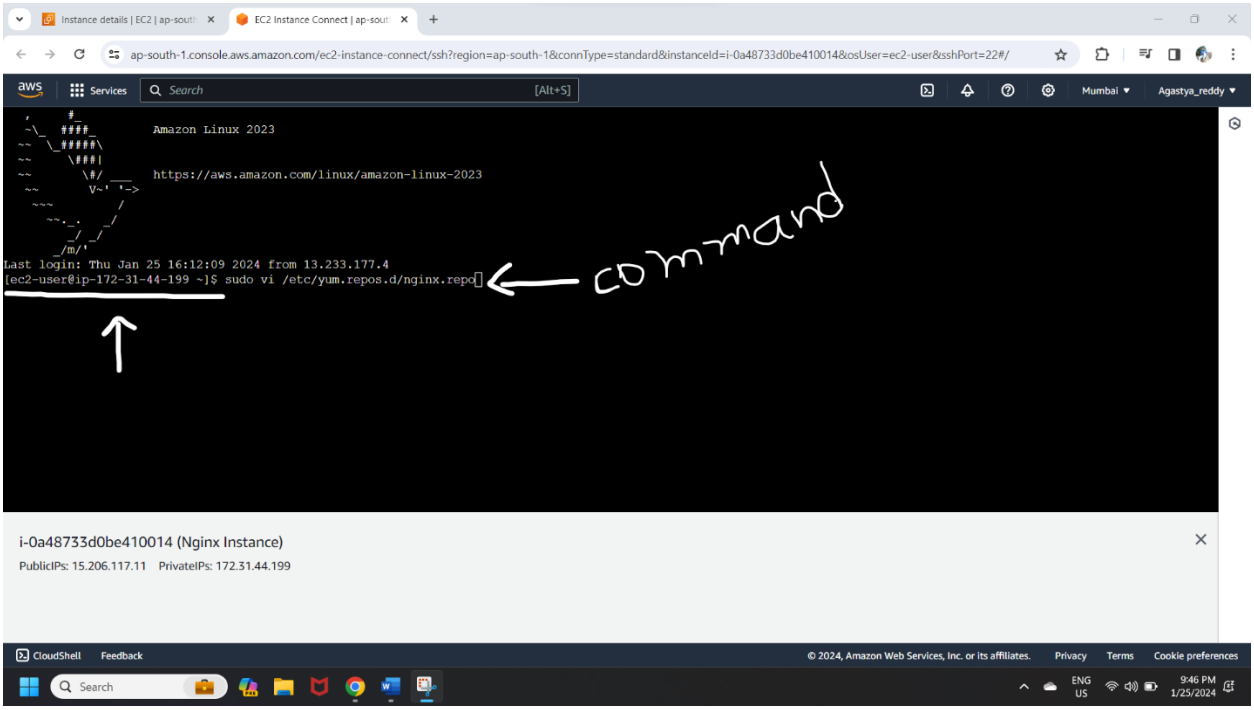


Step-36: Now you can see the option “connect to instance”. Then choose the connection type as “connect using EC2 instance connect”. Leave the username as default i.e. ec2-user. Then click on Connect.

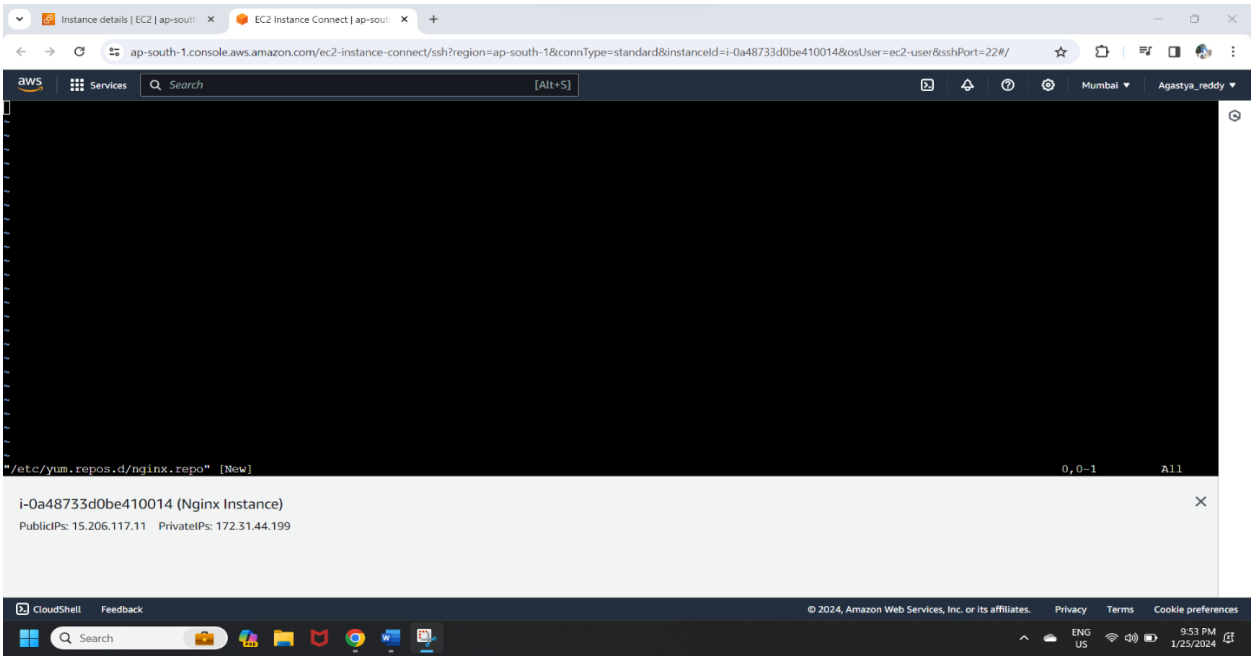


Step-37: After that you will get output displaying the “Amazon Linux 2023” and the login details. Now you need to right command after the **$.** Here you are going to so first step of nginx server installation i.e. you are going to right the command below.

