Overview of Task: Deploying a Scalable Web Application on AWS

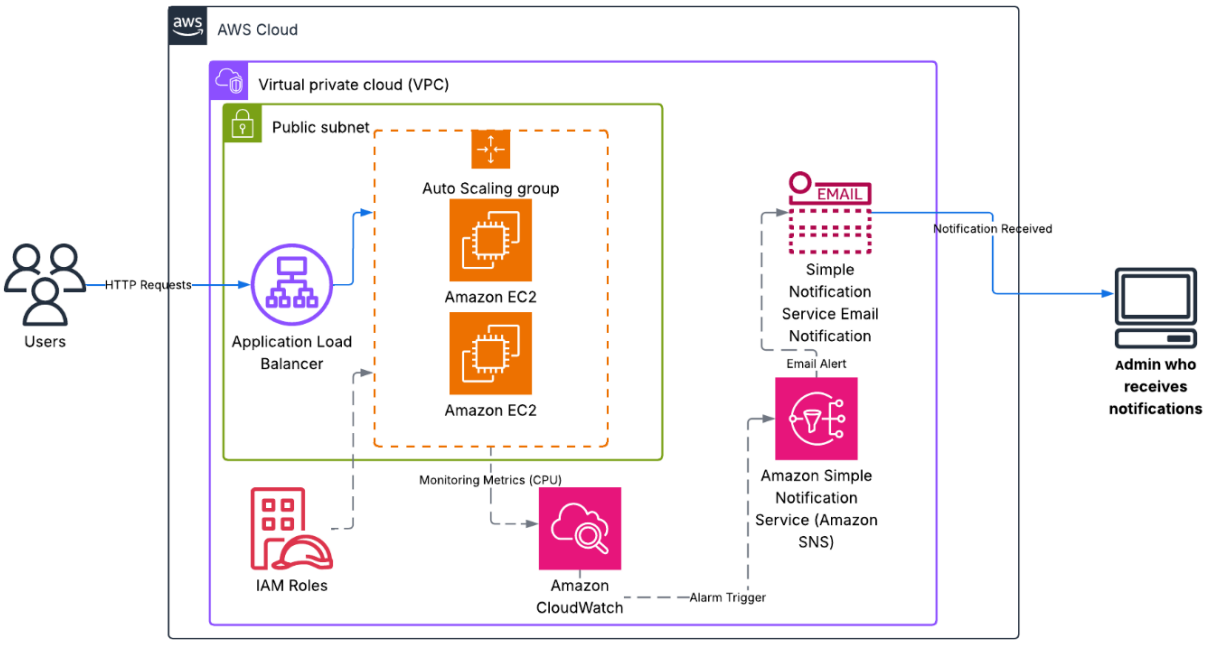
This document provides a comprehensive, step-by-step guide for deploying a highly available, fault-tolerant, and scalable web application on Amazon Web Services (AWS). The primary goal is to establish an infrastructure that automatically adapts to varying traffic loads, efficiently distributes incoming requests, and offers robust monitoring and alerting capabilities.

The deployment leverages a suite of integrated AWS services to achieve these objectives:

- Amazon Virtual Private Cloud (VPC): To create a secure and isolated network environment.
- Amazon EC2 (Elastic Compute Cloud): For provisioning and managing virtual servers that host the web application.
- Amazon Machine Image (AMI): To standardize the web server configuration for consistent and rapid instance deployments.
- EC2 Launch Templates: To define the instance specifications used by Auto Scaling.
- Auto Scaling Groups: To automatically adjust the number of EC2 instances based on demand, ensuring application availability and performance.
- Application Load Balancer (ALB): To efficiently distribute incoming web traffic across multiple EC2 instances, enhancing reliability and scalability.
- Amazon CloudWatch: For real-time monitoring of resource utilization and health, enabling proactive issue detection.
- Amazon Simple Notification Service (SNS): To deliver automated email alerts for critical events, such as scaling actions or performance thresholds being breached.
- IAM Roles: To securely grant necessary permissions to AWS services, adhering to the principle of least privilege

1: AWS Cloud Architecture Diagram

This foundational image presents the overall architecture of the web application within the AWS Cloud. It visually depicts how user requests are routed through an Application Load Balancer to EC2 instances managed by an Auto Scaling Group within a Public Subnet of a Virtual Private Cloud (VPC). The diagram also highlights the integration of Amazon CloudWatch for monitoring and Amazon SNS for email notifications to the administrator, providing a clear blueprint of the entire system

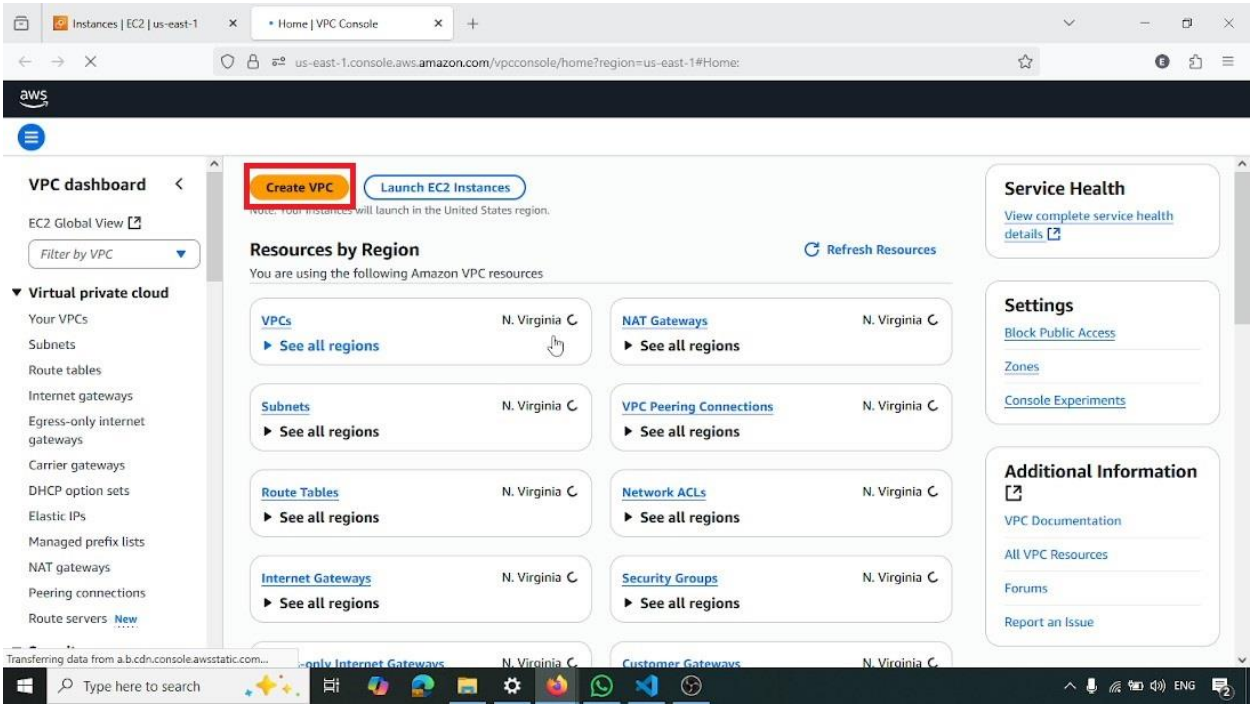


Step 1: VPC and Network Infrastructure Setup

This initial phase focuses on establishing the fundamental network environment within AWS, including the Virtual Private Cloud (VPC), subnets, Internet Gateway, and route tables, which are essential for secure and connected resources

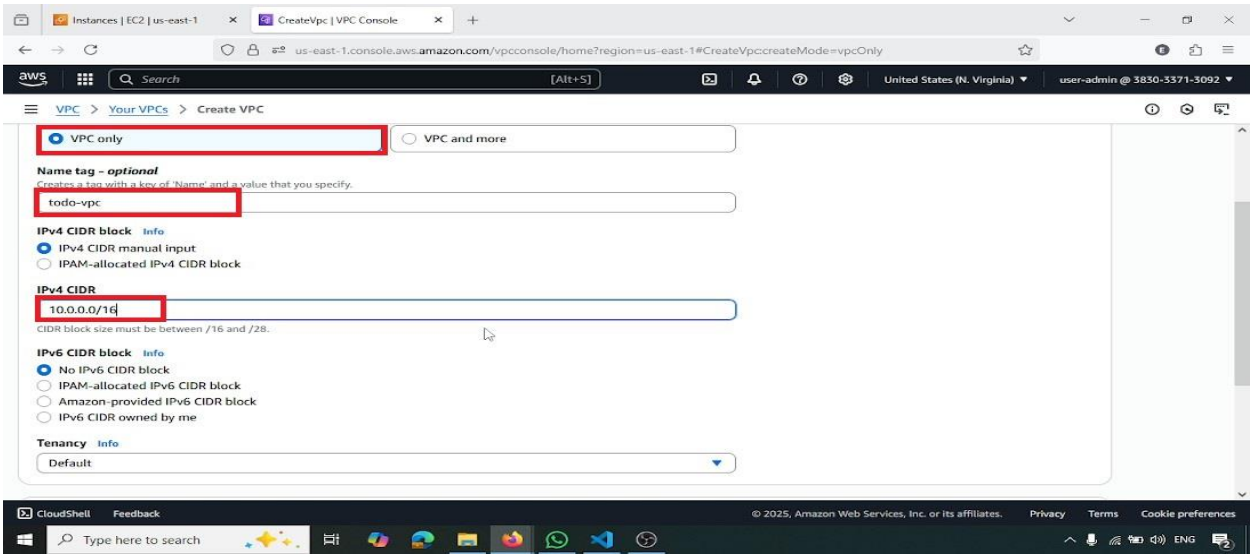
Initiating VPC Creation

This screenshot from the AWS VPC dashboard shows the "Create VPC" button highlighted, indicating the starting point for creating a new Virtual Private Cloud. This action begins the process of defining an isolated network space for the application



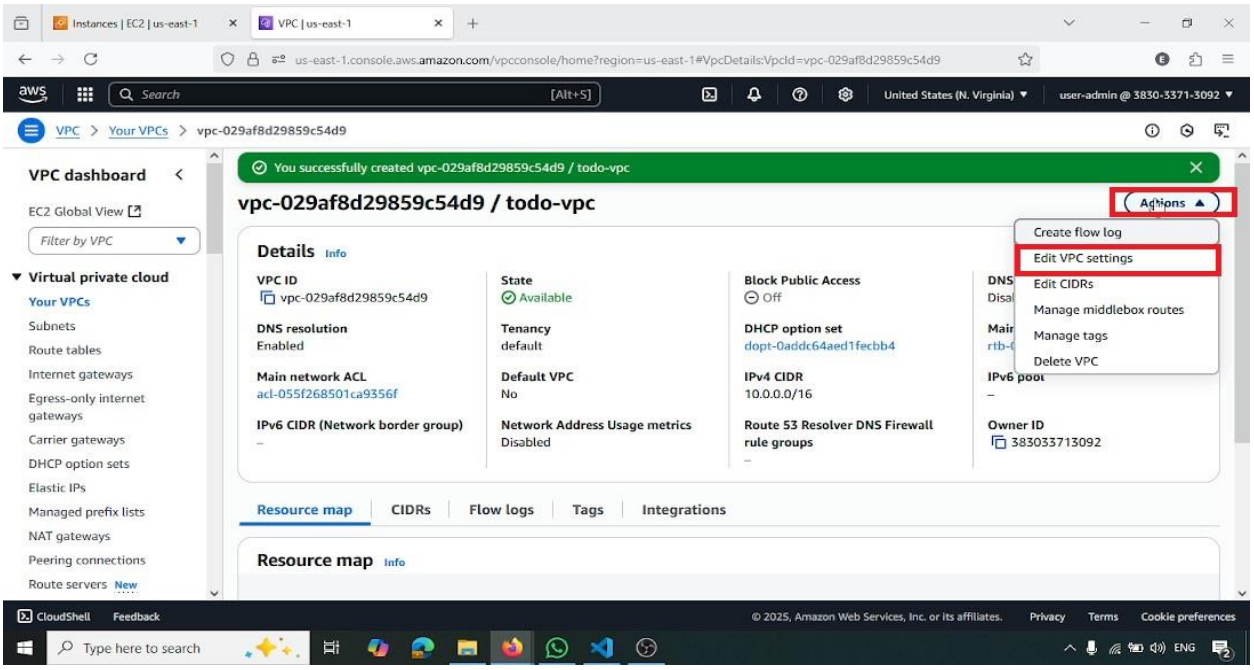
Creating VPC with CIDR Block

Here, the "Create VPC" wizard is shown with "VPC only" selected. The VPC is named todo-vpc, and its IPv4 CIDR block is set to 10.0.0.0/16. This step defines the network's IP address range and creates the VPC



