**Building A Highly Scalable Web App On AWS: A Step-By-Step Guide**

Hey everyone! Want to build a web application that can handle a **surge of traffic** without breaking a sweat? Look no further! This guide will take you through setting up a highly available and scalable web application architecture on AWS.

By the end of this post, you'll have a web app running on AWS that can automatically adjust its resources based on demand. Pretty cool, right?

Checkout this GitHub repository <https://github.com/peaceissa/AWS-Load-Balancing-and-Auto-Scaling> to access all the project files and images

**Here's what we'll cover:**

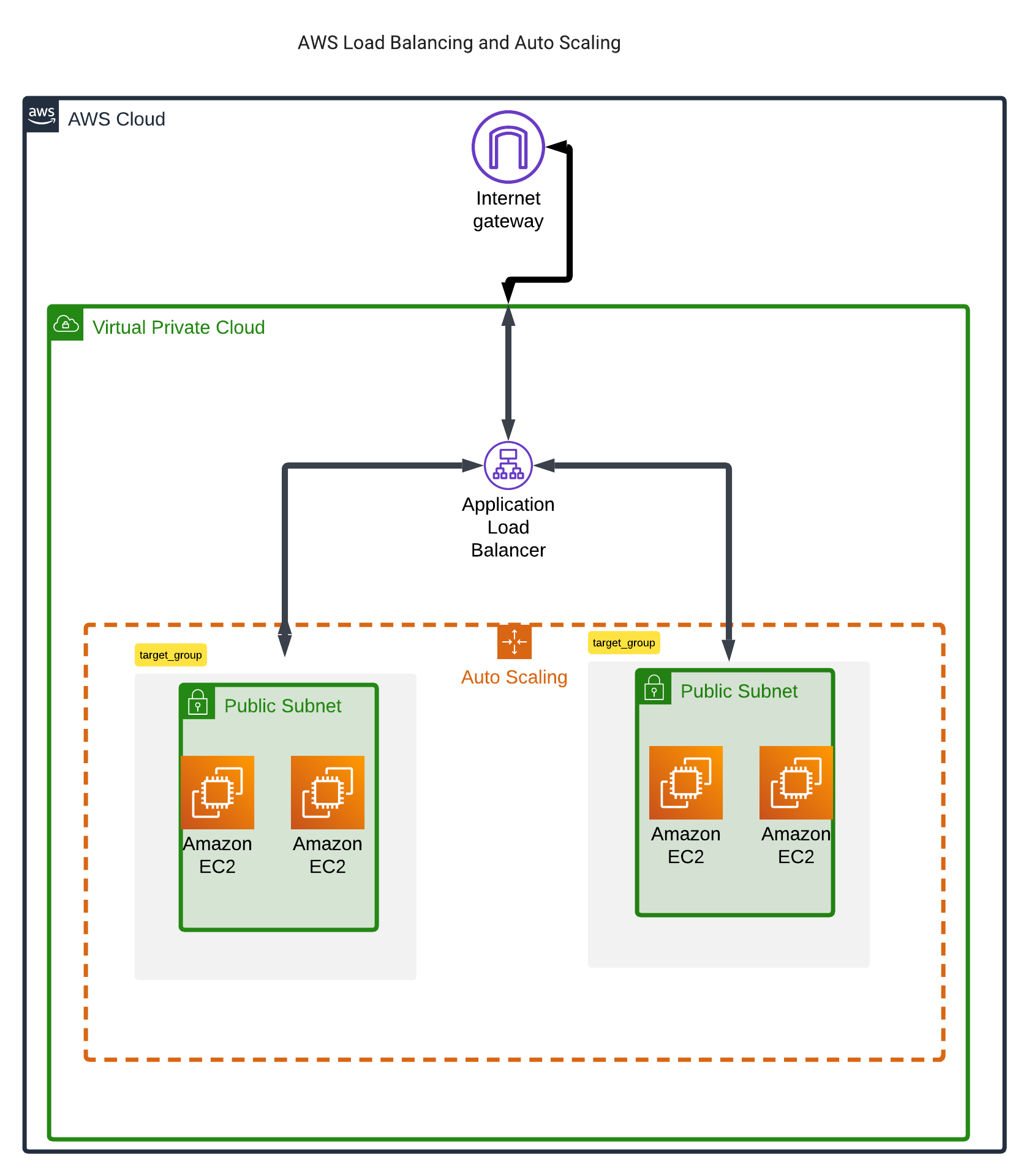
* Setting up an Application Load Balancer (ALB) to distribute traffic across multiple servers.
* Configuring Auto Scaling to automatically add or remove servers based on traffic.
* Using the AWS Stress Test Utility to simulate heavy traffic and see how your app scales.

**Prerequisites:**

* An AWS account (with enough permissions to create resources).
* A basic understanding of AWS services like EC2, Auto Scaling, and Application Load Balancers.
* Familiarity with the Linux command line (don't worry, it's not too scary!).

Alright, let's get scaling!

**ARCHITECTURE DIAGRAM**



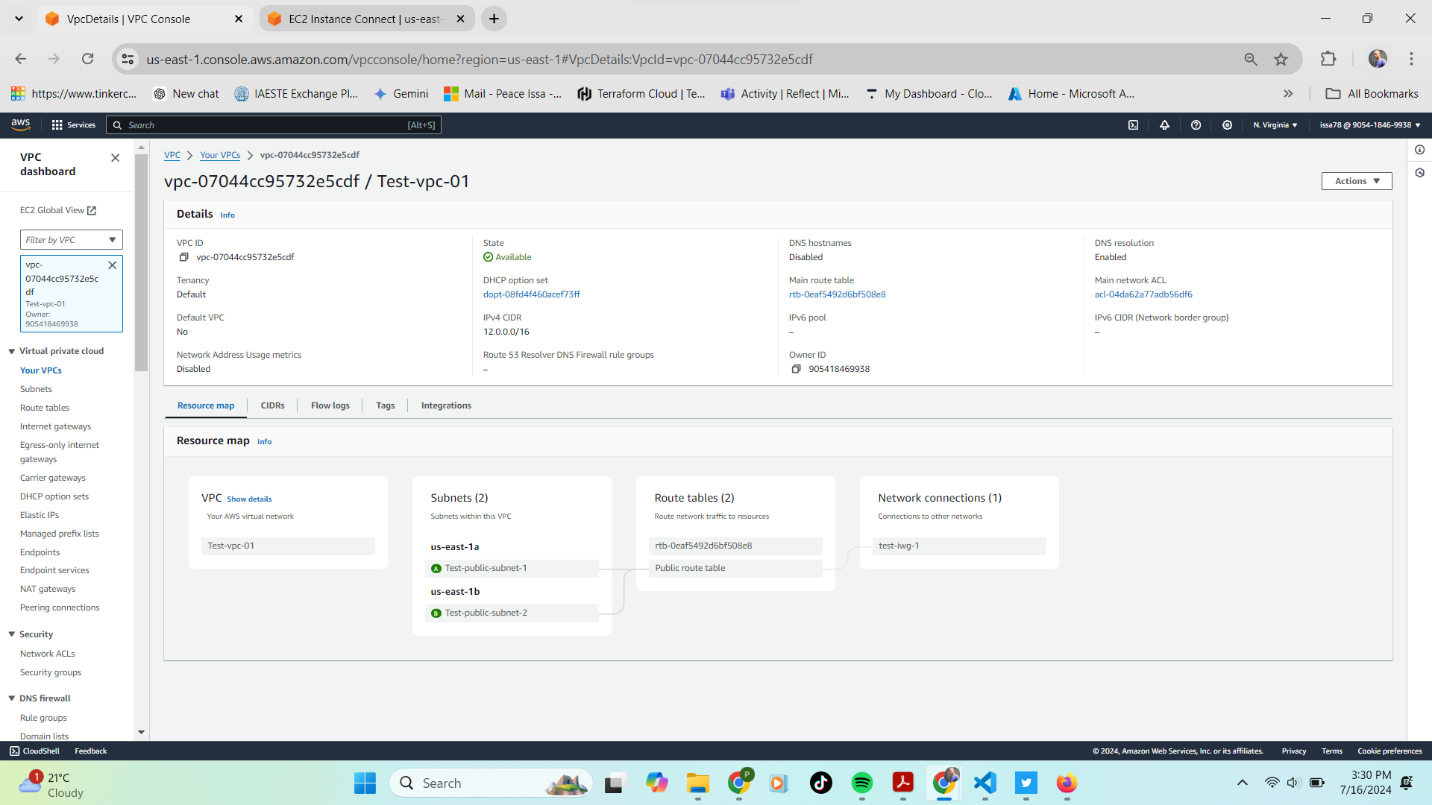
We will be using the above architecture diagram for reference when setting up our resources.