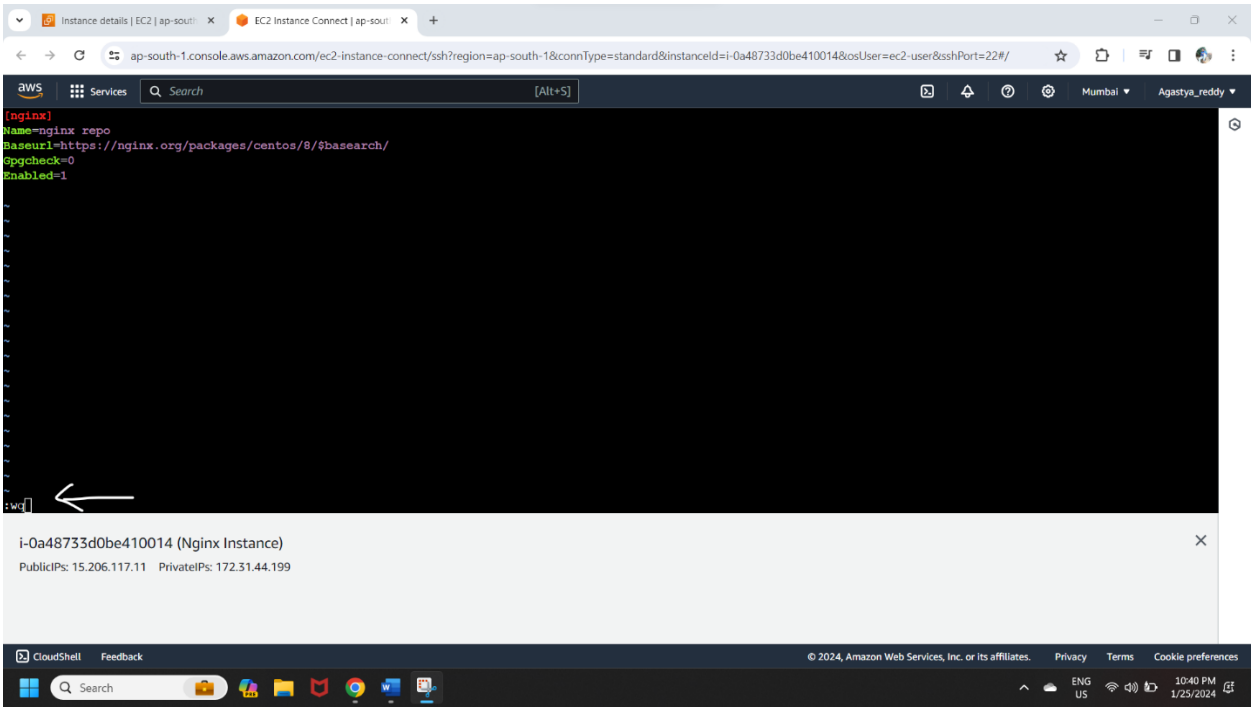
Sudo vi /etc/yum.repos.d/nginx.repo



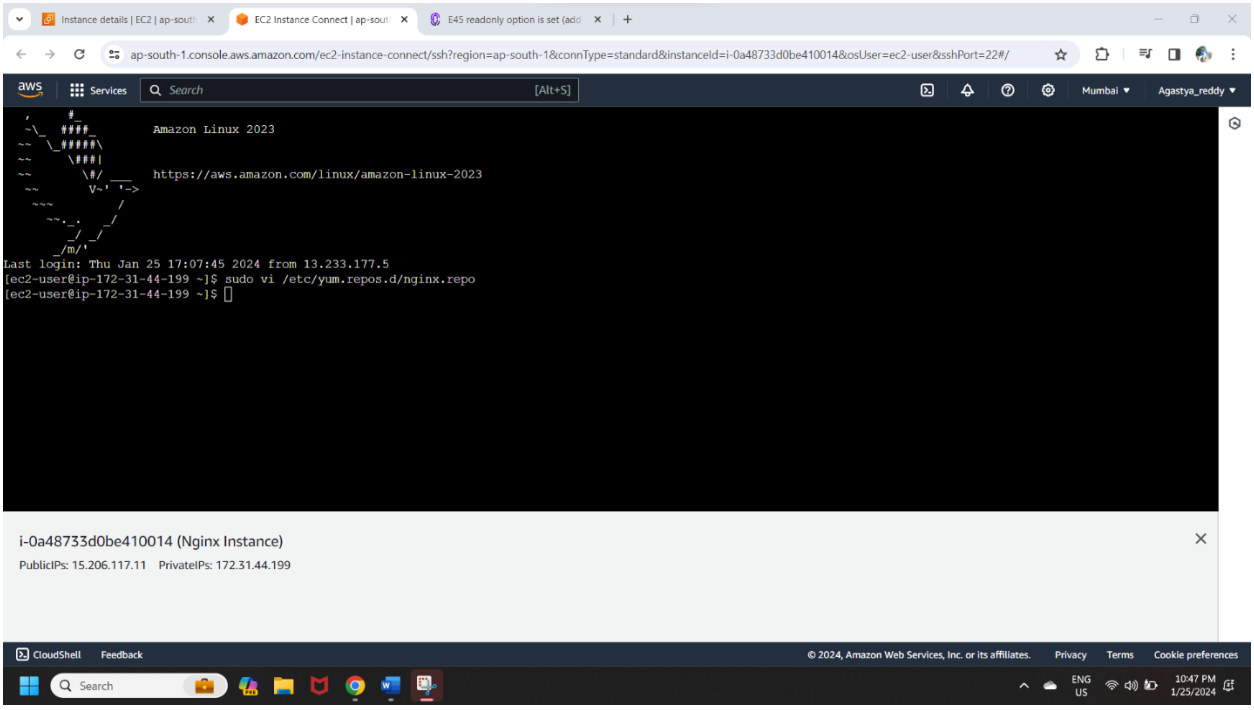
If the command is executed in the cloud shell, then you get the output that shown in below image.