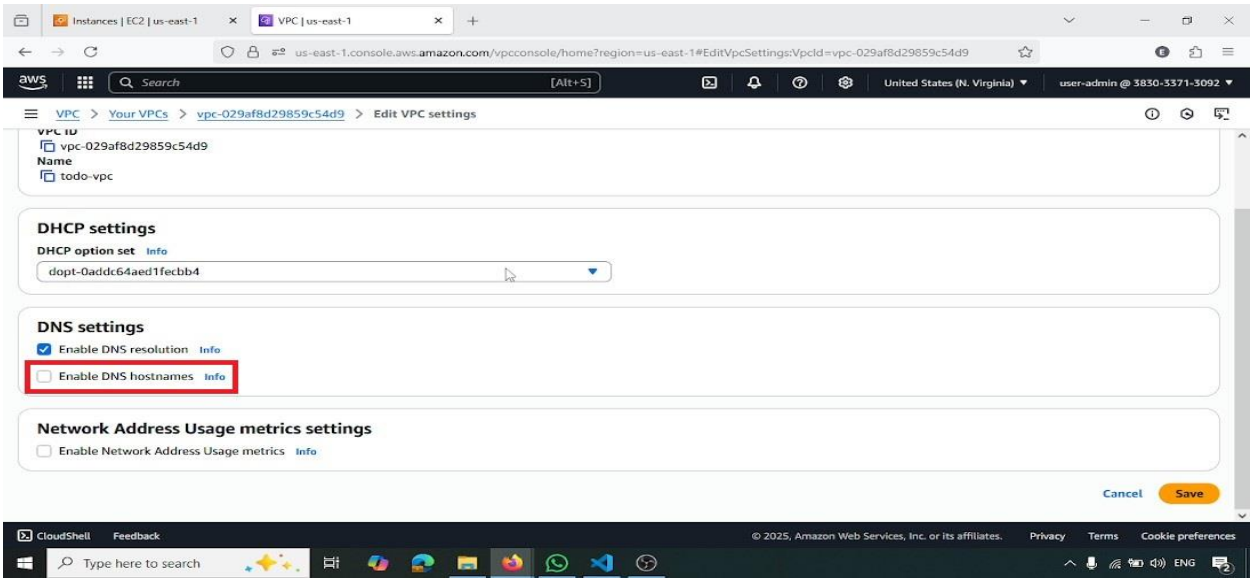
Editing VPC Settings

This image displays the "Actions" dropdown within the VPC console, with "Edit VPC settings" highlighted. This action is taken to modify the VPC's configuration after its initial creation



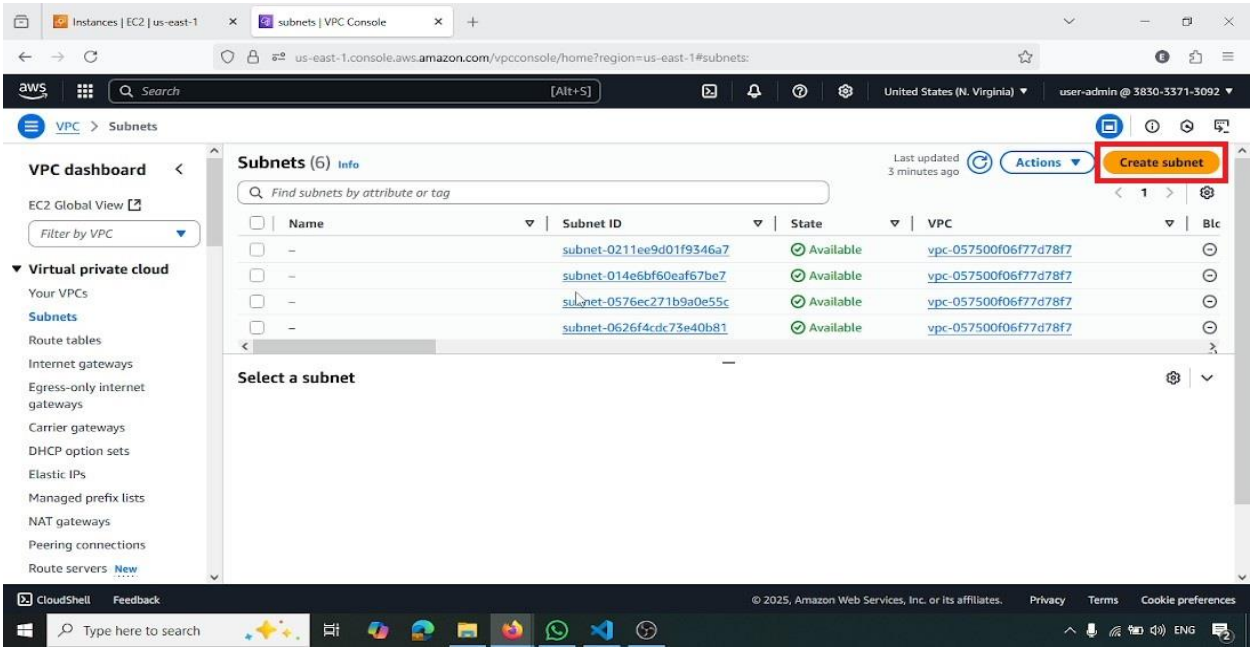
Enabling DNS Hostnames for VPC

Within the VPC settings, the "Enable DNS hostnames" checkbox is highlighted and selected. This ensures that instances launched within this VPC will automatically receive public DNS hostnames, facilitating easier access and management



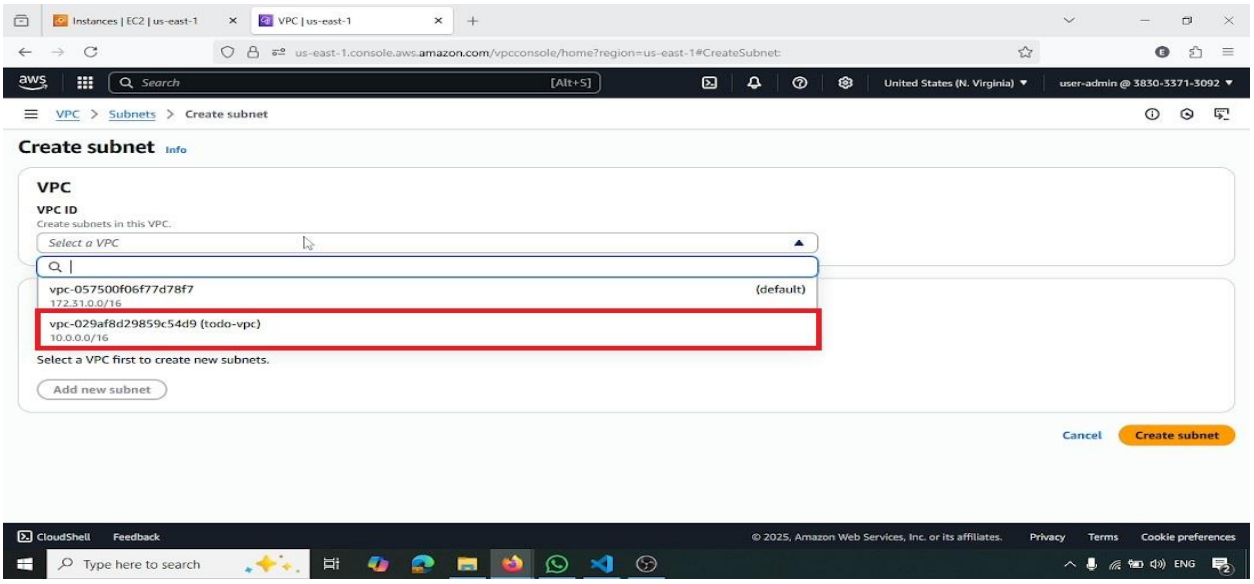
Initiating Subnet Creation

This screenshot from the AWS VPC console, under the "Subnets" section, shows the "Create subnet" button highlighted. This action initiates the process of creating subnets within the newly defined VPC



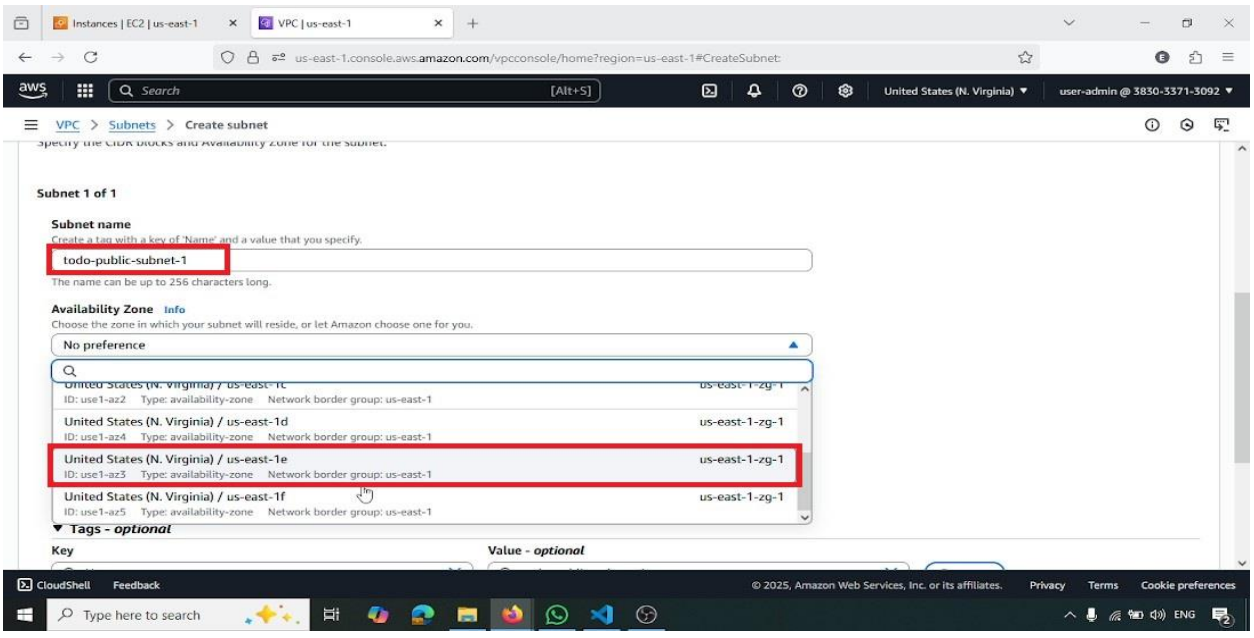
Selecting VPC for Subnet Creation

In the "Create subnet" wizard, the todo-vpc (vpc-029af8d29859c54d9) is selected from the "VPC ID" dropdown. This ensures that the new subnet is provisioned within the correct VPC