Remember to include route tables and security groups in your architecture diagram. I recommend using lucid chart as it comes preloaded with architecture icons for various cloud service providers.

**Step 1: Setting Up and Configuring Your VPC**

Navigate to vpc by searching vpc on the search bar on your AWS management console.

1. click create vpc, add an appropriate name for this case I used Test-vpc-01 then assign an appropriate CIDR block. 12.0.0.0/16 in my case.
2. click on internet gateways on the left panel navigation menu, click on create internet gateway assign name (Test-igw) create then click on the actions Tab, attach to vpc and select your vpc to attach to the internet gateway.
3. Next up we will set up the subnets. We will require at least two public subnets to demonstrate load balancing. Click on create subnet and select your vpc. For our first subnet we will use Test-public-subnet-01 for the name, us-east-1a for the Az, 12.0.1.0/24 for the IPv4 subnet CIDR block. For our second subnet we will use Test-public-subnet-02 for the name, us-east-1b for the Az, 12.0.3.0/24 for the IPv4 subnet CIDR block. **Note:** *Ensure to use at least two different availability zone as it will be a requirement when setting up an Auto Scaling Group*
4. Navigate to create route table, add route table name (Test-public-RT) then select your vpc (Test-vpc-01). Once created navigate to subnet associations, click on edit subnet associations select the two public subnets and save associations. click one edit routes, add route then input 0.0.0.0/0 as the Destination, internet gateway as the target then select the internet gateway you created in step 2.



Once you are done with the four steps your vpc resource map should resemble the one above.

**STEP 2: SETTING UP SECURITY GROUPS**

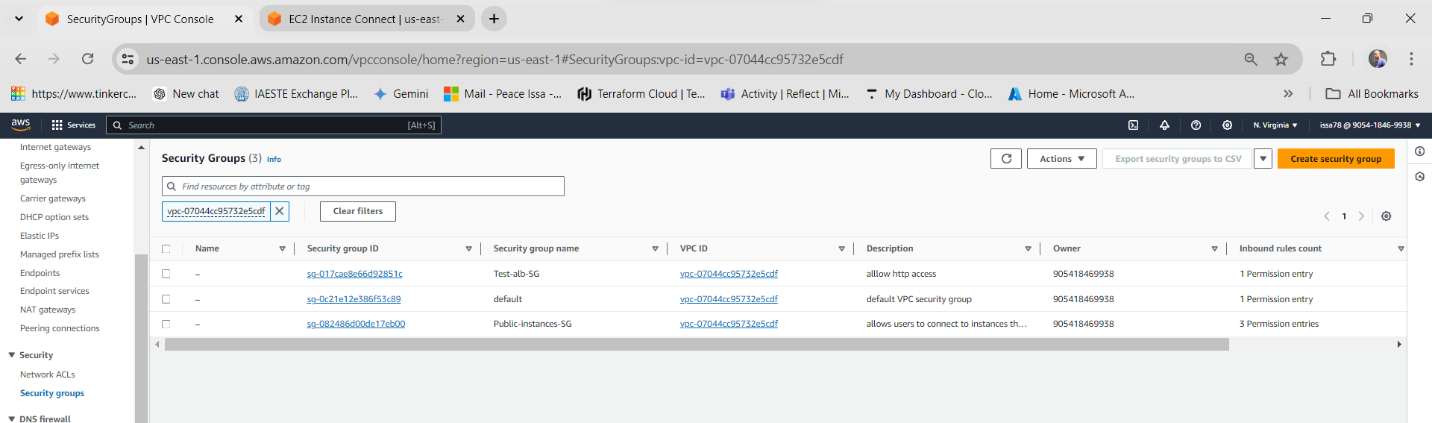
We will set up two security groups that will be using with our subnets. You can also add Network ACLs to increase security but ensure to use the appropriate rules.

The first security group will be for the instances that we’ll be launching to enable connection to the instances through http, ssh and https.

* Select security groups then click on create security group
* Add name (Test-public-instances-SG) description – allow internet access then select your vpc (Test-vpc-01)
* Add an inbound rule with the type ssh and source type as anywhere-IPv4 and other rule with the type of http and source type anywhere IPv4

The second security group will be for the application load balancer to allow http traffic through.

* Select security groups then click on create security group
* Add name (Test-ALB-SG) description – allow http access then select your vpc (Test-vpc-01)
* Add an inbound rule with the type of http and source type anywhere IPv4



**STEP 3: Building a Traffic Director, The Application Load Balancer (ALB)**

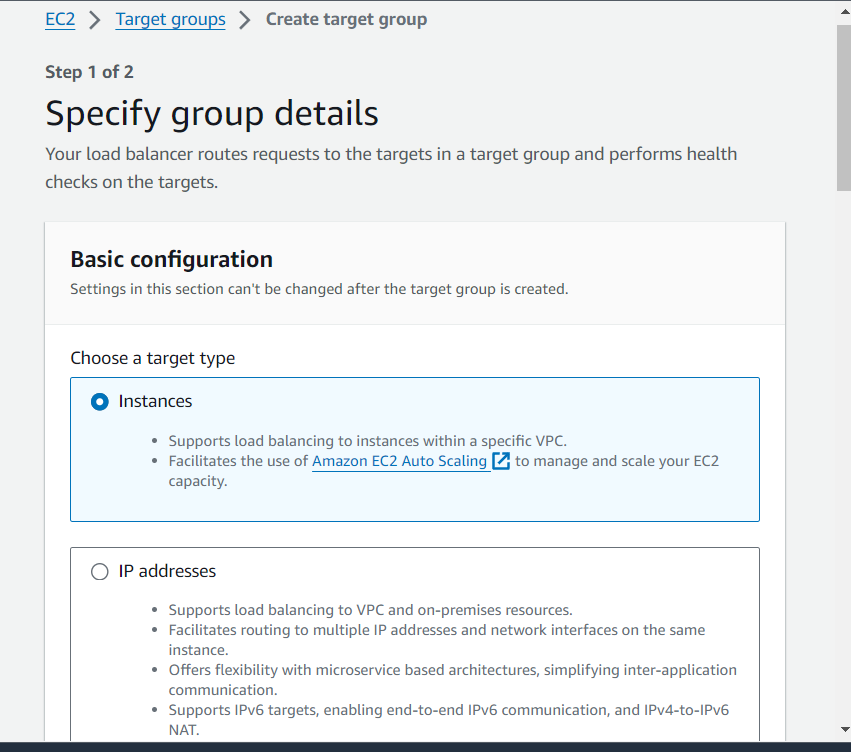
Think of the ALB as a smart traffic cop. It directs incoming traffic to the most appropriate server in your web application.