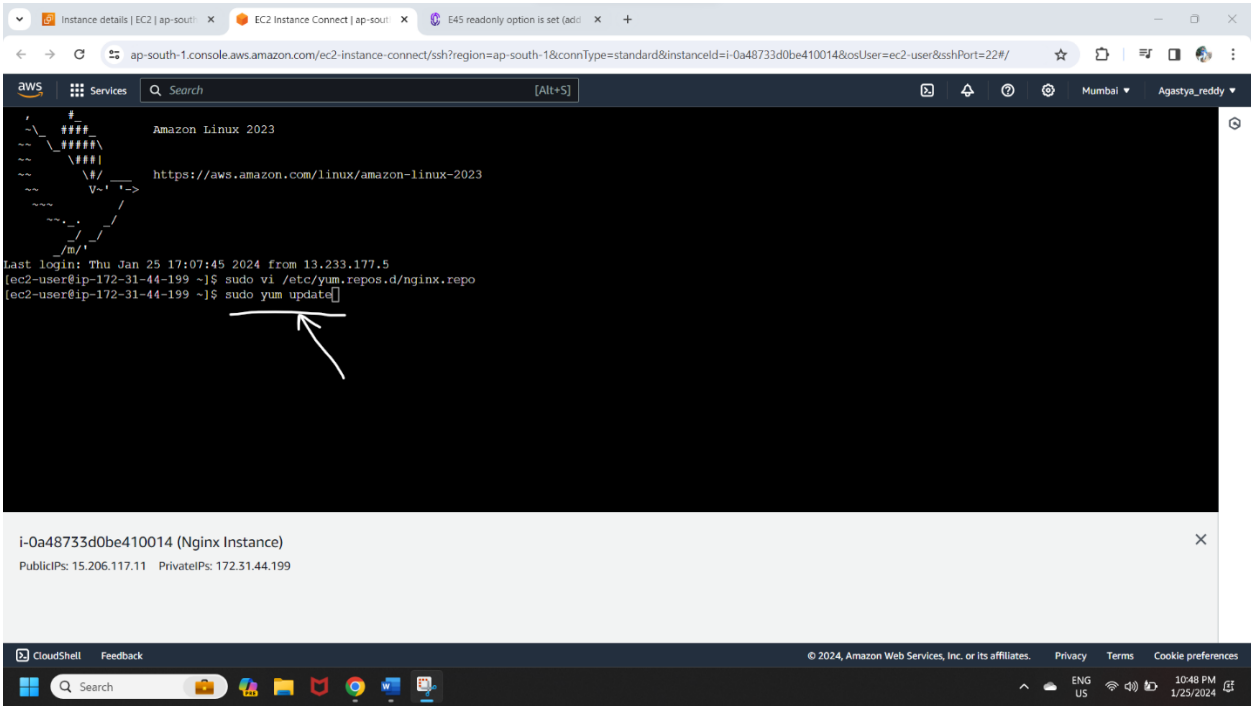
Step-38: Next you need to insert the code which we have seen in step-2 of nginx server installation. And you need to give "**:wq**" to save the file.



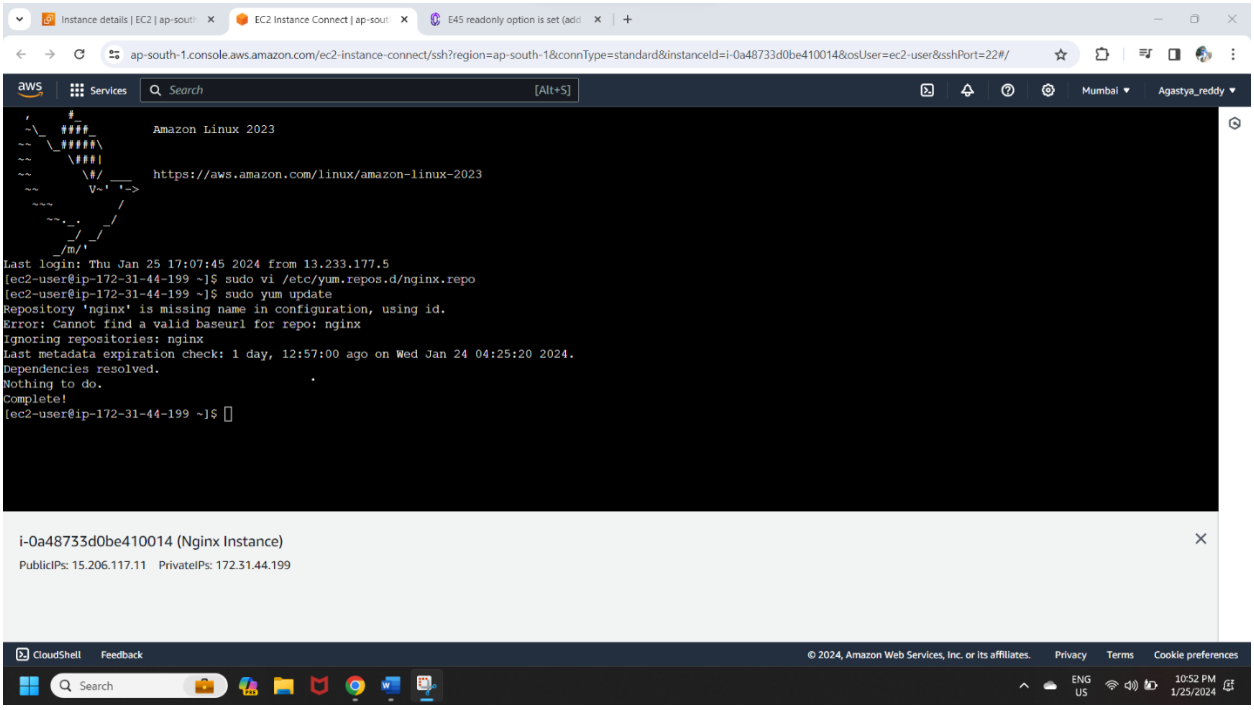
Now you can see the output screen as below.