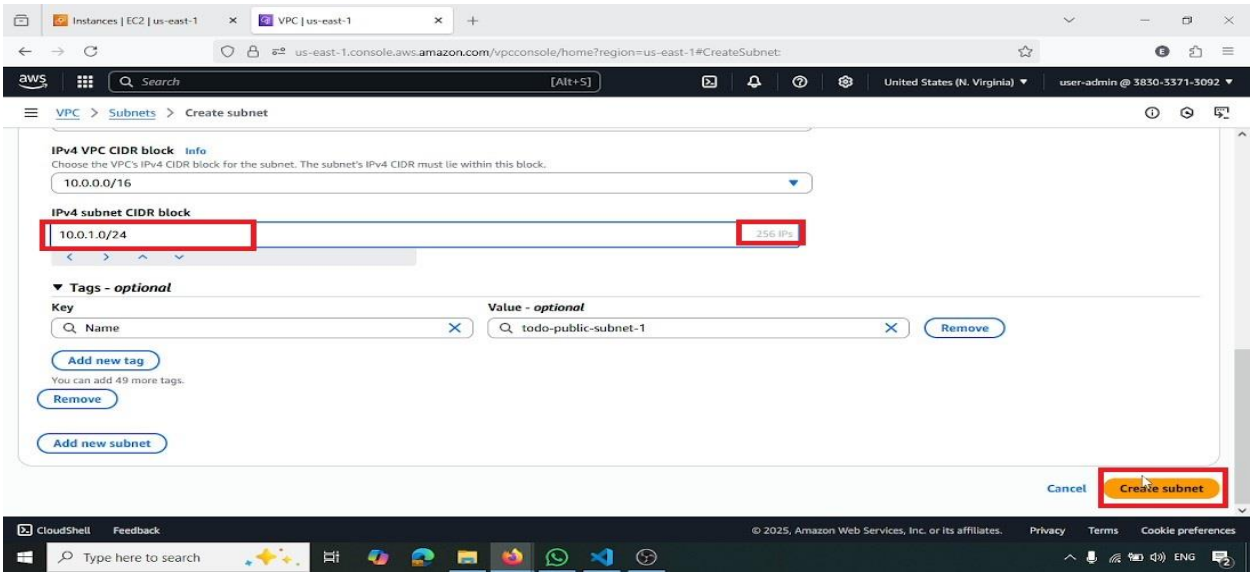
Selecting Availability Zone for First Public Subnet

The "Availability Zone" dropdown is shown with us-east-1e highlighted. This selection specifies the physical location for the first public subnet, contributing to the application's high availability



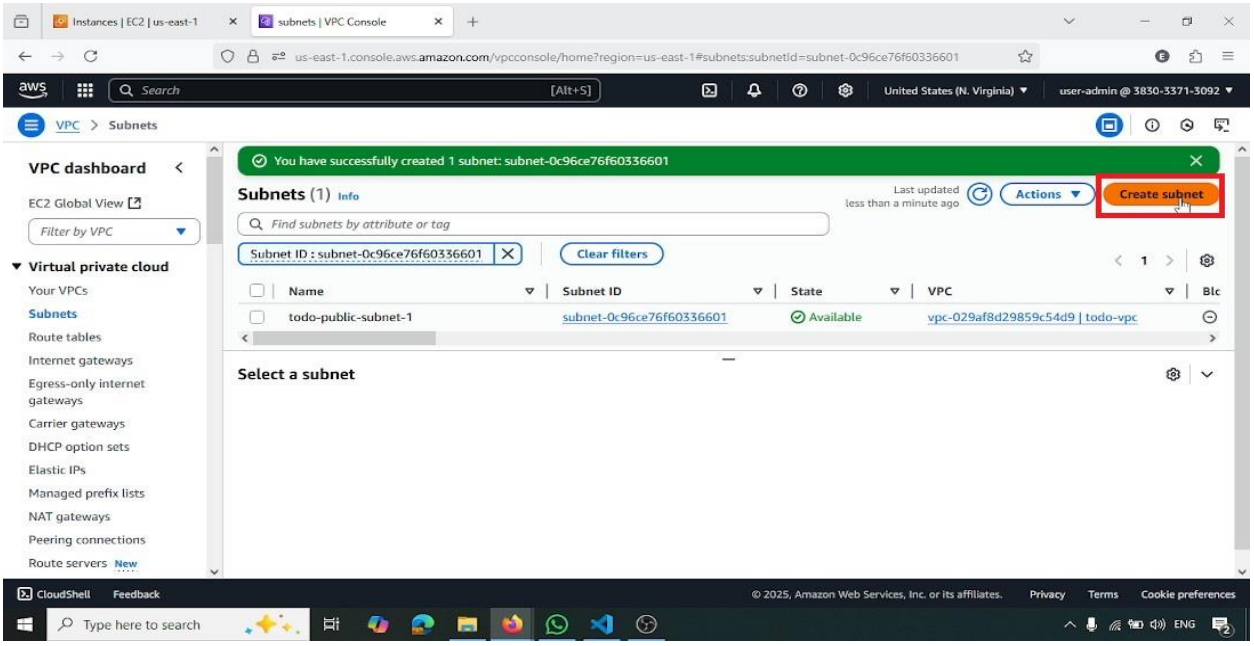
Creating First Public Subnet

The "Create subnet" wizard displays the "Subnet name" as todo-public-subnet-1 and the "IPv4 subnet CIDR block" as 10.0.1.0/24. This action creates the first public subnet, which will host internet-facing resources



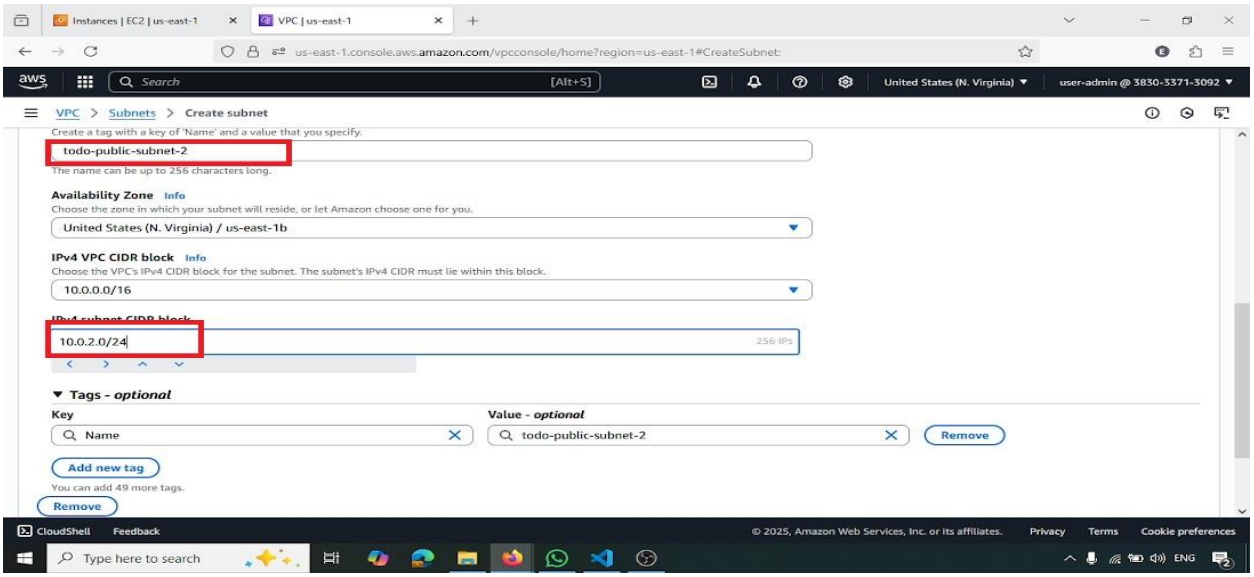
Creating Second Public Subnet

Similar to the previous step, this screenshot shows the creation of todo-public-subnet-2 with an "IPv4 subnet CIDR block" of 10.0.2.0/24. This second public subnet is created in a different Availability Zone (us-east-1b) to enhance fault tolerance



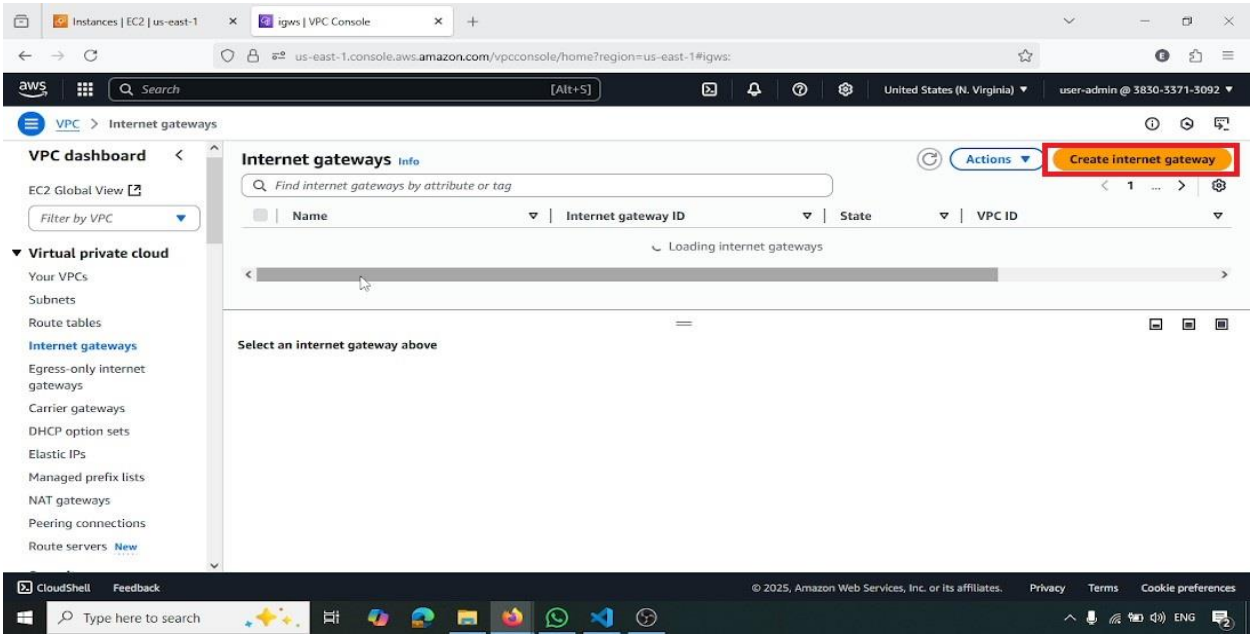
Creating Second Public Subnet

The "Create subnet" wizard displays the "Subnet name" as todo-public-subnet-2 and the "IPv4 subnet CIDR block" as 10.0.2.0/24. This action creates the first public subnet, which will host internet-facing resources



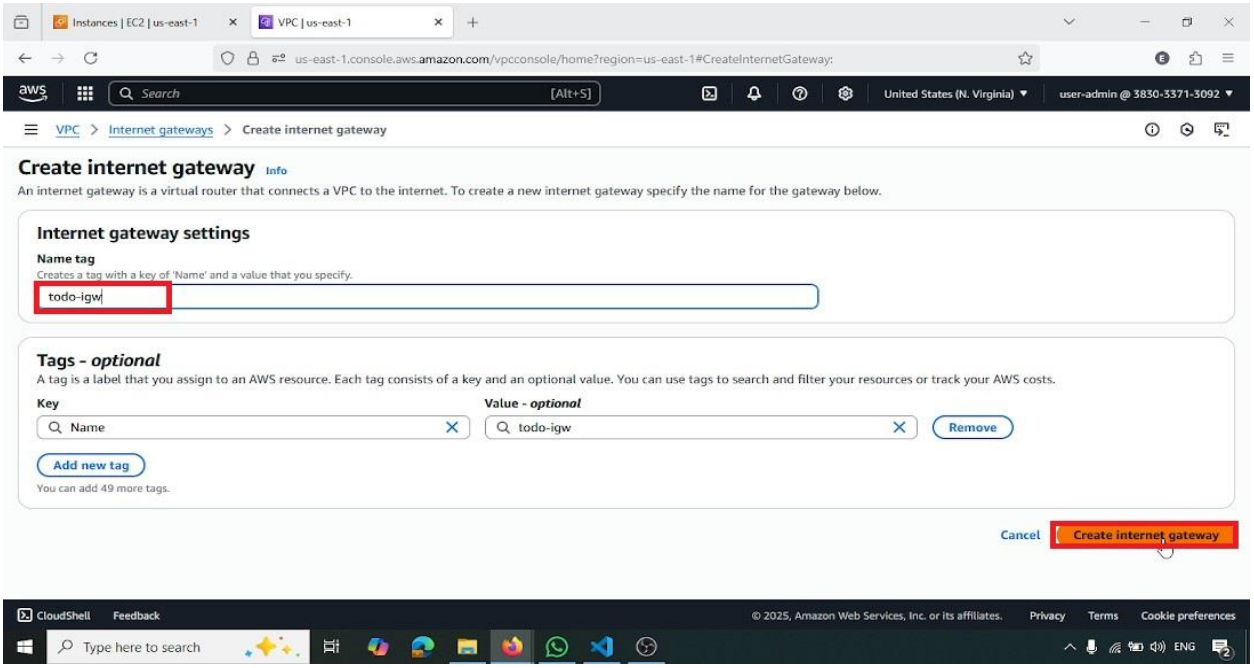
Initiating Internet Gateway Creation

From the AWS VPC console, the "Create internet gateway" button is highlighted. This action starts the process of creating a gateway that enables communication between the VPC and the internet