**3.1. Creating Target Groups:**

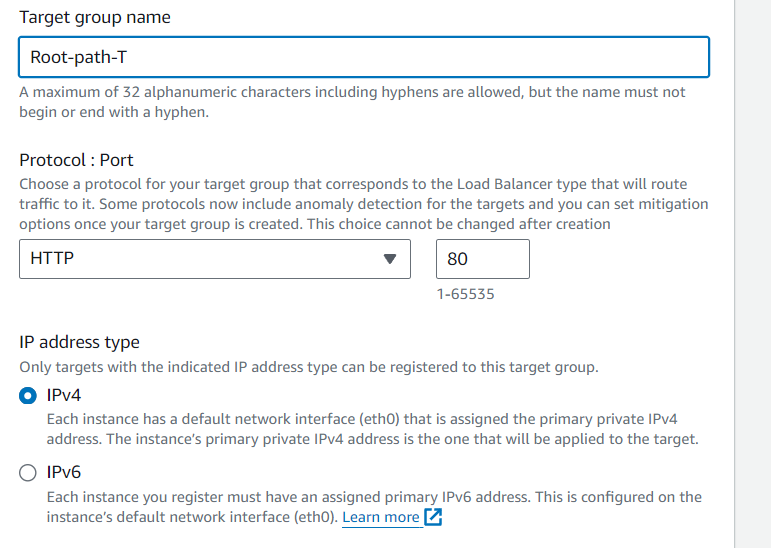
We will create the Target groups first. Target groups are like teams of servers that the ALB can send traffic to. We'll create two:

1. **First Target Group:** This group will handle requests for the root path (/). Name it something descriptive like "Root-path-Tg."

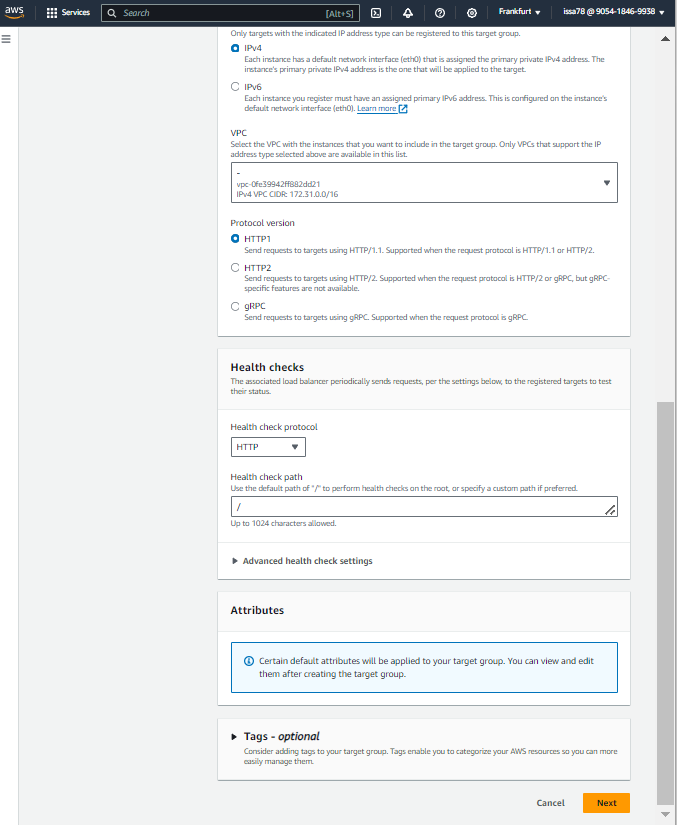
* Head to the EC2 Dashboard on the left panel navigation menu.
* Under "Load Balancing," select "Target group" and click "Create Target Group."
* Choose "instances" as target type and give it a memorable name (e.g., "Root-path-Tg").



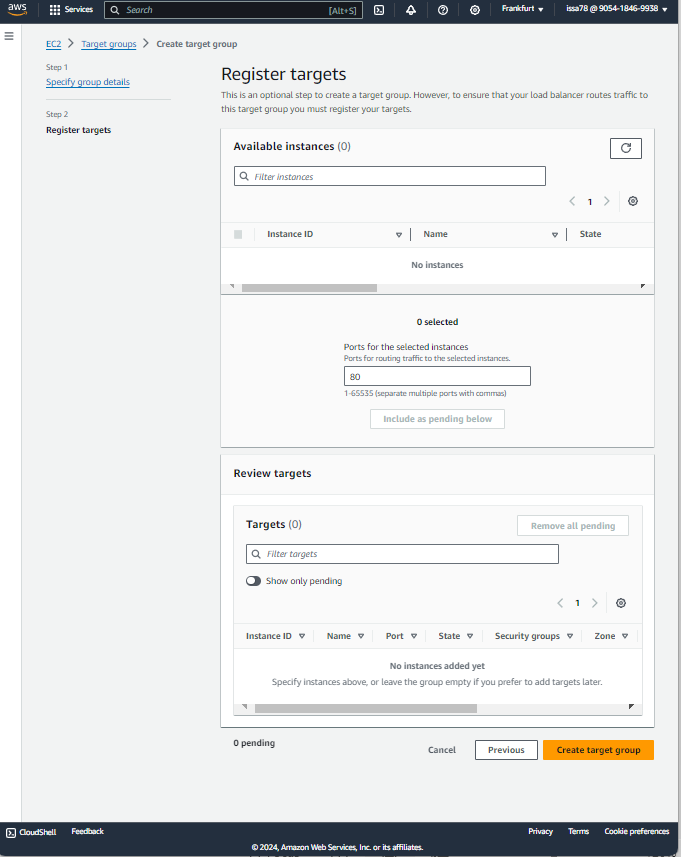
* Make sure "HTTP" is selected for the protocol 80 for port and IPv4 for IP address type.



* Select your vpc and leave the other settings with their default values.



* Click on next and then create target group



1. **Second Target Group:** This group will handle requests specifically for the /users’ path. Name it "users-path-Tg."
   * Repeat the steps above but use a different name (users-path-Tg) and 81 for the port

Once the Target group is created it should resemble the one in the picture below.