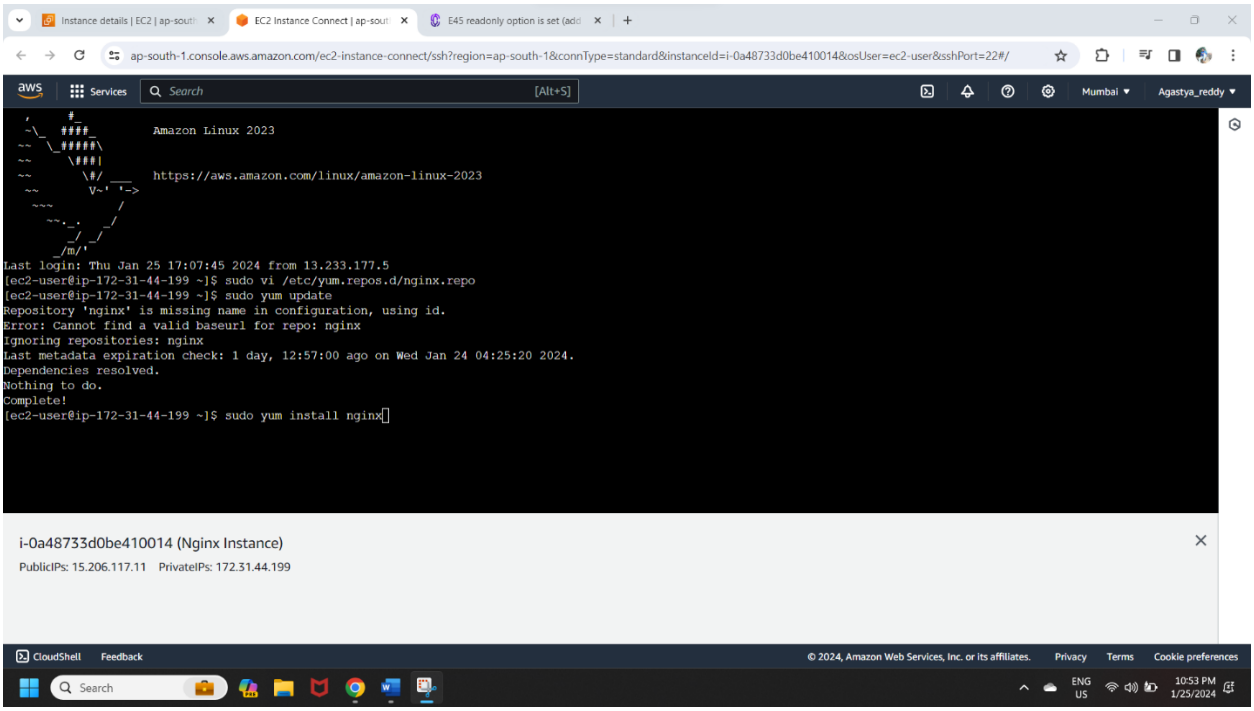
Step-39: Next you need to give the code i.e. "**sudo yum update**" and press enter button.



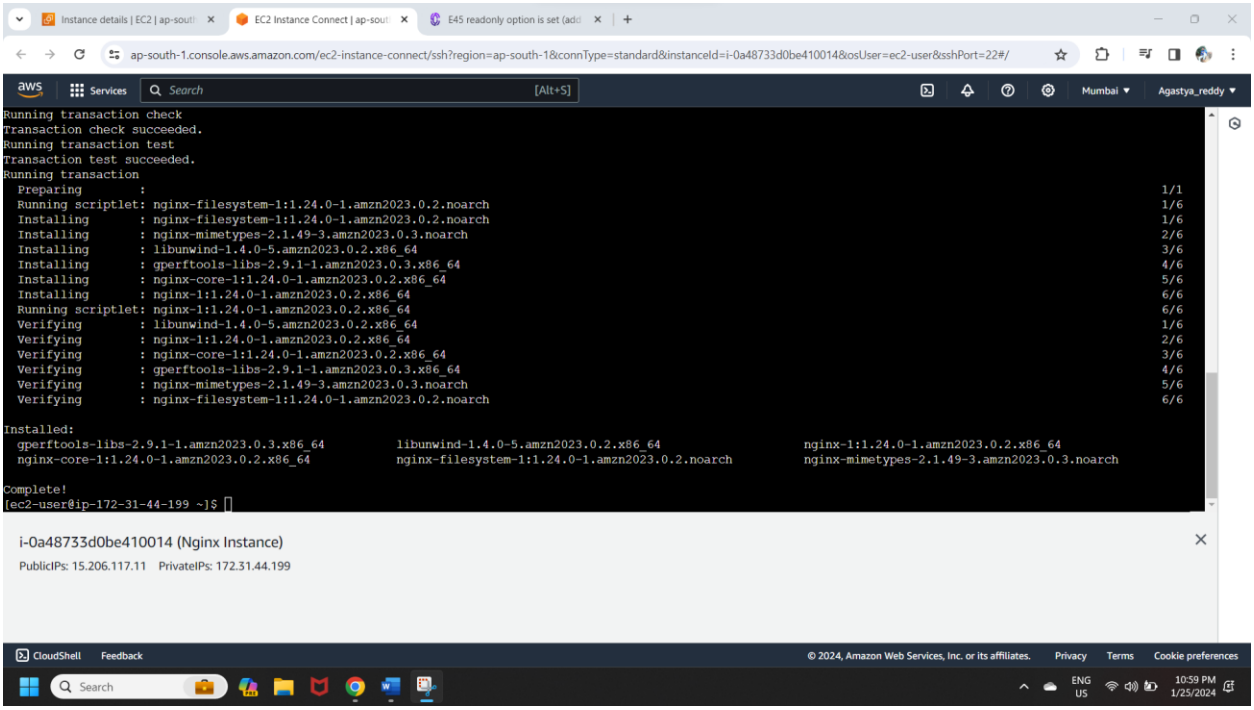
Now the code will run and yum will be updated. Please wait for some time to update. Then you can see the yum update process has completed and it shows all the process had done and complete word to you.



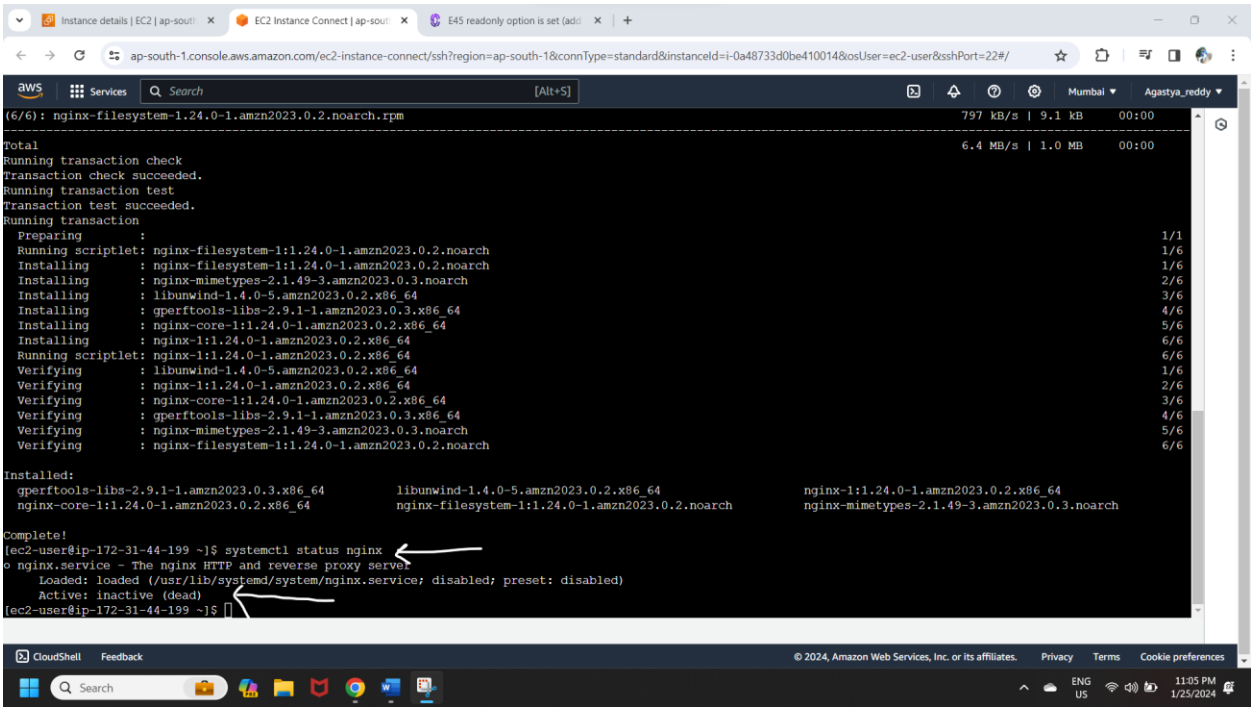
Step-40: Next you need to write the code i.e. **"sudo yum install nginx"** and press enter. Then you can get the option displaying as "**Is this ok [Y/N]"** and you need enter Y and press enter key.