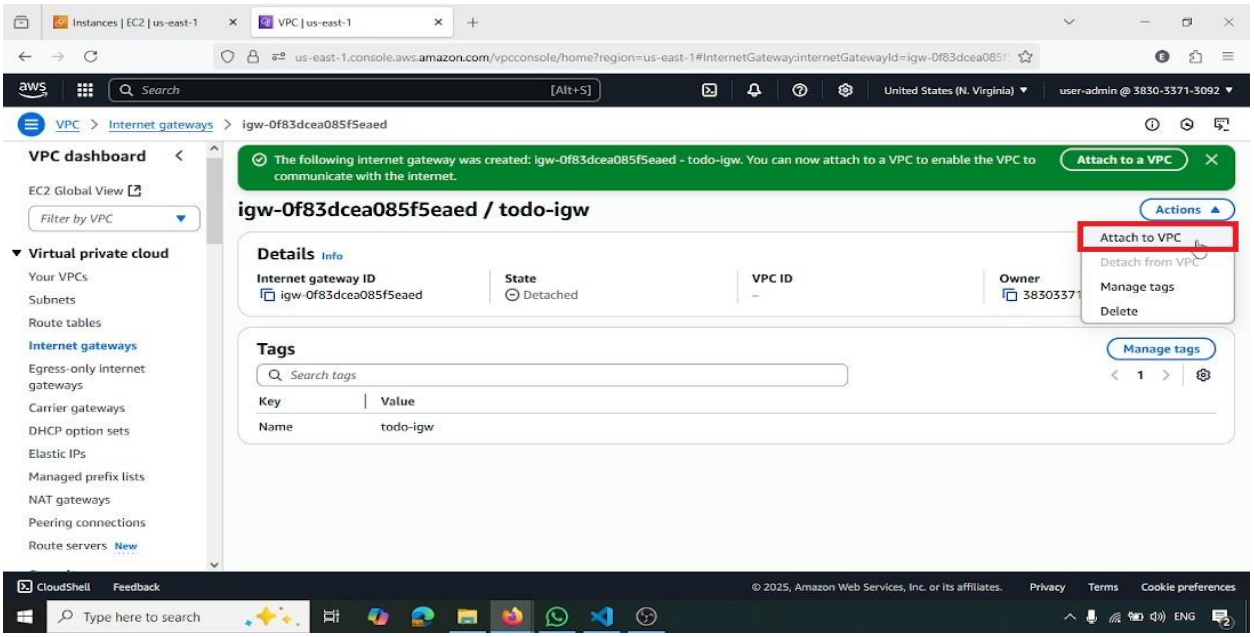
Naming Internet Gateway

The "Create internet gateway" wizard shows the "Name tag" entered as todo-igw. This step assigns a descriptive name to the internet gateway



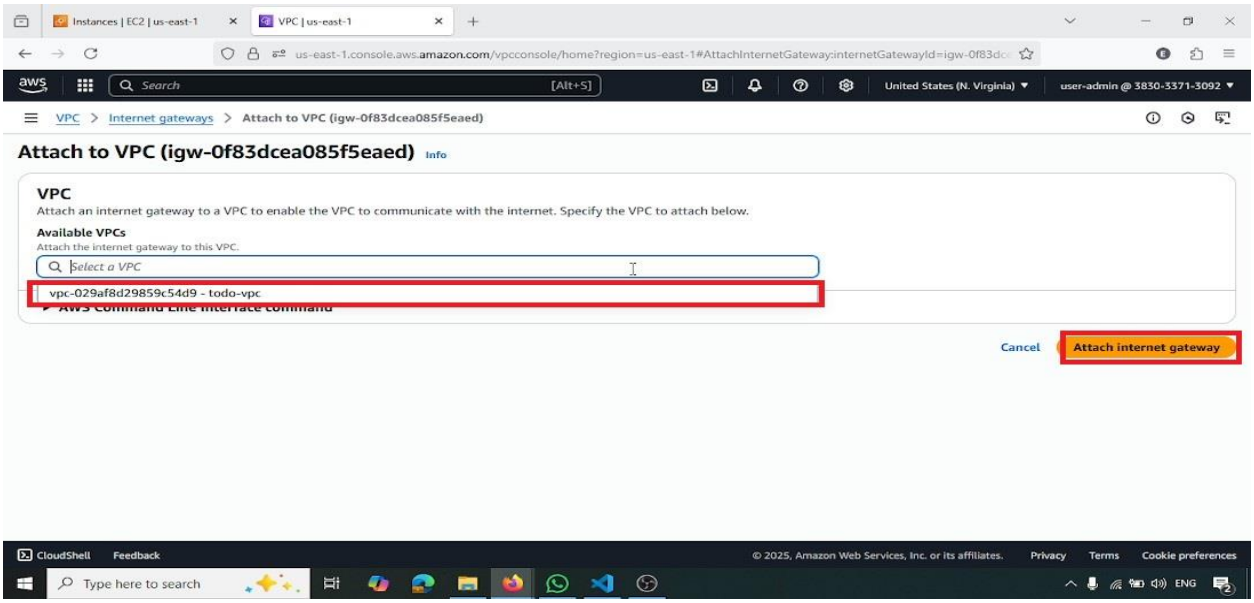
Attaching Internet Gateway Action

This image shows the "Actions" dropdown for the newly created internet gateway, with "Attach to VPC" highlighted. This action is taken to link the internet gateway to the todo-vpc



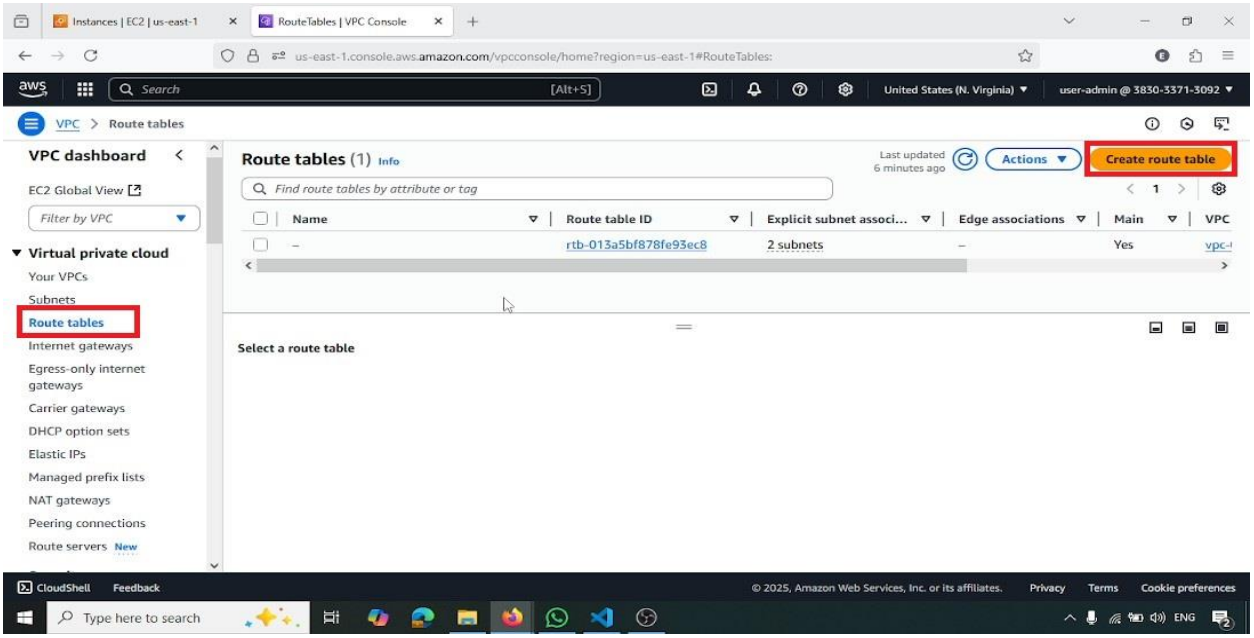
Attaching Internet Gateway to VPC

In the "Attach to VPC" wizard, the todo-vpc (vpc-029af8d29859c54d9) is selected. Clicking "Attach internet gateway" connects the gateway, allowing internet access for resources within the VPC



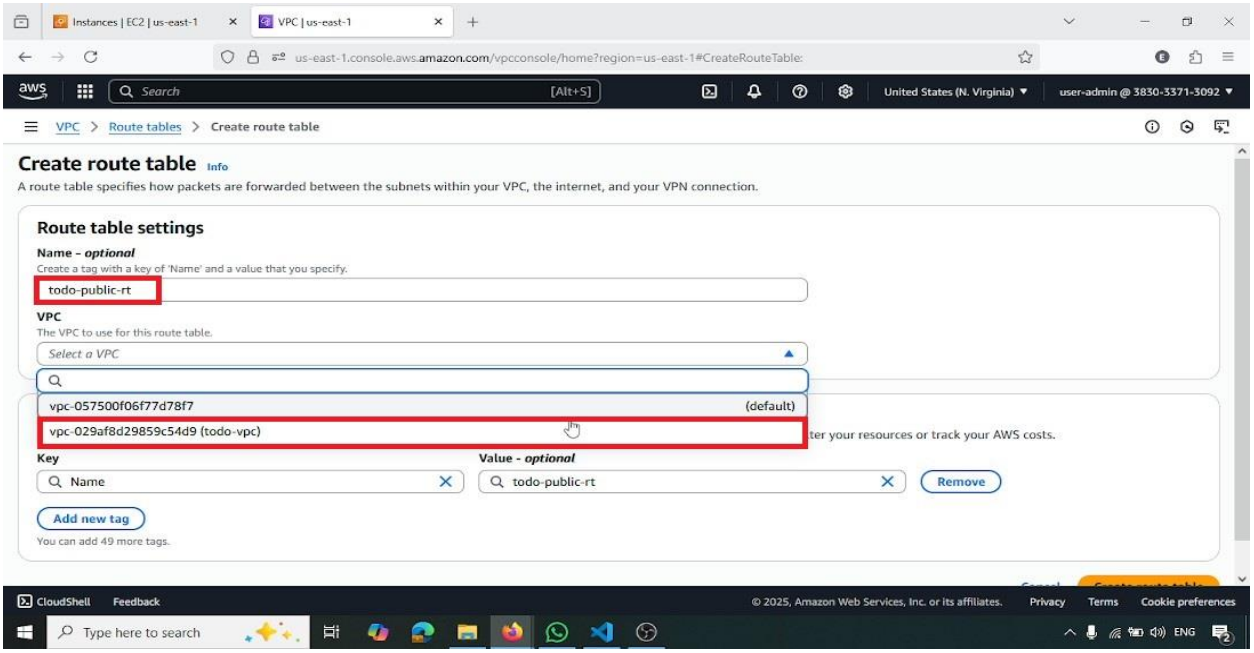
Initiating Route Table Creation

From the AWS VPC console, the "Create route table" button is highlighted. This action begins the process of defining custom routing rules for network traffic within the VPC