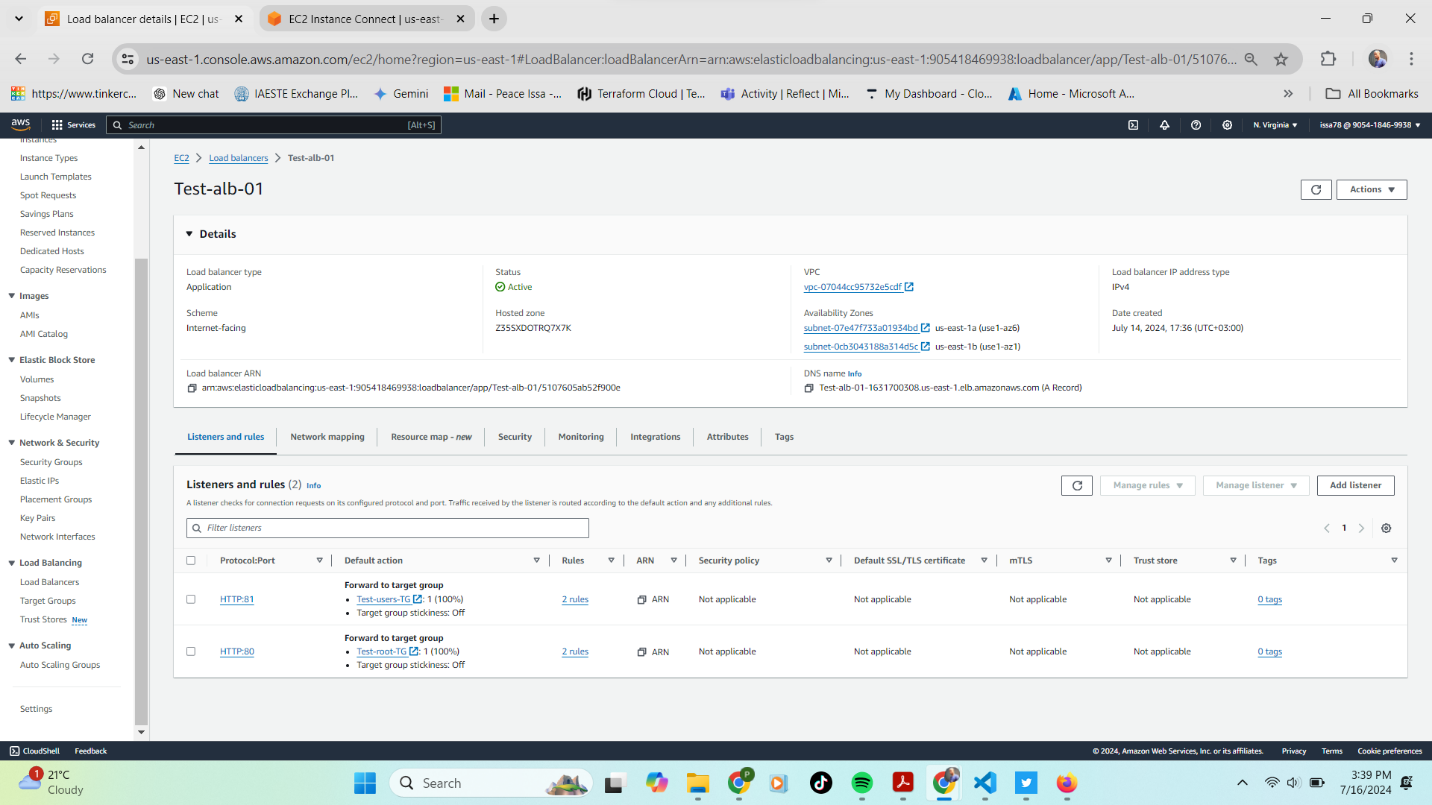
A screenshot of a computer

Description automatically generated

* 1. **Creating the application Load Balancer**

1. Head to the EC2 Dashboard.
2. Under "Load Balancing," select "Load Balancers" and click "Create Load Balancer."
3. Choose "Application Load Balancer" and give it a memorable name (e.g., "Test-ALB").
4. Make sure "Internet-facing" is selected for the scheme and IPv4 Load balancer IP address type.
5. Select your VPC and choose at least two Availability Zones for high availability (think of them as data centers in different locations).
6. Select the security group (Test-ALB-SG) created earlier in STEP 2.
7. Select the root-path-Tg to add the root-path target group to the listeners and routing tab. Click on Add listener to add users-path-Tg to the Listeners on your ALB.
8. Leave the other configurations as default and click create load balancer

Once the load balancer is created, it should resemble the one below with two listeners using HTTP protocol, one running on port 80 and the other on port 81.



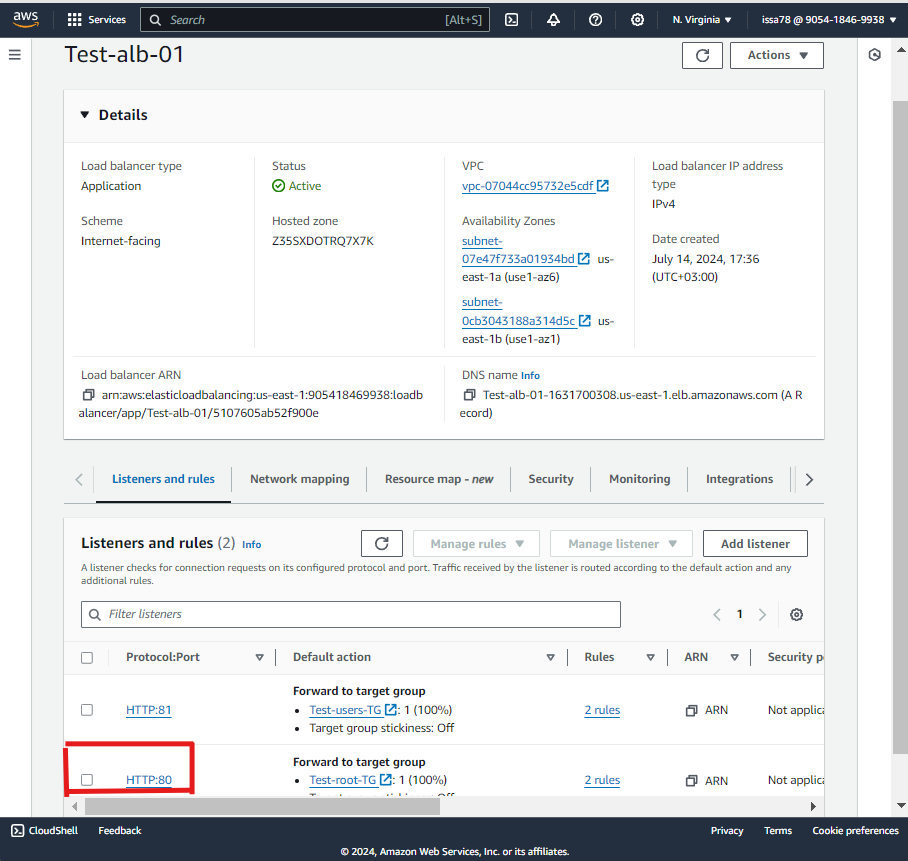
* 1. **Setting Up the Rules: Who Gets What Traffic?**

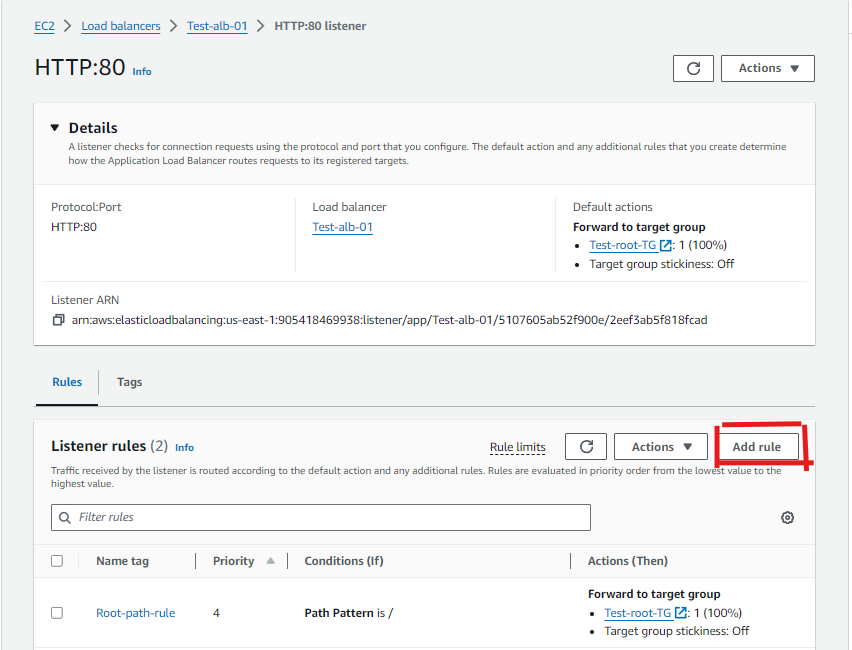
Now, we need to tell the ALB which target group to send traffic to based on the URL path. Here's how:

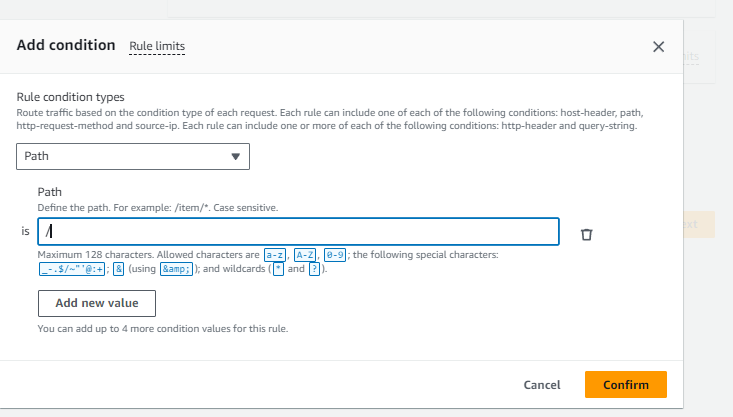
We'll create two rules:

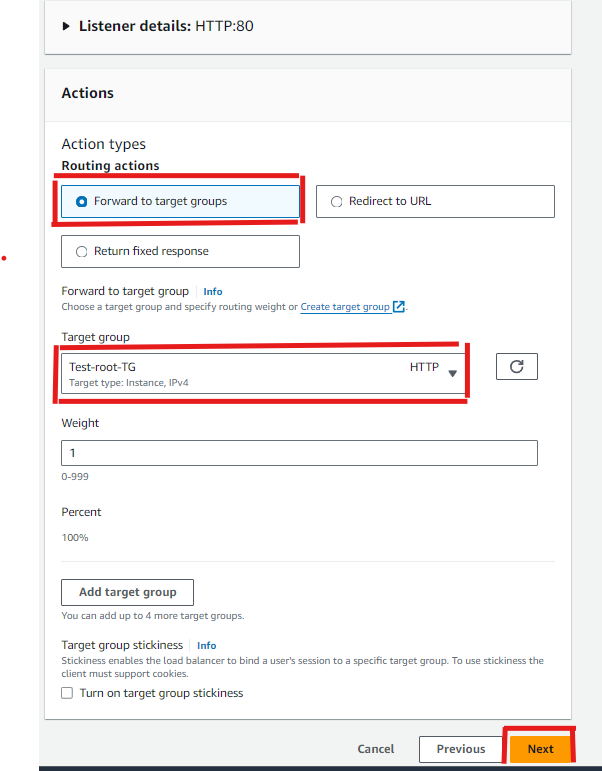
If the path is /, forward traffic to the "Root-path-Tg" group.

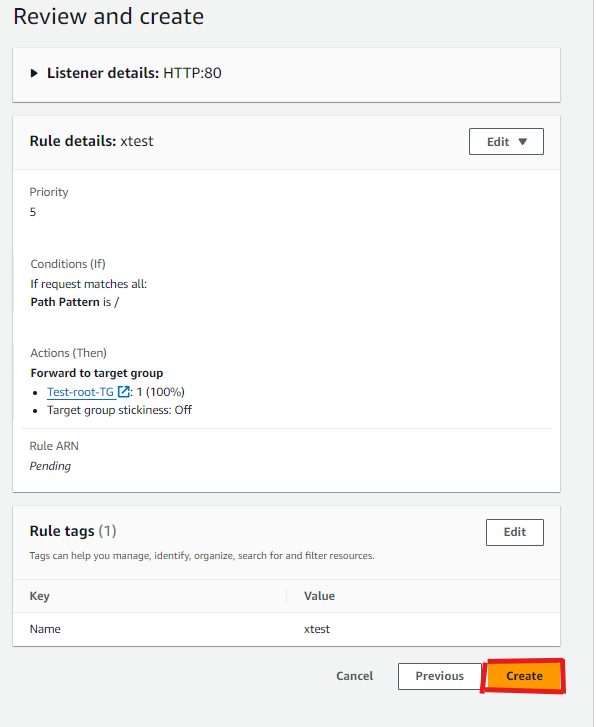
1.Go to the "Listeners and rules" tab for your ALB and click "HTTP:80" for the HTTP listener.



2.Then click on add rule, assign name and click next  


1. Click on add condition, choose path and define it as / for root then click on confirm.  
   

4. Select Forward to target groups as the action type and the Test-root-Tg as the target group then click next.  


1. Set the rule priority level as 4 and click next.
2. Review and click on create.  
   

If the path is /users, forward traffic to the "users-path-Tg" group

Repeat the above steps but ensure that the path is set to /users and the rule priority level is set to 2.

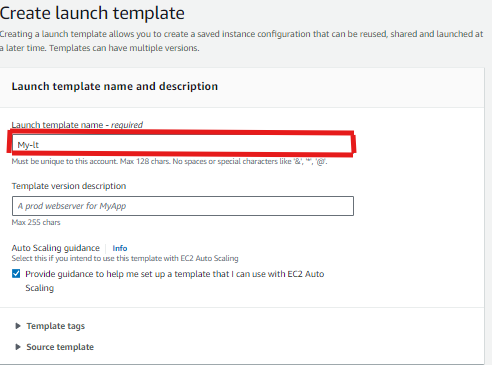
**4. Introducing Auto Scaling: The Muscle Behind the Magic**

Auto Scaling ensures your web application has enough resources to handle traffic spikes. Here's how to set it up:

1. **Create a Launch Template:**

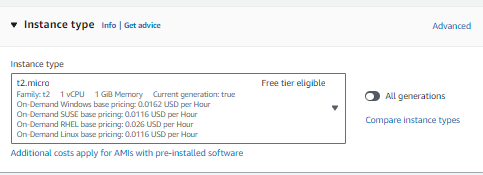
This is a blueprint for your web servers. Configure it with the appropriate AMI (operating system image), instance type, and user data script (a script that installs and configures your web server software).

* 1. In the left-hand navigation pane, click on "Launch Templates" under the "Instances" section.
  2. Click the "Create launch template" button.
  3. Enter a name for your launch template (e.g., MyLaunchTemplate). Provide a brief description (optional).

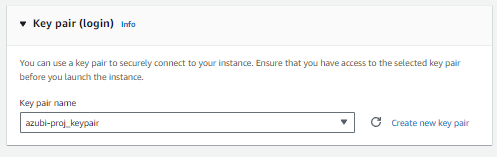


* 1. Configure Instance Details
     1. Select an Amazon Machine Image (AMI) for your instances. You can search for a specific AMI or select one from the list.  
        A screenshot of a computer

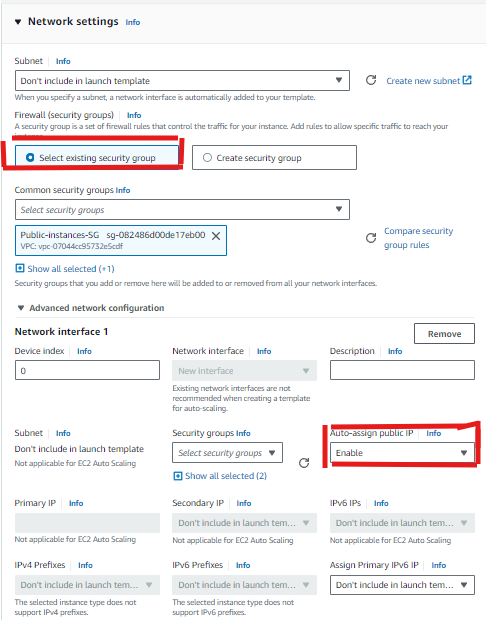
        Description automatically generated
     2. Choose the instance type that suits your needs (e.g., t2. Micro to avoid charges as it is free tier eligible).



* + 1. Select an existing key pair or create a new one to connect to your instances.