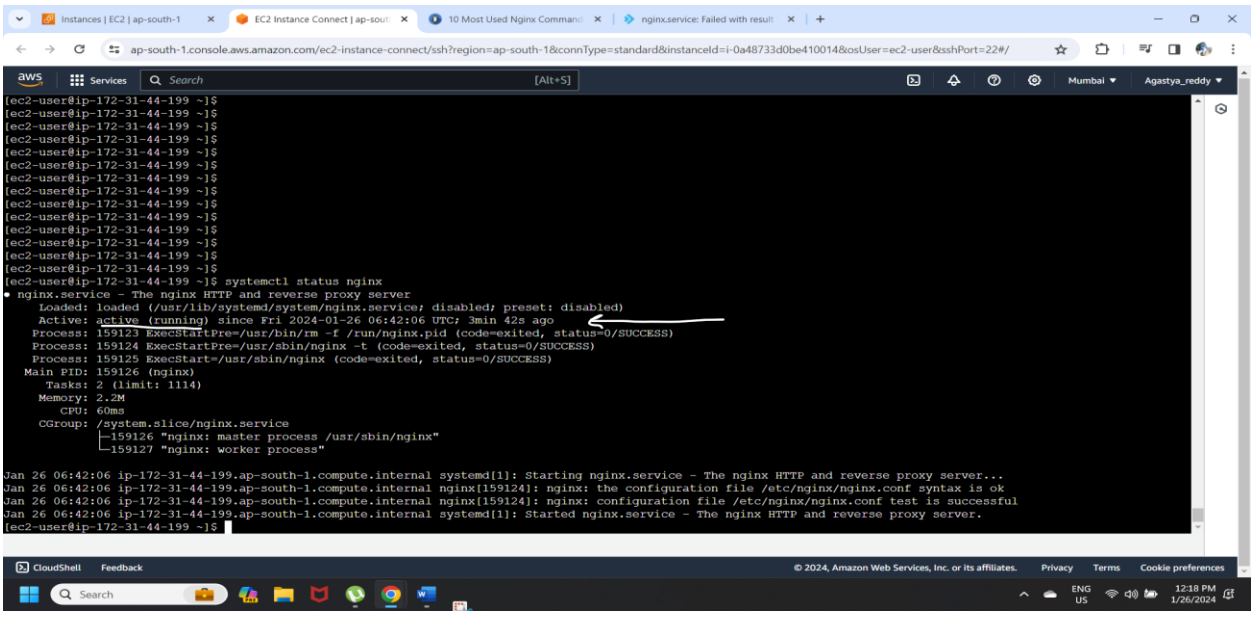
And after entering Y, your transaction will start. It will take some time to complete the installation. If the installation is completed, then you can see the complete line below.



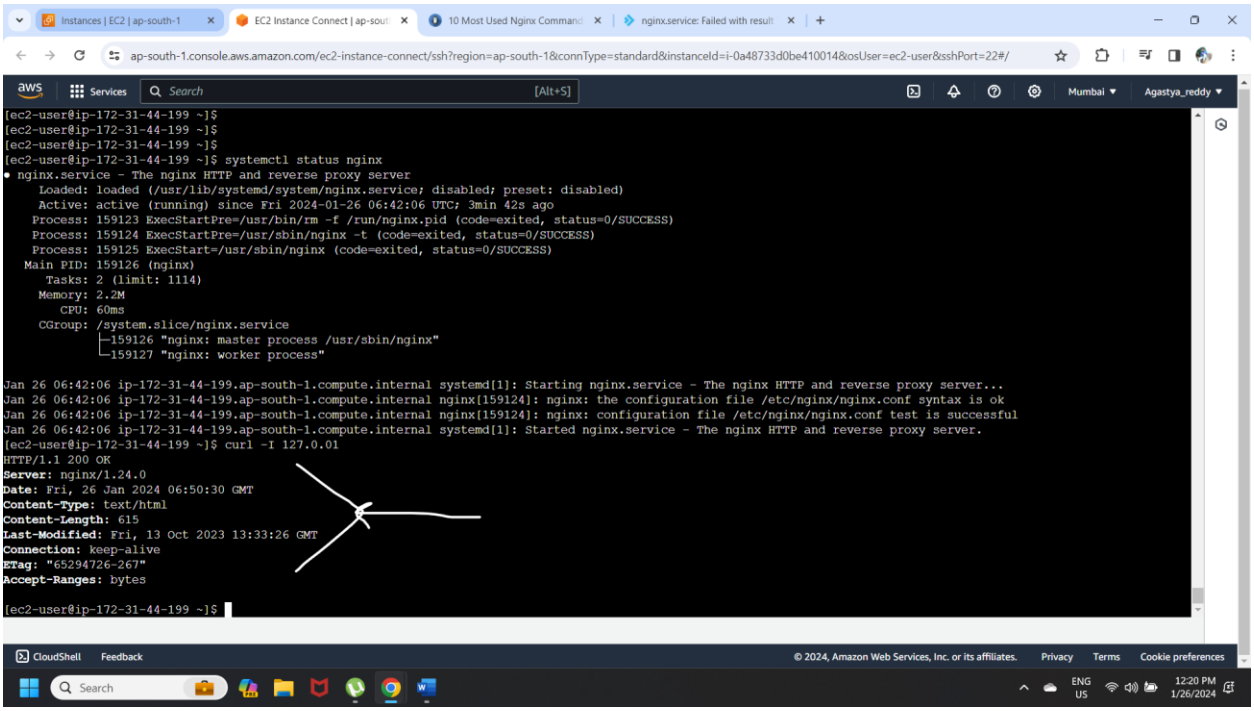
Step-41: Now we need to check the status of the of nginx by writing the code as "**systemctl status nginx**" and press enter key then you can get the status as "**Inactive**" in the screen.