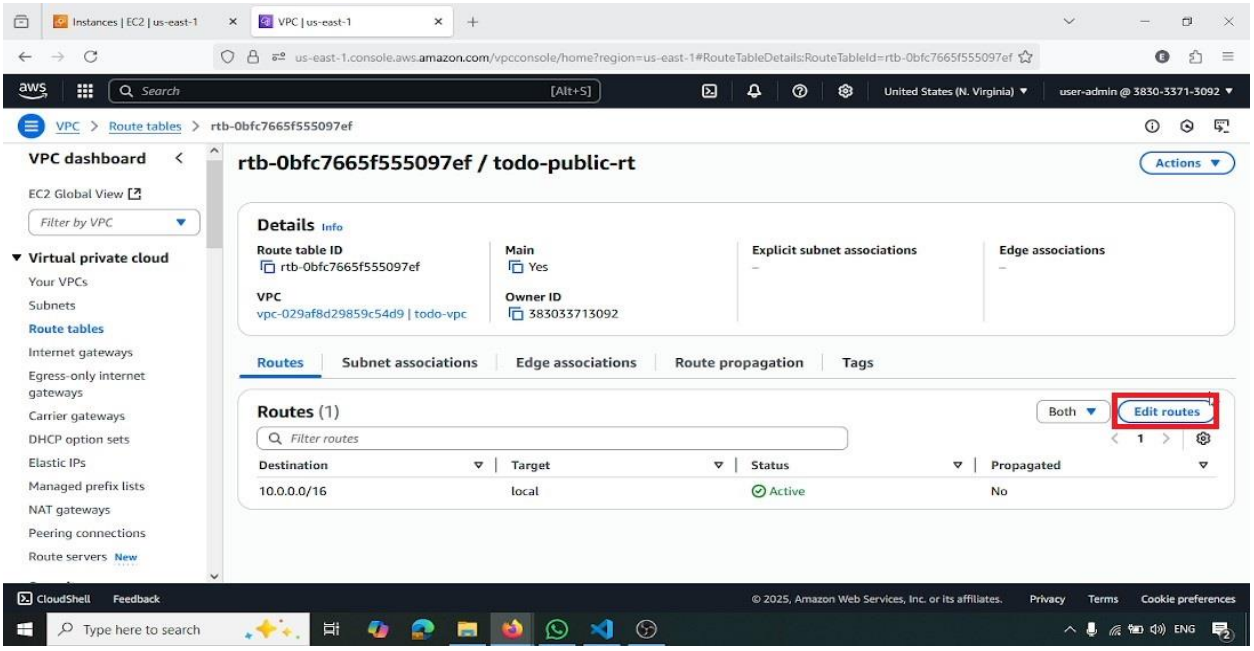
Creating Public Route Table

The "Create route table" wizard shows the "Name tag" as todo-public-rt and the "VPC" selected as todo-vpc. This creates a dedicated route table for the public subnets



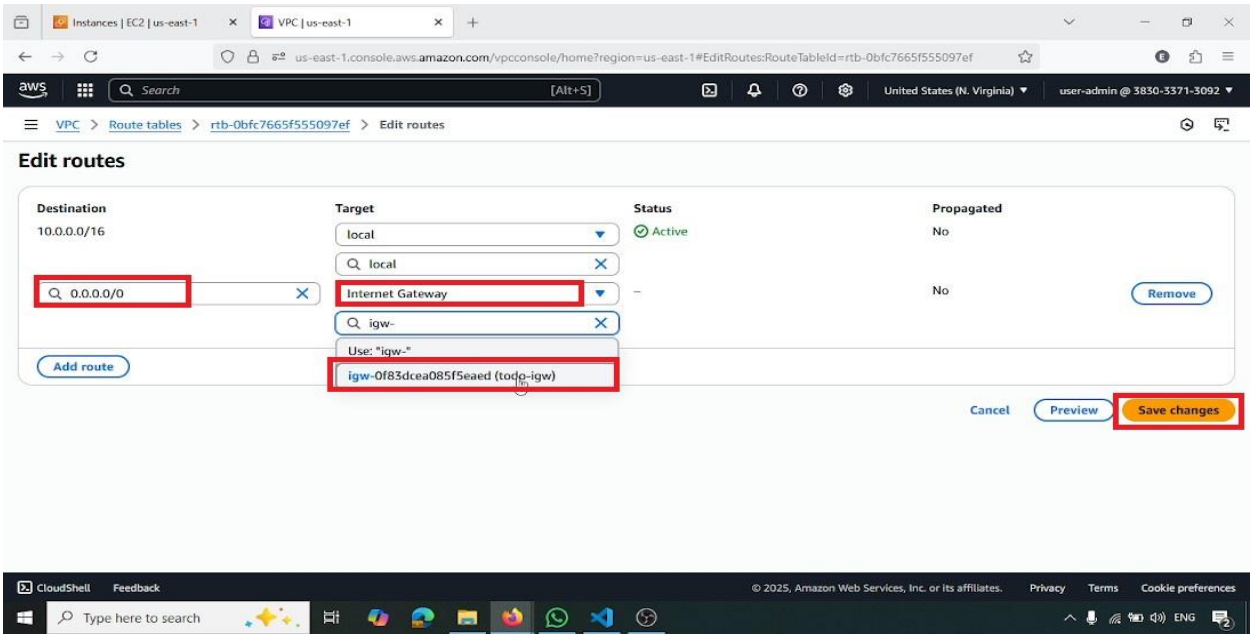
Editing Routes for Public Route Table

This screenshot shows the "Edit routes" button highlighted within the todo-public-rt details. This action is taken to add or modify routing rules for the public route table



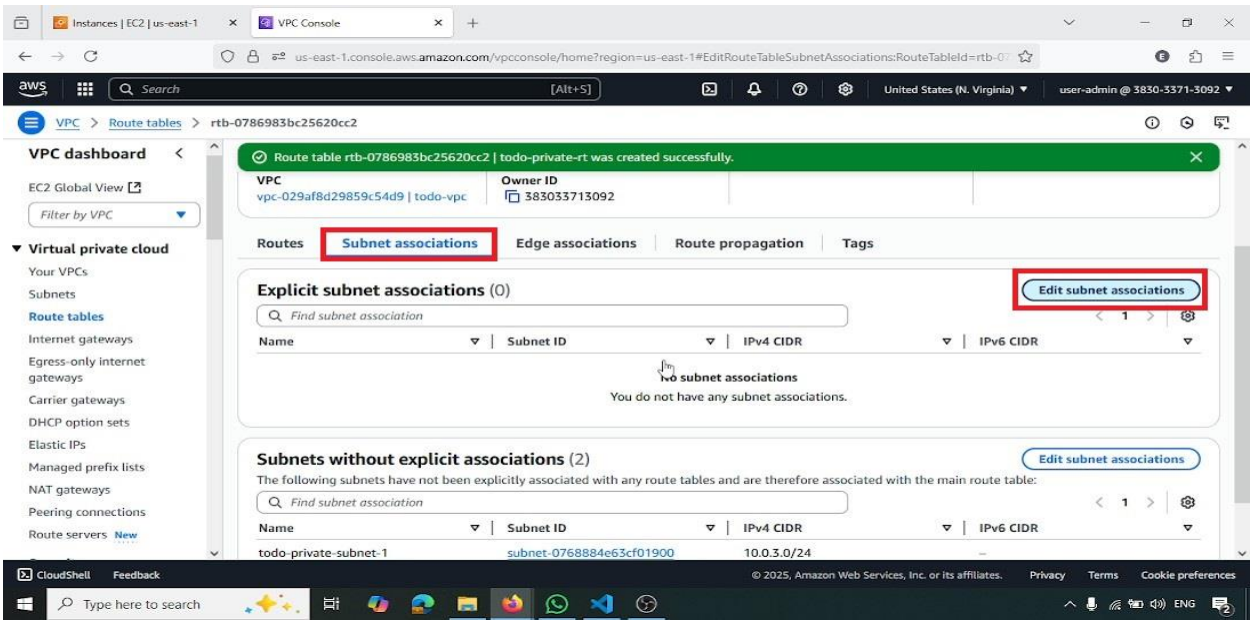
Adding Internet Gateway Route to Public Route Table

A new route is added to todo-public-rt. The "Destination" is set to 0.0.0.0/0 (representing all internet traffic), and the "Target" is the todo-igw internet gateway. This ensures that all outbound internet traffic from the public subnets is routed correctly



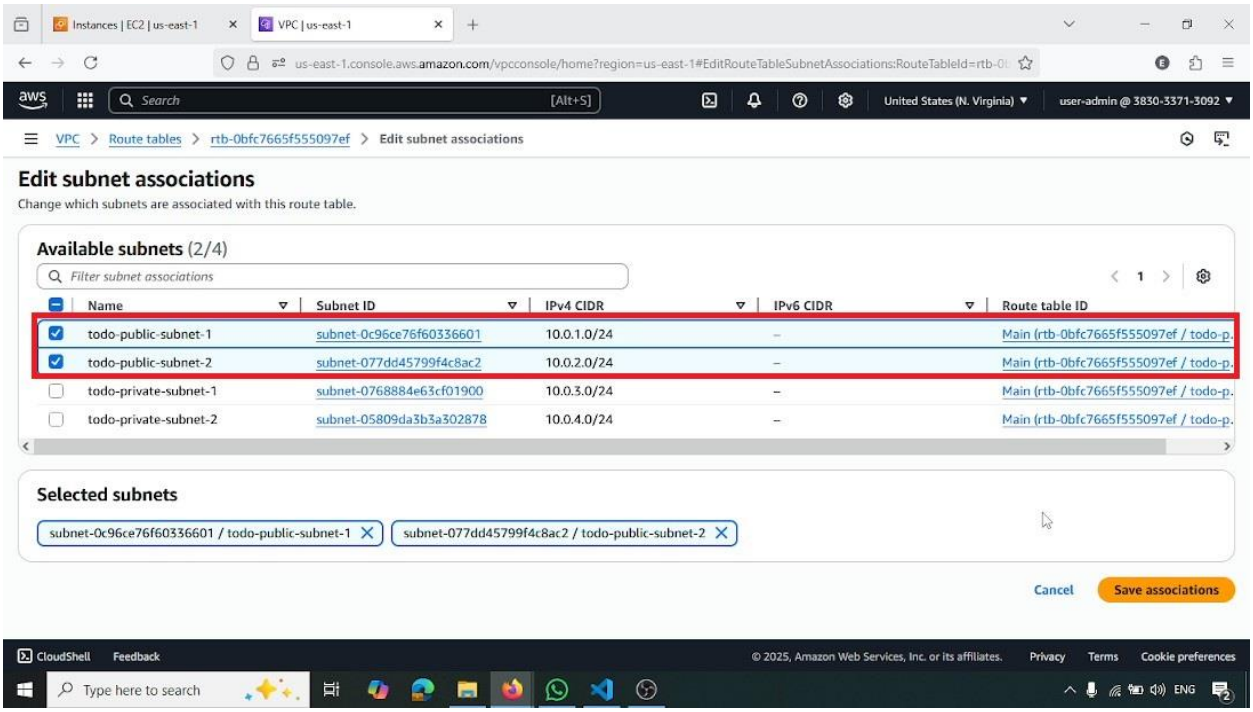
Editing Subnet Associations for Route Table

This image shows the "Edit subnet associations" button highlighted within the todo-public-rt details. This action is taken to link the public subnets to this route table



Associating Subnets with Public Route Table

The two public subnets, todo-public-subnet-1 and todo-public-subnet-2, are checked for association with the todo-public-rt route table. "Save associations" is highlighted, confirming that these subnets will now use this route table for their routing