* + 1. Choose the VPC (Test-vpc-01) where you want to launch your instances and select the two subnets, we created earlier within our VPC.
    2. Click on advanced network configuration, add network interface then Enable auto assign public address this option to allow ec2 access through the internet



* 1. Choose one or more security groups (Test-public-instances-SG) to associate with the instances.
  2. Configure the root volume and any additional EBS volumes. You can modify the size, type, and other settings.
  3. Add tags to your instances for better management and identification.

Advanced Details

1. User data:

Add user data to run commands or scripts at the instance launch. For our case, a script to:

* 1. install apache​
  2. start apache​
  3. enable apache​
  4. create two files with the following **metadata:**
  5. index.html should contain the ami-id
  6. users.html should contain user data

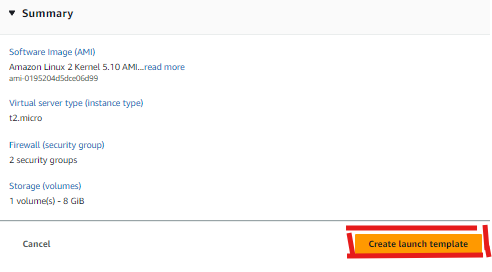
1. #!/bin/bash
2. # Update and upgrade system
3. yes | sudo apt update
4. yes | sudo apt upgrade
5. # Install Apache
6. yes | sudo apt install httpd
7. sudo systemctl enable httpd
8. sudo systemctl start httpd
9. yes | sudo apt install apache2
10. sudo systemctl enable apache2
11. sudo systemctl start apache2
12. yes | sudo apt install curl
13. # Get the AMI ID and write it to index.html
14. AMI\_ID=$(wget -q -O - http://169.254.169.254/latest/meta-data/ami-id || die \"wget ami-id has failed: $?\")
15. echo "<html><body><h1>AMI ID: ${AMI\_ID}</h1></body></html>" > /var/www/html/index.html
16. # Get the user data and write it to users.html
17. USER\_DATA=$(curl -s http://169.254.169.254/latest/user-data)
18. echo "<html><body><pre>${USER\_DATA}</pre></body></html>" > /var/www/html/users.html
19. # Restart Apache to ensure changes are applied
20. sudo systemctl restart apache2

Add the above script to the user data field and enable meta data access.

A screenshot of a computer

Description automatically generated

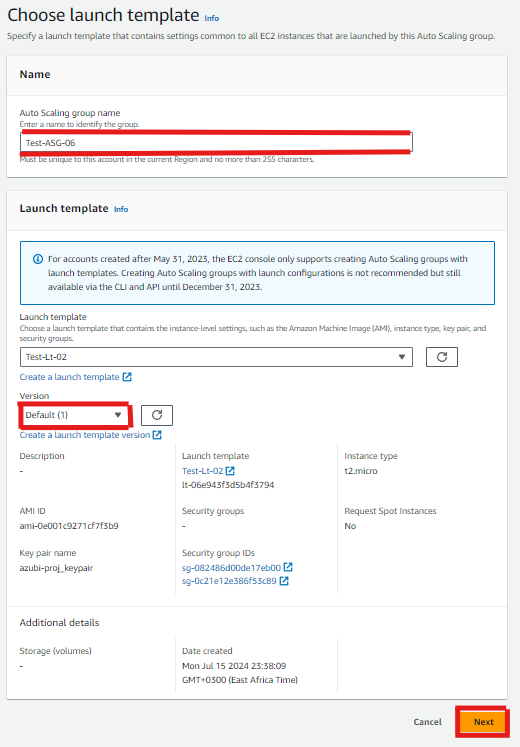
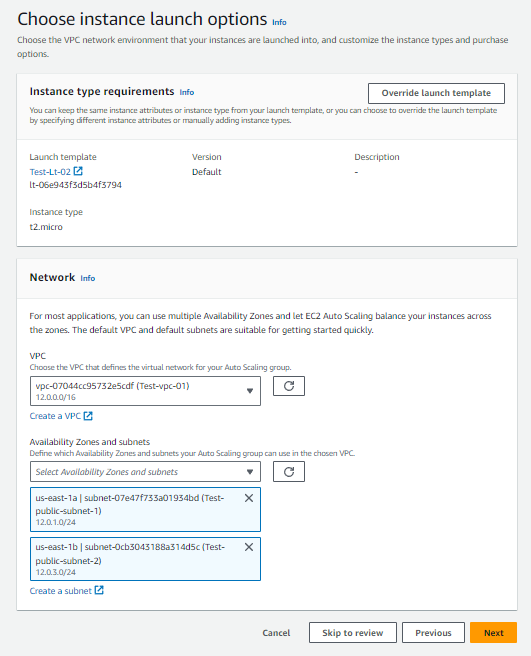
Review all the configurations, then click the "Create launch template" button to create the template.

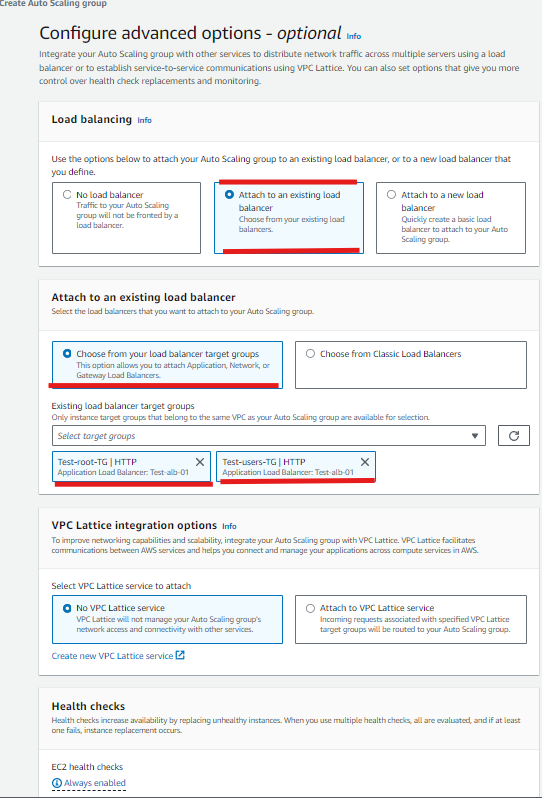


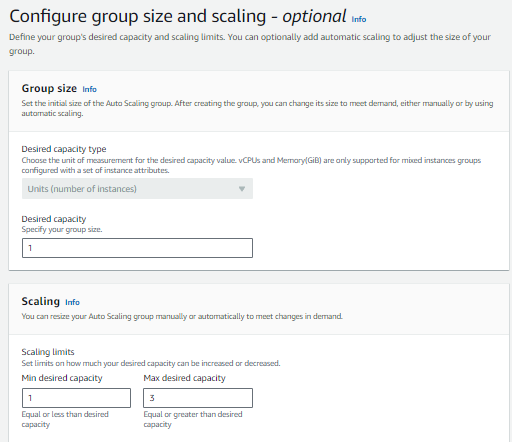
1. **Create an Auto Scaling Group:**

This group will manage the creation and deletion of your web servers based on defined rules. Configure it with the launch template, VPC (virtual private cloud), target groups (we want both!), and a scaling policy that automatically adds servers when CPU utilization gets high (e.g., above 20%).

1. In the AWS Management Console, navigate to EC2 under the "Compute" section.
2. In the EC2 Dashboard, click on Auto Scaling Groups under the "Auto Scaling" section in the left-hand menu.
3. Click on the Create Auto Scaling group button.
4. Choose launch template or configuration:
5. Enter a name for your Auto Scaling group.

* Select the Launch template option and choose the launch template you created previously from the drop-down menu.
* Select the version of the launch template (use the default version or the latest one), then click Next.  
  