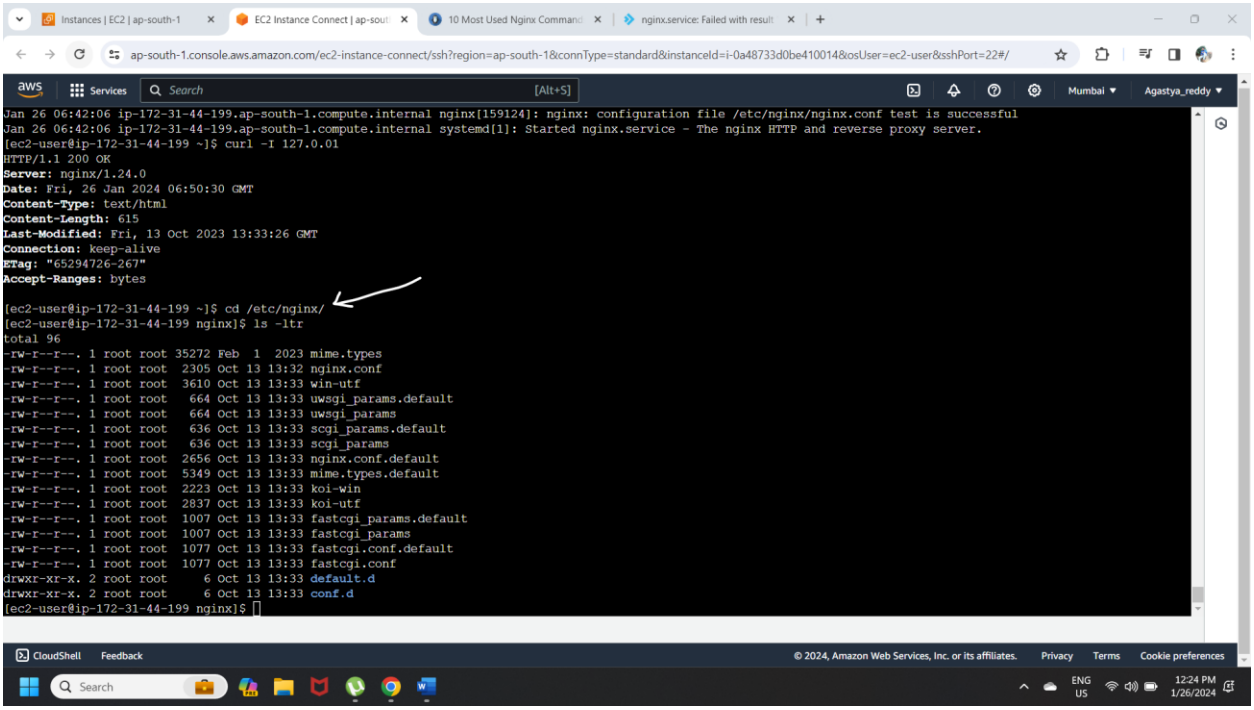
Step-42: You see that nginx is inactive(dead). To start your nginx server you need to give the code i.e. **"systemctl start nginx"** and press enter then the nginx will be started and if you need to check whether nginx is started the give code that have give before which is used to check the status of the nginx. If you start nginx, in background the nginx server will start running. To check the status that nginx server running or not, use the command i.e. **"systemctl status nginx"**.



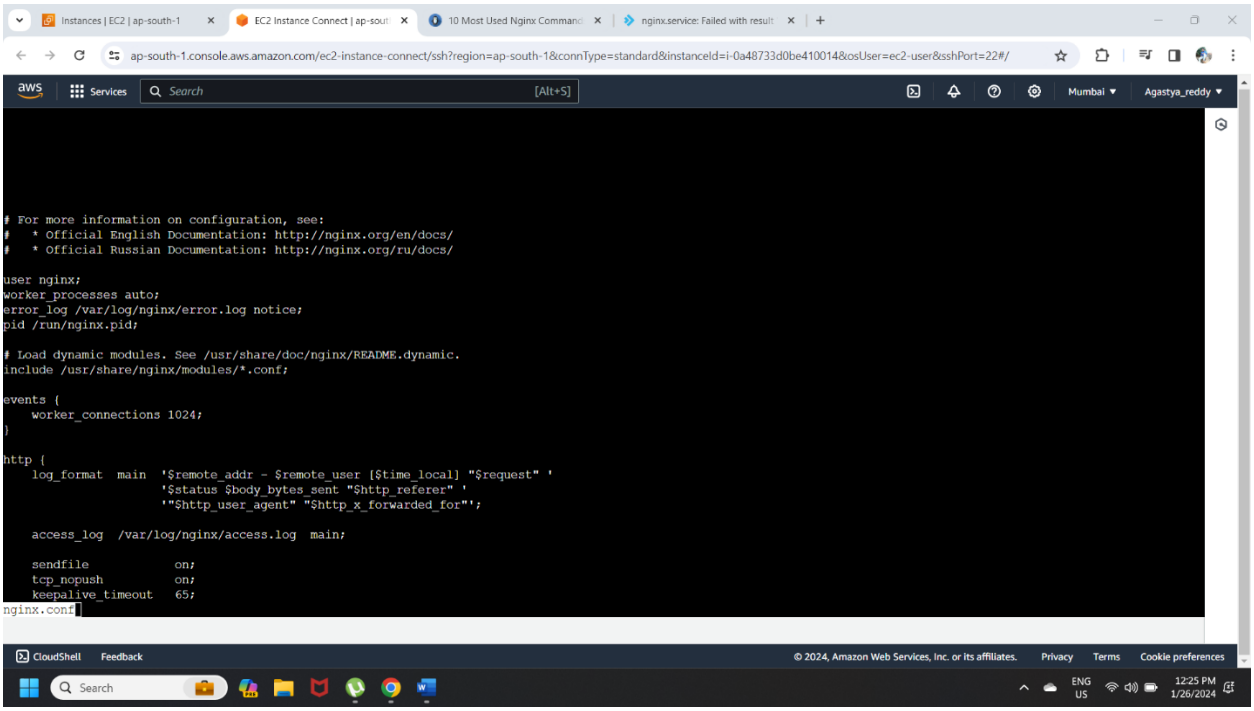
Step-43: Then next you are going to give the curl command i.e. **"curl -I 127.0.01"** which is used to verify the nginx server. Then you can see all the details of nginx as shown in below image.