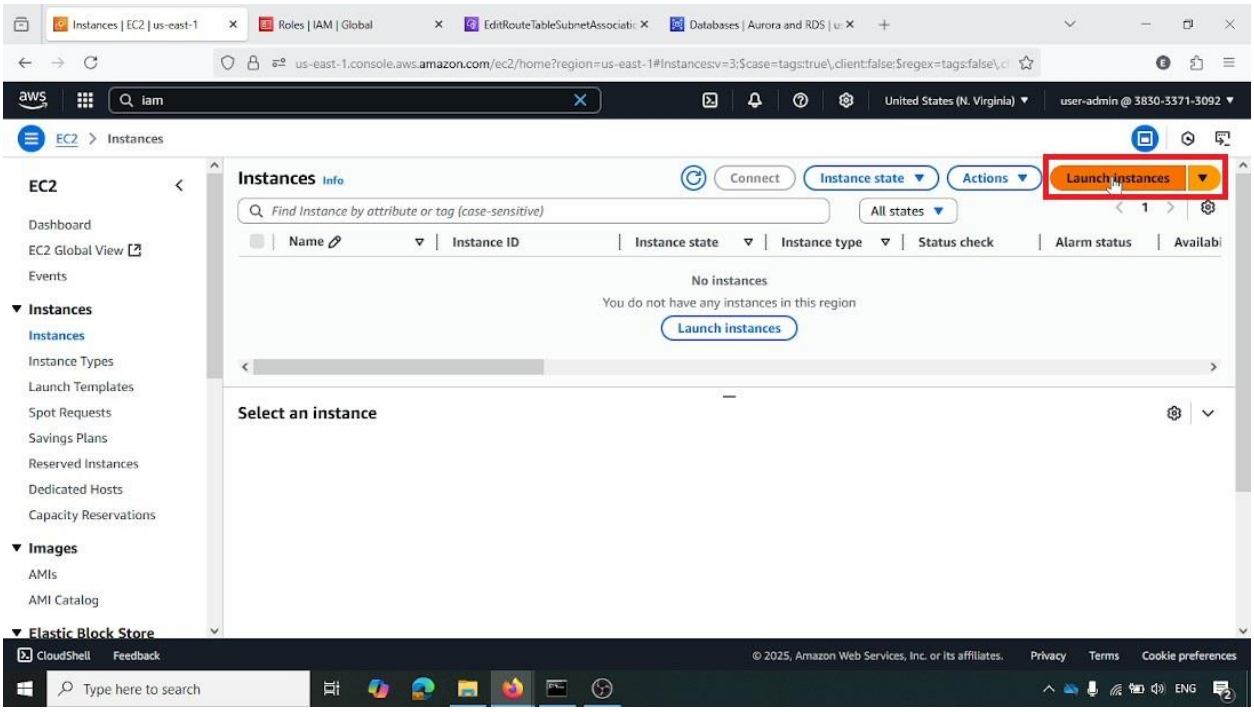


Step 2: EC2 Instance and AMI Preparation

This phase involves launching an initial EC2 instance, configuring it with the web application, creating a custom Amazon Machine Image (AMI) from it, and then preparing a launch template for Auto Scaling.

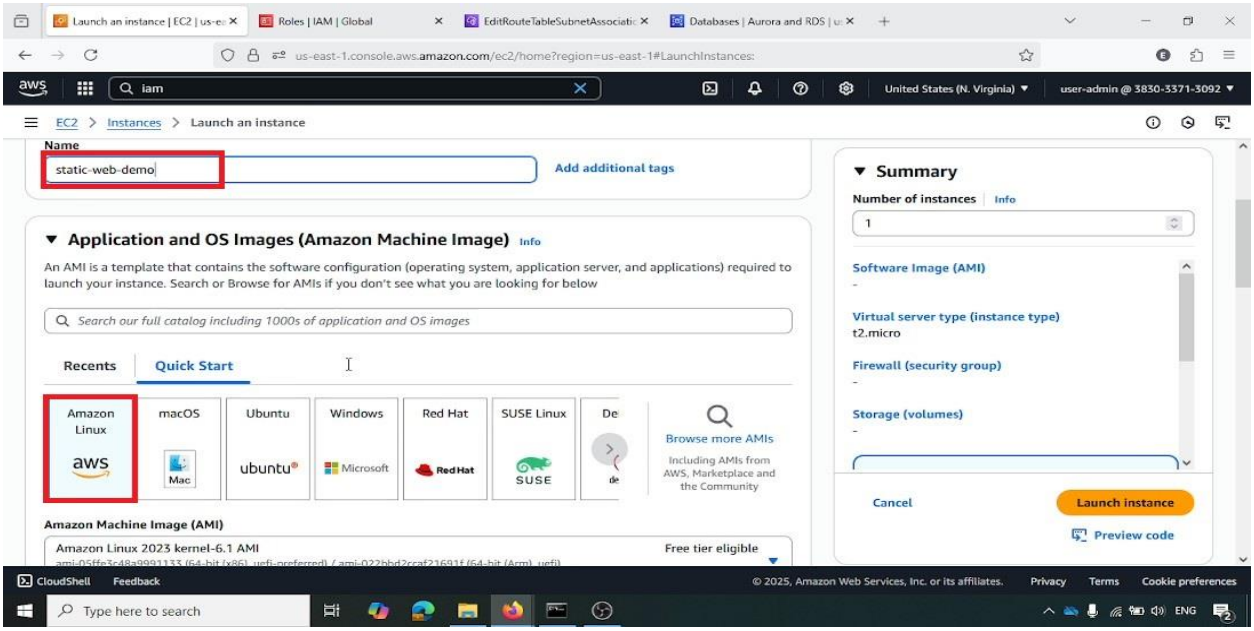
Initiating EC2 Instance Launch

From the AWS EC2 console, the "Launch instances" button is highlighted, initiating the process of creating a new virtual server



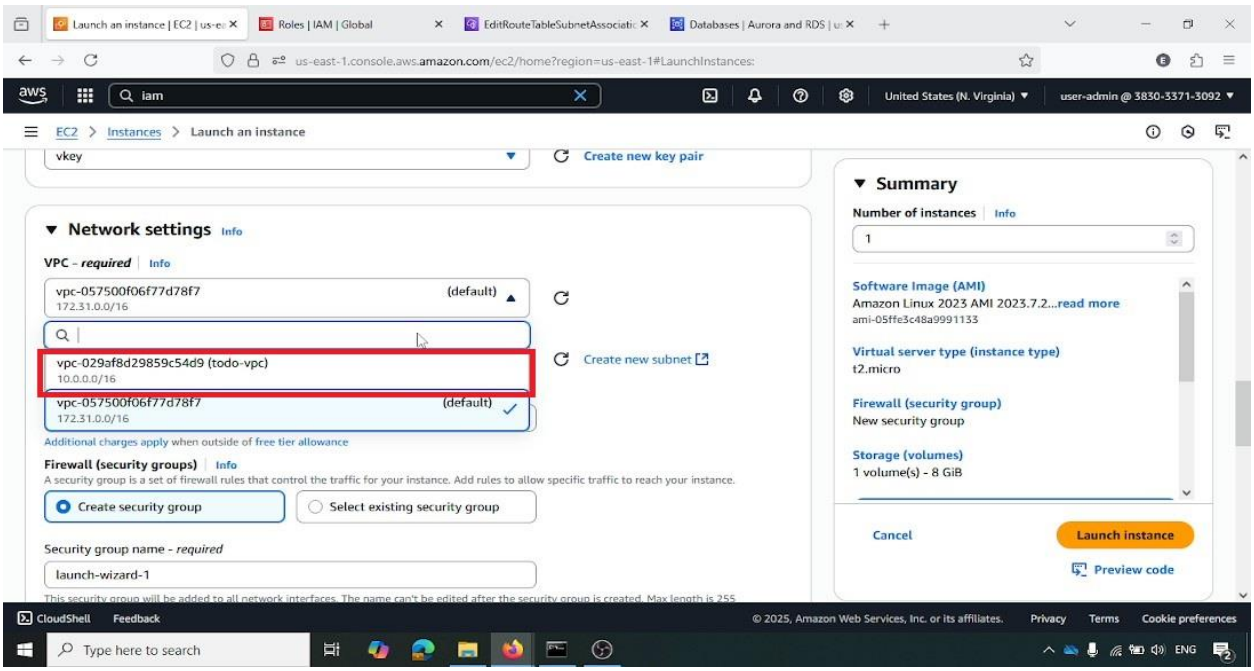
Naming Instance and Selecting AMI

The "Launch an instance" wizard shows the instance named static-web-demo and "Amazon Linux" selected as the base operating system image



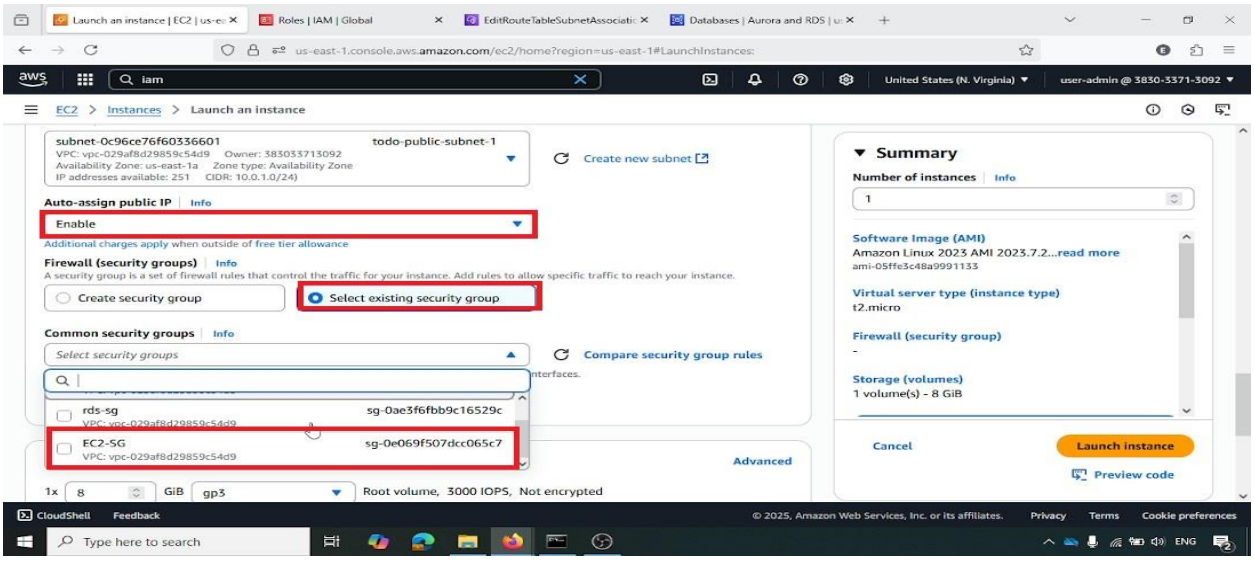
Selecting VPC for Instance Launch

The todo-vpc (vpc-029af8d29859c54d9) is selected from the "VPC" dropdown, ensuring the instance is launched within the correct network