* VPC: Choose the VPC where you want to launch the instances.
* Subnets: Select the two public subnets in your VPC(Test-vpc-01). Instances will be launched in these subnets.  
  

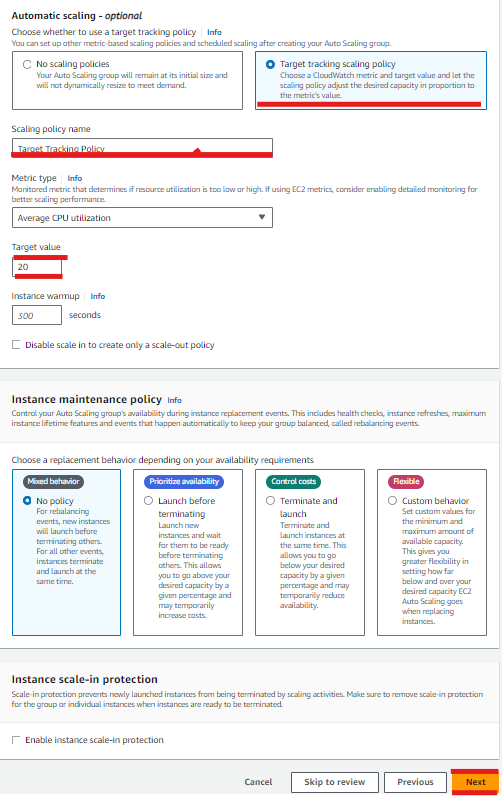
1. We need to configure load balancing, select attach to an existing load balancer, select choose from your load balancer Target groups and select the two target groups created earlier. Leave the other configurations as default and click next.  
   
   * + Desired capacity: Enter the number of instances you want to start with. (1)
     + Minimum capacity: Enter the minimum number of instances. (1)
     + Maximum capacity: Enter the maximum number of instances. (3)



Choose how you want to scale your instances (e.g., target tracking, step scaling, or simple scaling).

For Target tracking scaling policy, click on Add policy and configure the policy:

* + - Scaling policy type: Select Target tracking scaling policy.
    - Metric type: Select a metric, such as Average CPU utilization.
    - Target value: Set a target value, such as 20%.
    - Configure other options as required.



* Configure notifications: You can configure notifications for different events like instance launch, termination, etc. This step is optional.

1. Add tags: Optionally, add tags to your Auto Scaling Group. Tags help in identifying and managing resources.
2. Review: Review all the configurations you have made.
3. Click on the Create Auto Scaling group button to create the group.

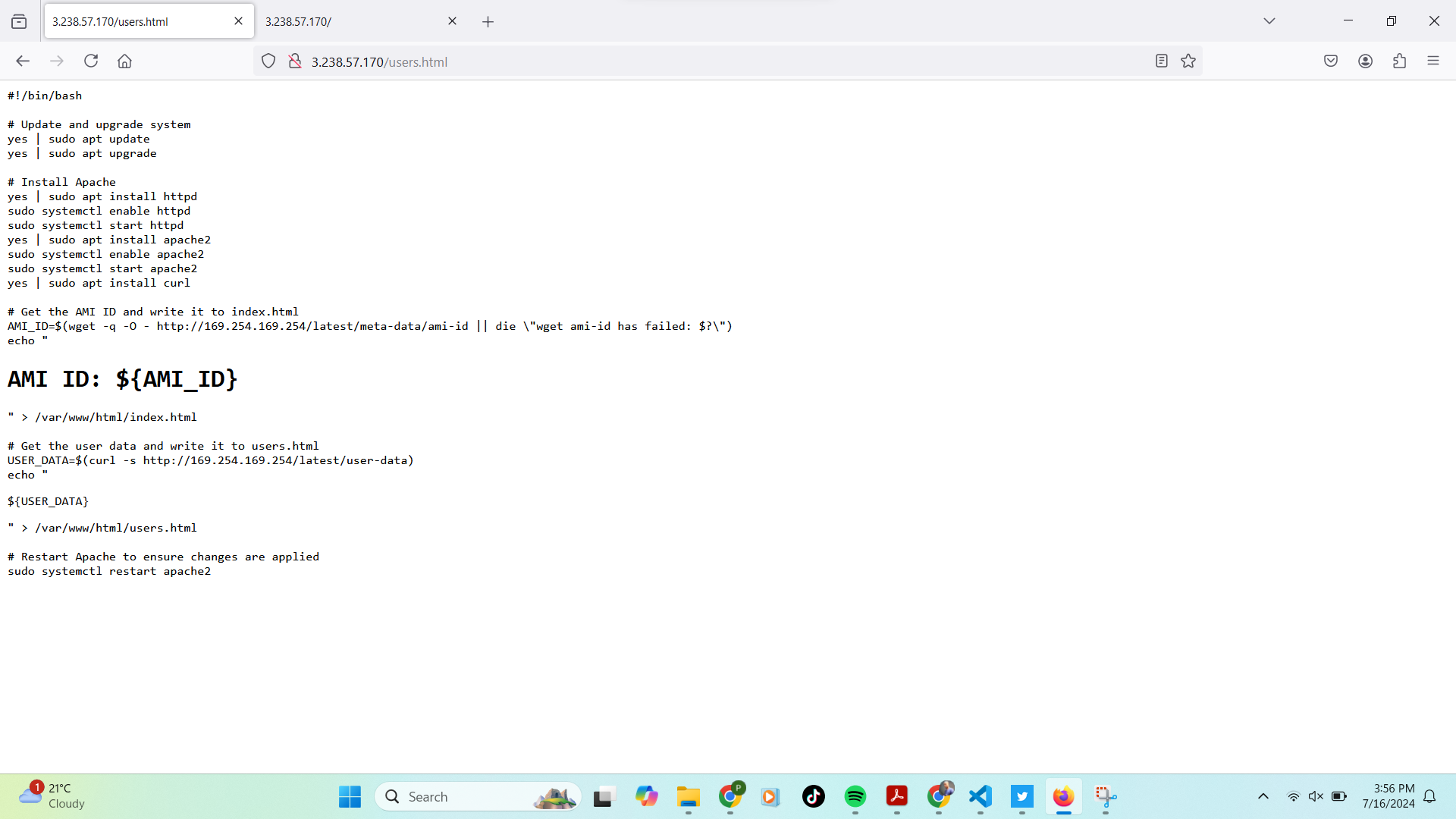
**5. Simulating the Stress Test: Can Your App Handle It?**