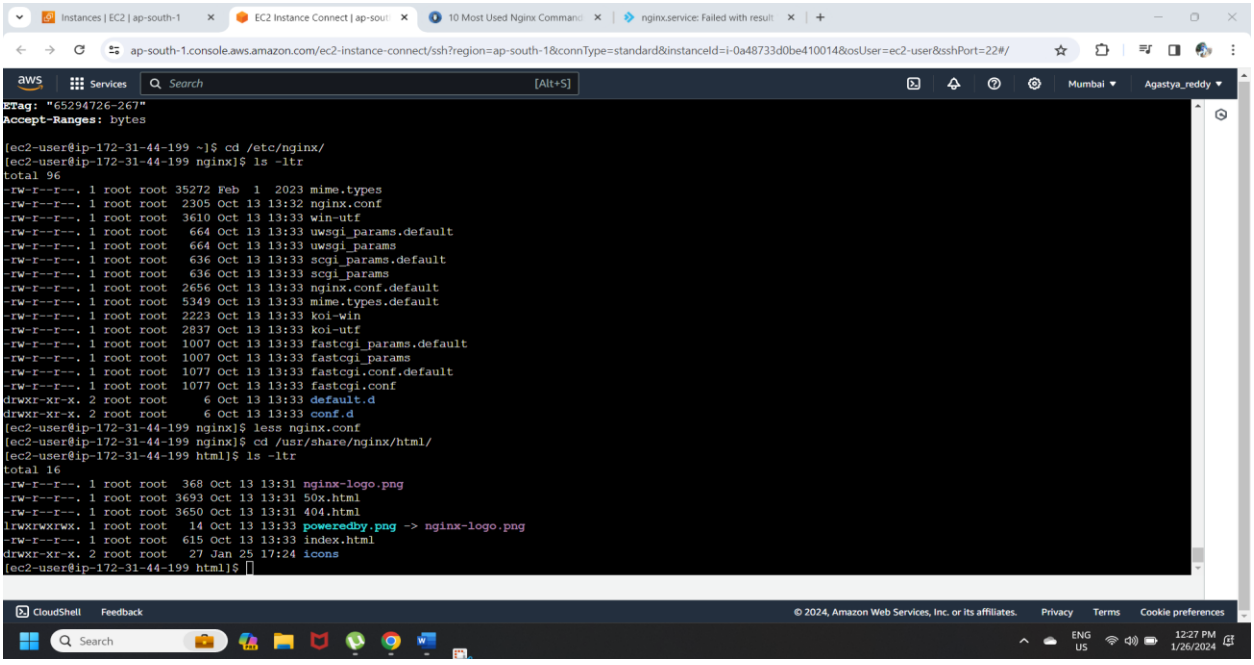
Step-44: Nginx stores its configuration files with **"cd /etc/nginx/"** and use **"ls -ltr"** command to display all the files.



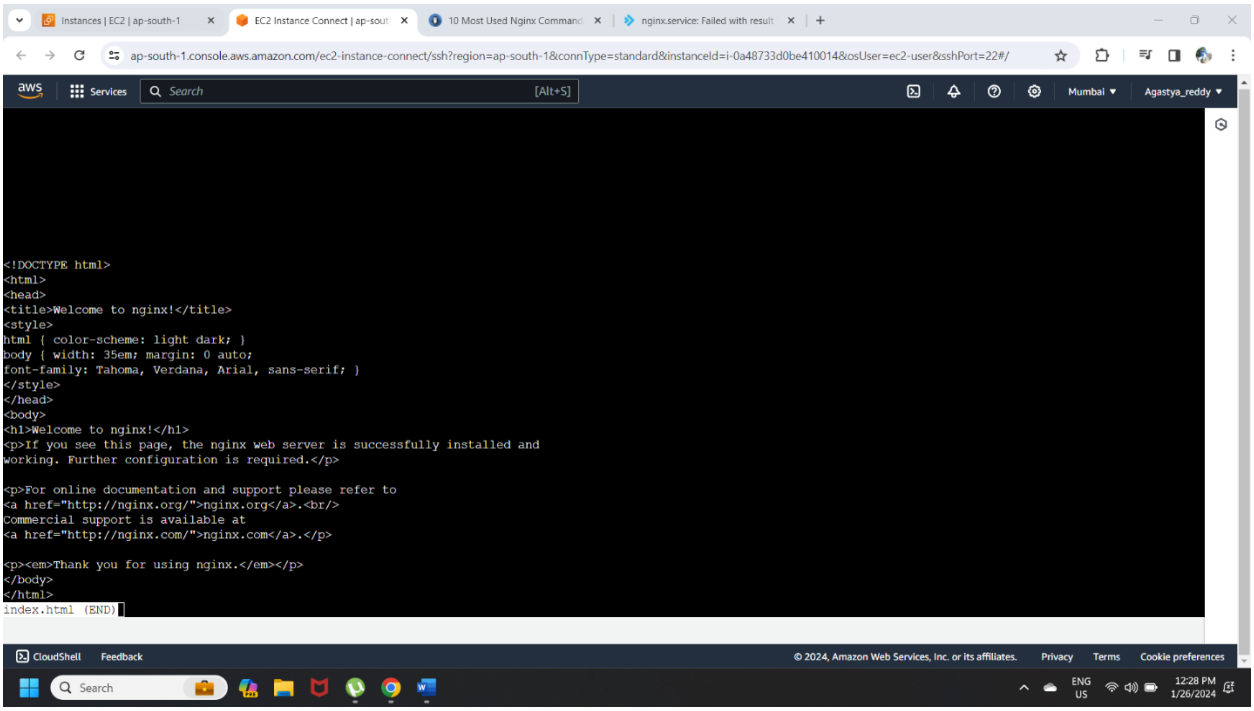
Step-45: After the above step, to see the nginx configuration you need to give this command i.e. **"less nginx.conf".** Then you can see the documentation type of nginx, user setup, and other configuration details also.