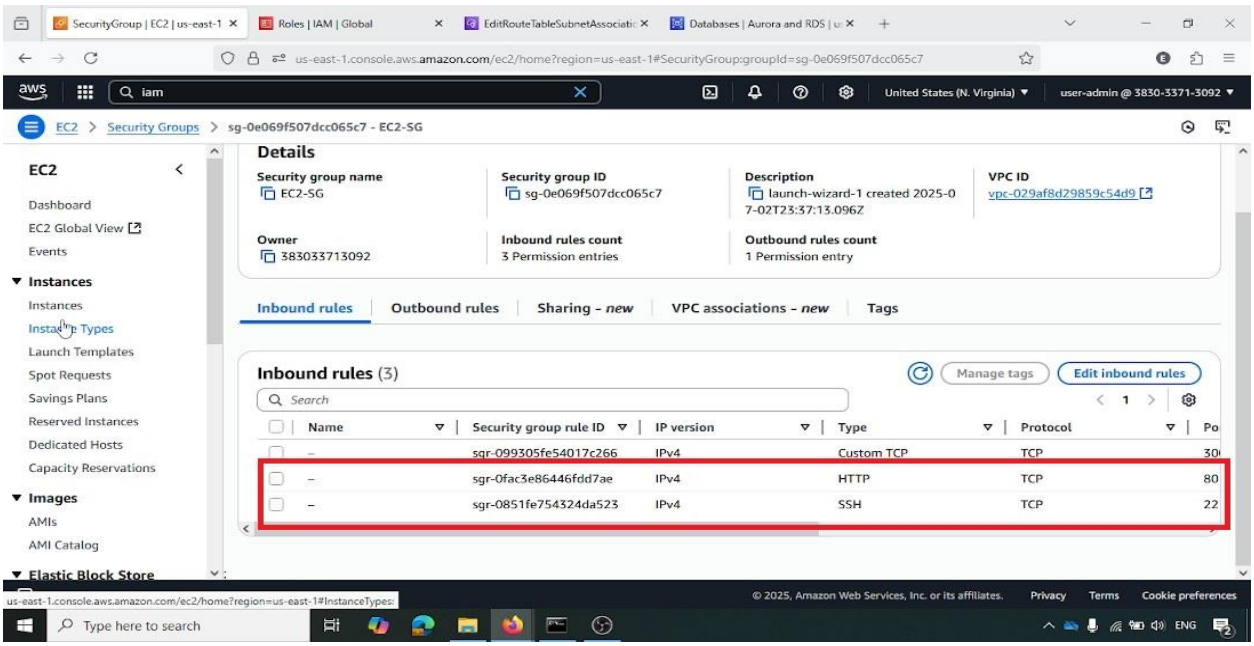
Selecting Security Group for Instance Launch

The EC2-SG security group (sg-0e069f507c0c065c7) is highlighted and selected. This security group controls inbound and outbound traffic for the instance



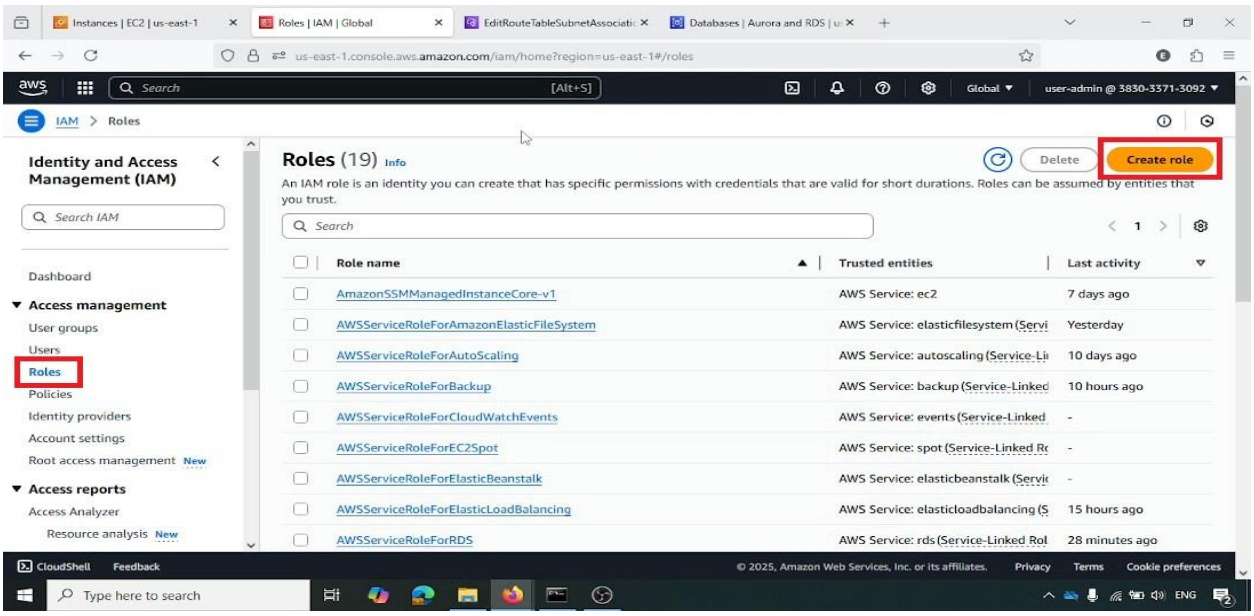
Security Group Inbound Rules

This screenshot shows the inbound rules for EC2-SG, specifically allowing HTTP traffic on Port 80 and SSH traffic on Port 22 from all IPv4 sources. This enables web access and secure remote management



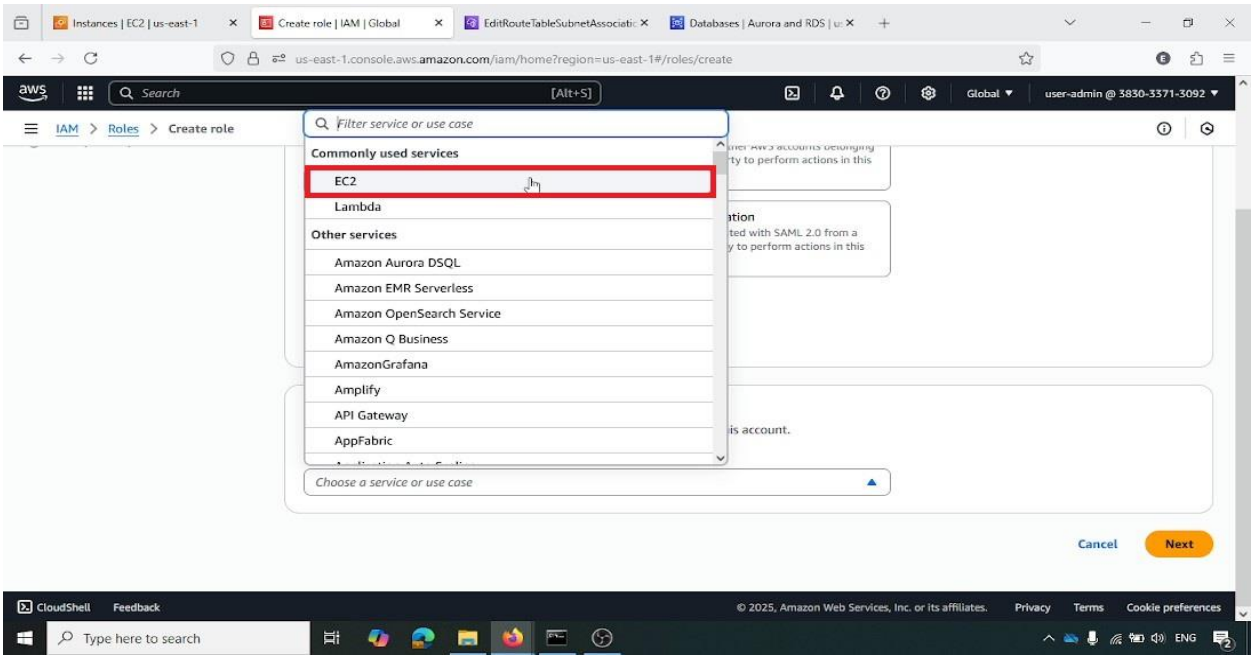
Initiating IAM Role Creation

From the AWS IAM console, the "Create role" button is highlighted, starting the process of creating a new Identity and Access Management (IAM) role



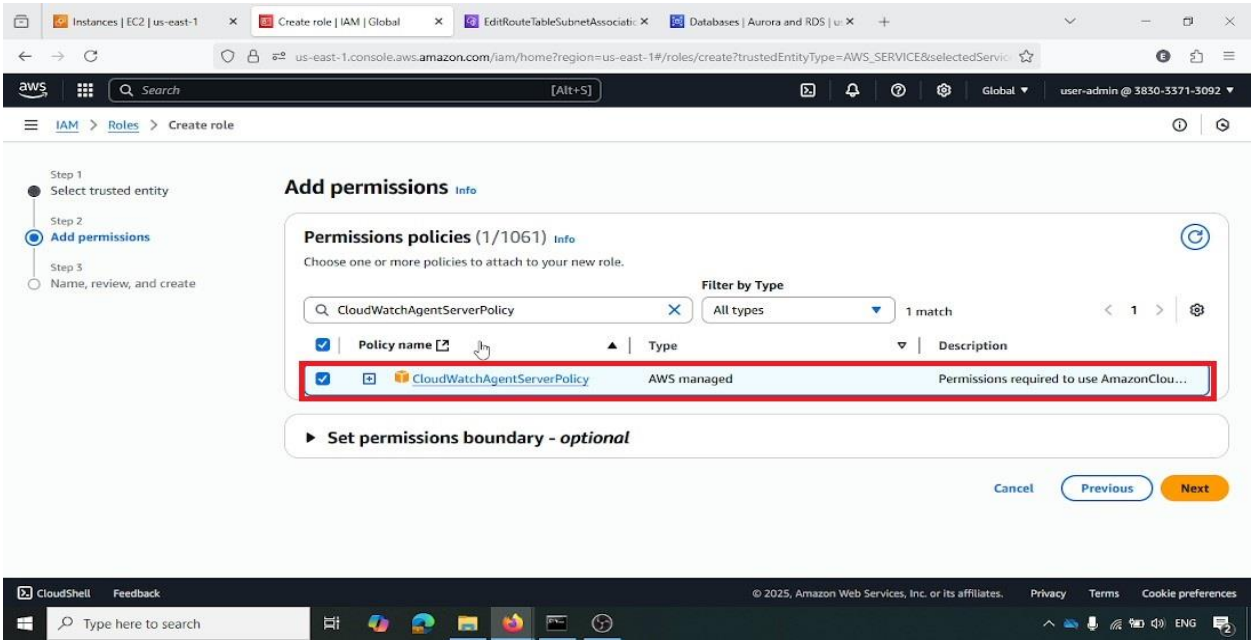
Selecting EC2 as Trusted Entity

In the "Create role" wizard, "EC2" is selected as the trusted entity, allowing EC2 instances to assume this role and its associated permissions



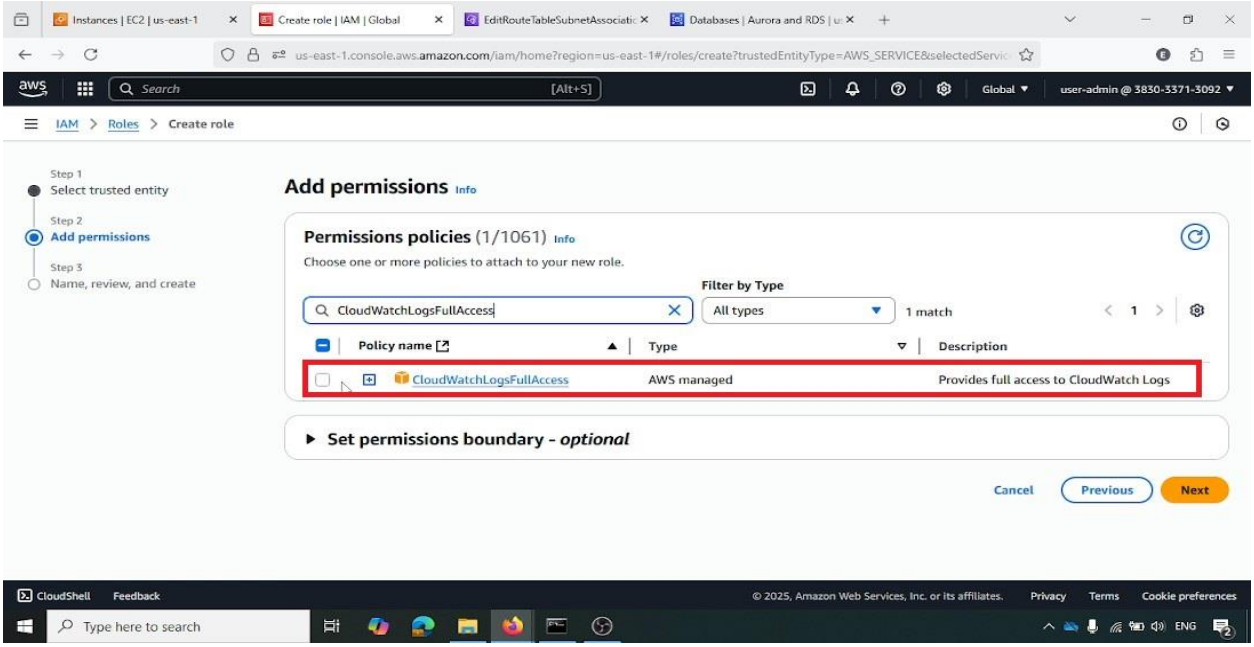
Attaching CloudWatchAgentServerPolicy

The CloudWatchAgentServerPolicy is highlighted and selected. This policy grants the necessary permissions for the CloudWatch agent to collect metrics from the EC2 instance