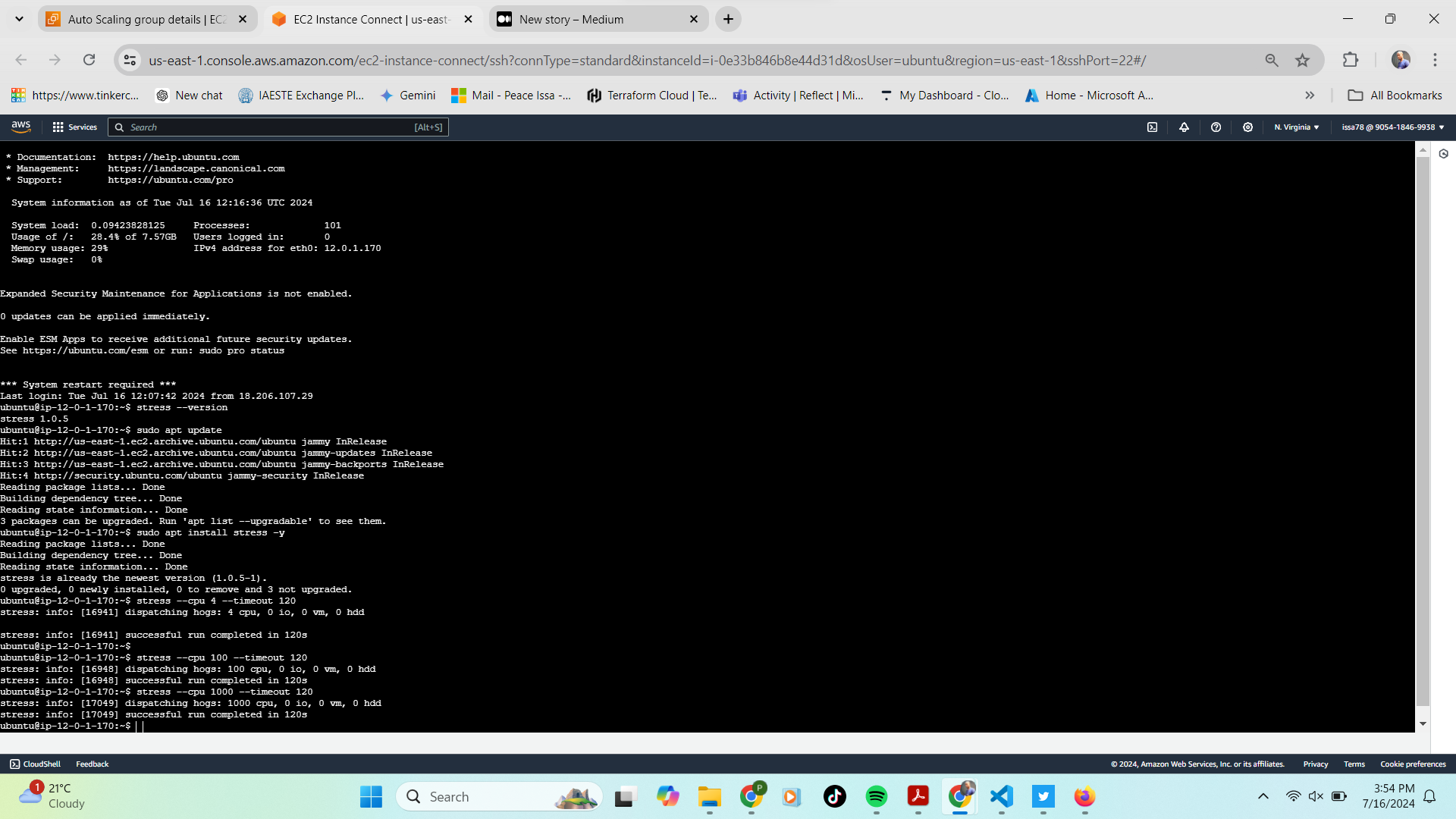
Now, let's see how your web application scales under pressure!

1. Connect to one of your EC2 instances using the public IPv4 address to check the AMI id and user data.

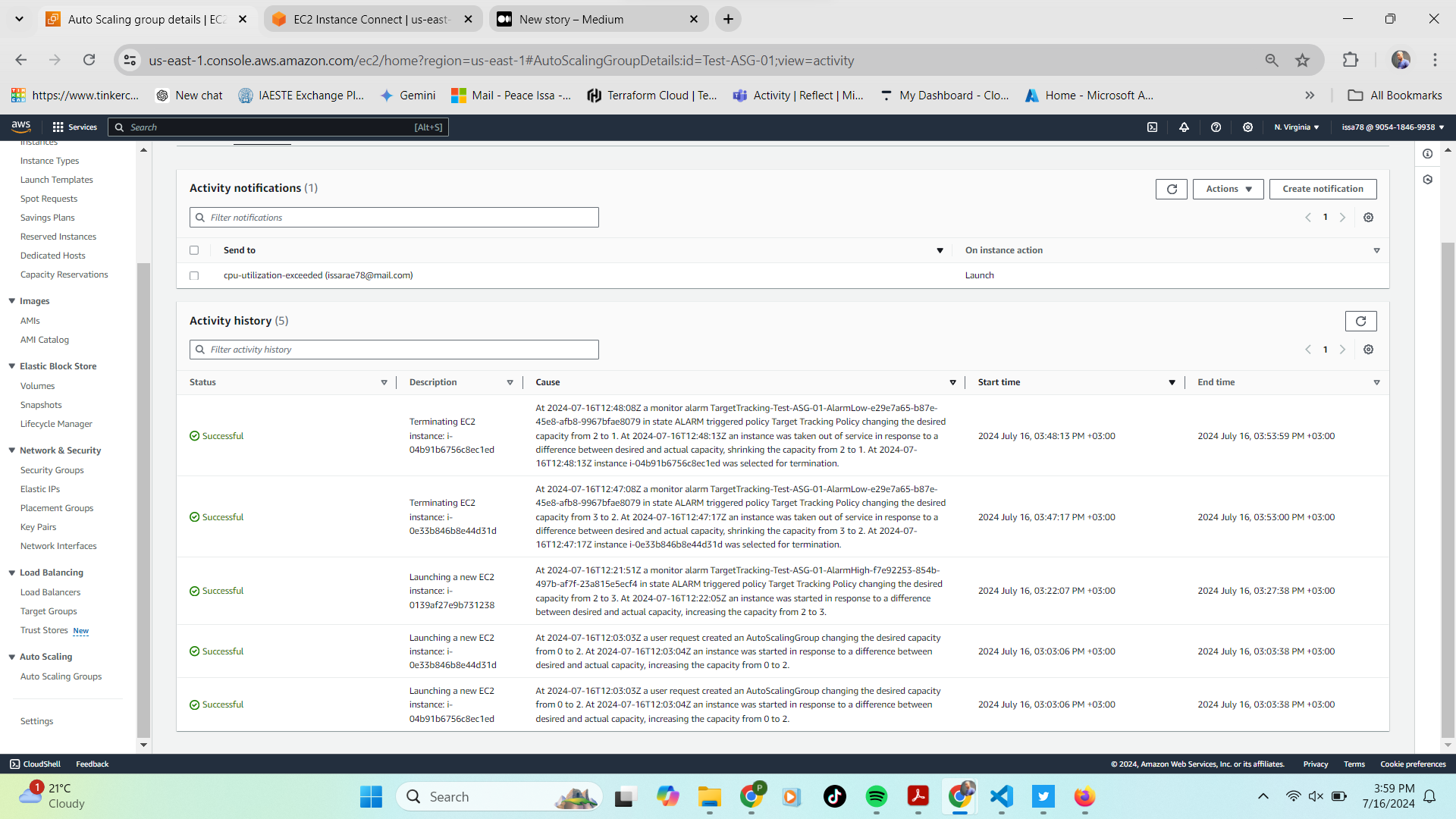
A screenshot of a computer

Description automatically generated

1. Add users.html to the url to access the user data page  
   
2. Connect to one of your EC2 instances using SSH or Ec2 instance connect.
3. Install the AWS Stress Test Utility to simulate high CPU usage.



1. Run the stress test for a few minutes and observe your auto scaling activity.



**Head back to the Auto Scaling Group in the EC2 Dashboard.** You should see the scaling policy kicking in and adding new servers to handle the increased load. Pretty neat, huh?

**6. Conclusion: You've Built a Scalable Web App!**

Congratulations! You've successfully deployed a highly available and scalable web application architecture on AWS.

**Bonus Tip:** Don't forget to check out the resources section below for further learning and troubleshooting. Feel free to leave a comment and share your experiences with building scalable applications on AWS!

**Contact:**

* **LinkedIn**: [LinkedIn](http://www.linkedin.com/in/peace-issa)
* **GitHub:** <https://github.com/peaceissa>
* **Email:** issarae78@gmail.com

**Resources:**

* AWS Documentation
* Youtube