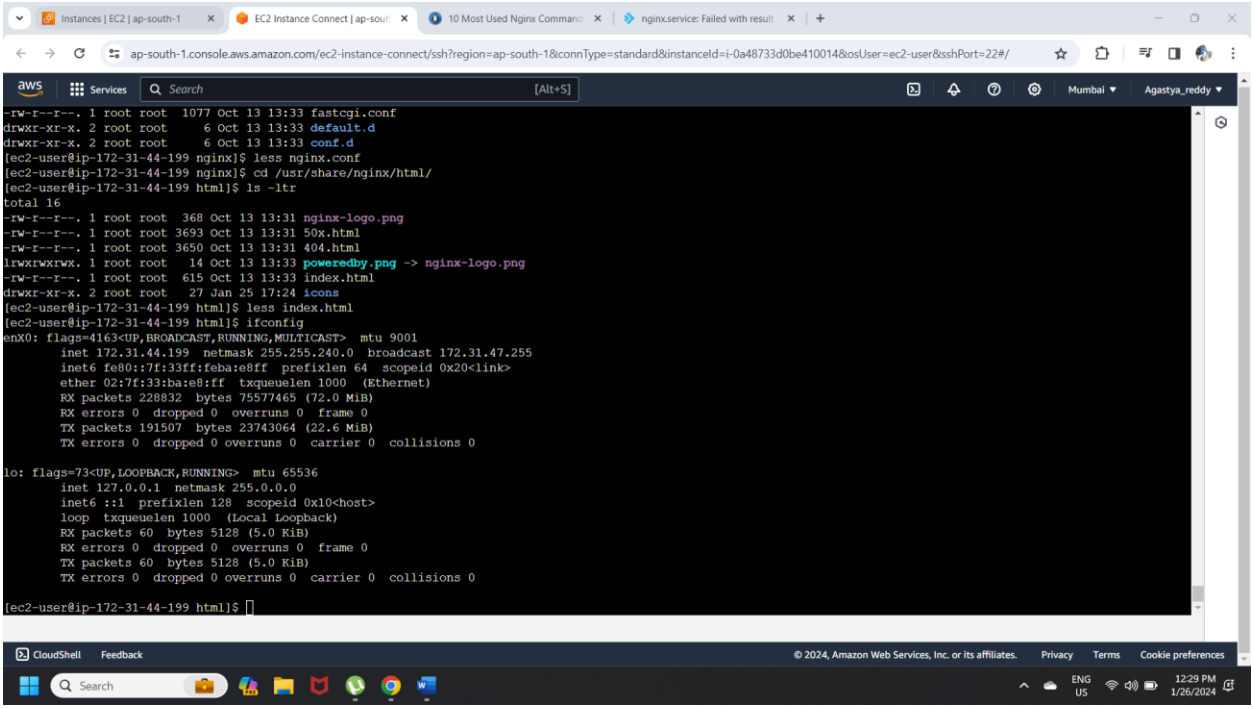
Step-46: To see the html code of your webpage, you need to give the command **"cd /usr/share//nginx/html/"** and go to the location by giving **"ls -ltr".** Then you can see the html files and some image files also.



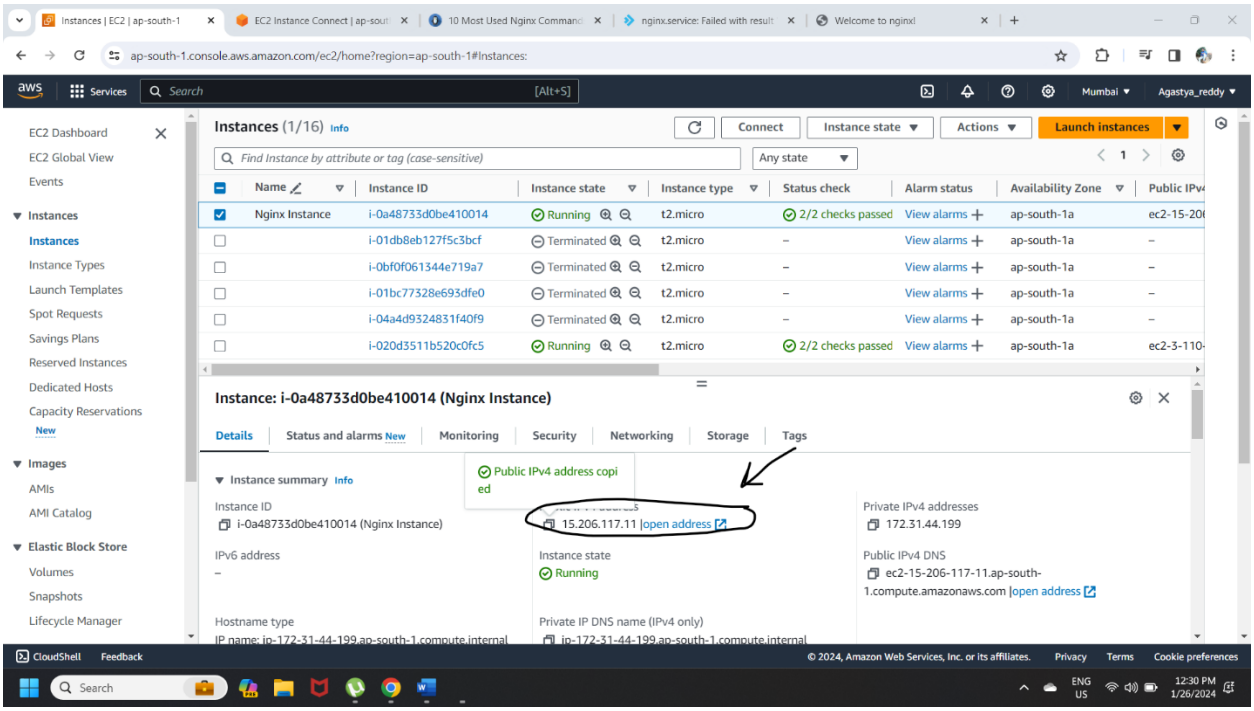
Step-47: To see the html file use "**less index.file",** then you see the html file and if you need you can edit the file as you needs.



Step-48: Now use **"ifconfig"** command to see the IP address of your web server.



Step-49: Next go to the instances. Now we are going to request our web server does it working properly or not by copying our instance Public IPv4 address or Private IPv4 address.



Step-50: After copying the IPv4 address, paste the address in the new tab then you can see the webpage below. If you don't get the output, please check all the steps, then you can get the result.