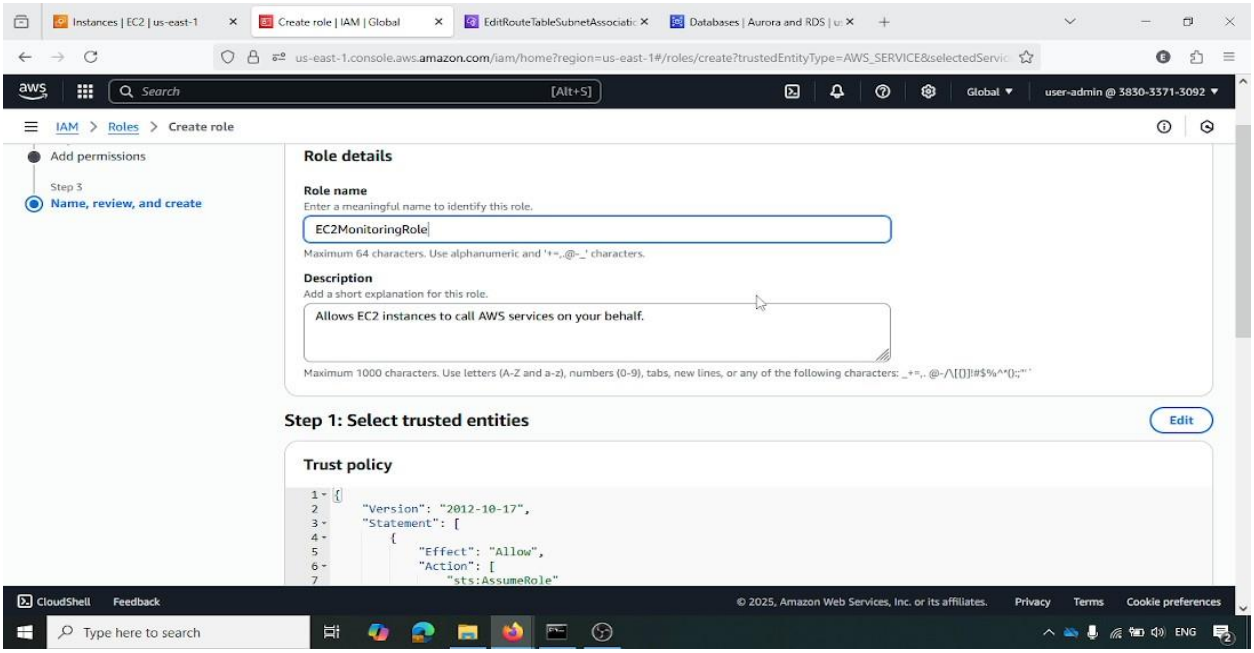
Attaching CloudWatchLogsFullAccess Policy

The CloudWatchLogsFullAccess policy is highlighted and selected. This policy grants permissions for the EC2 instance to send logs to Amazon CloudWatch



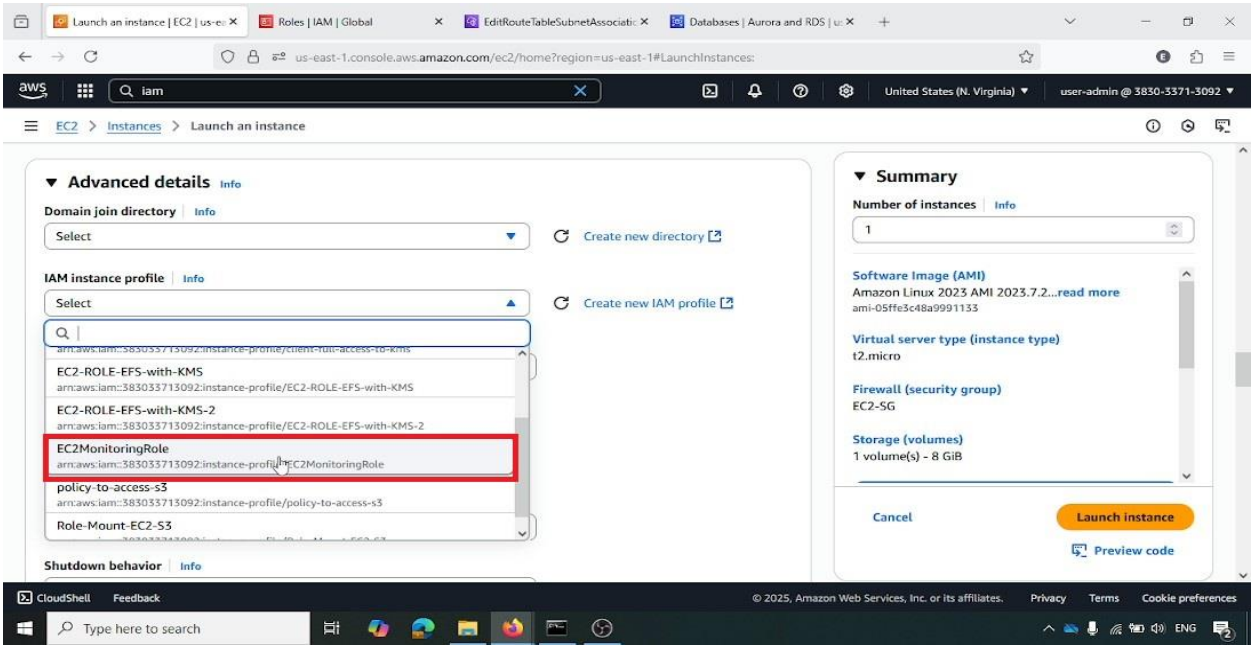
Naming IAM Role

The IAM role is named EC2MonitoringRole. This role will be assigned to EC2 instances to allow them to interact with CloudWatch for monitoring



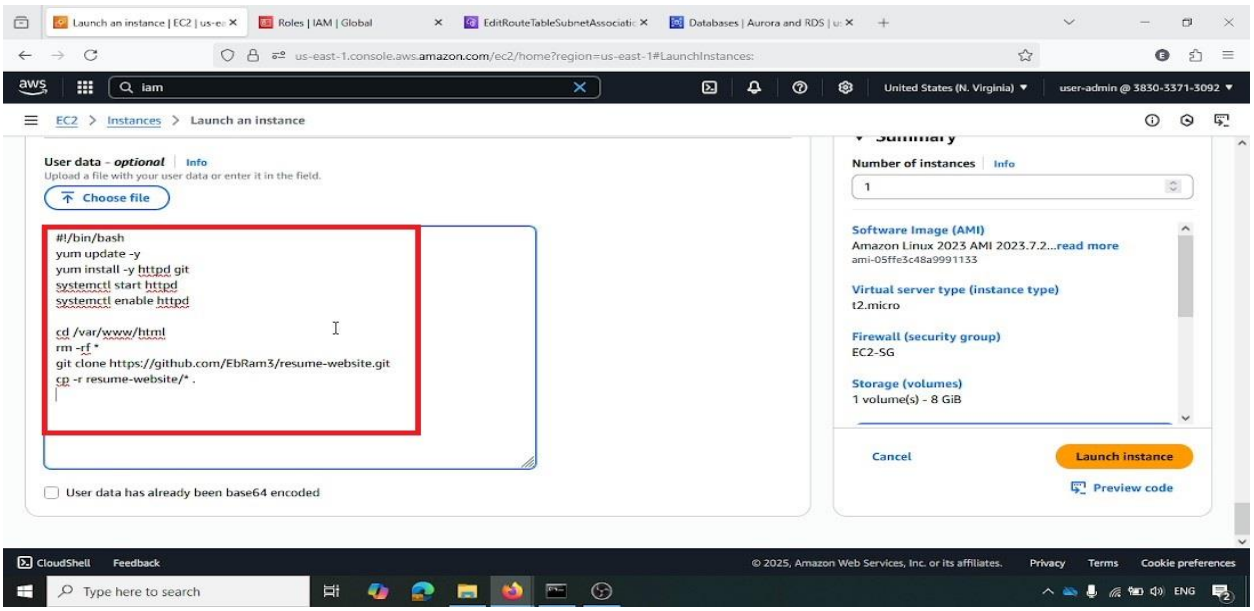
Selecting IAM Instance Profile

In the "Launch an instance" wizard, the EC2MonitoringRole is selected as the IAM instance profile. This attaches the previously created IAM role to the EC2 instance



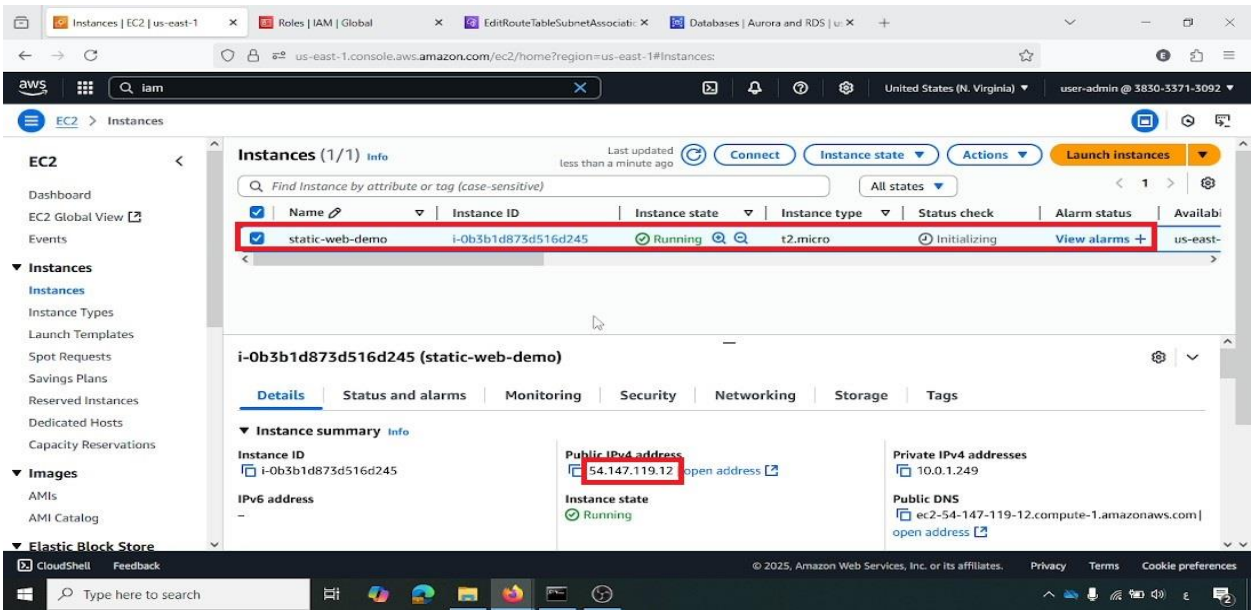
User Data Script for Instance Launch

A bash script is provided as user data. This script automates the installation of Apache HTTP Server, enables it, and deploys the web application by cloning a GitHub repository into the web root (/var/www/html). This ensures the web server is ready upon instance launch



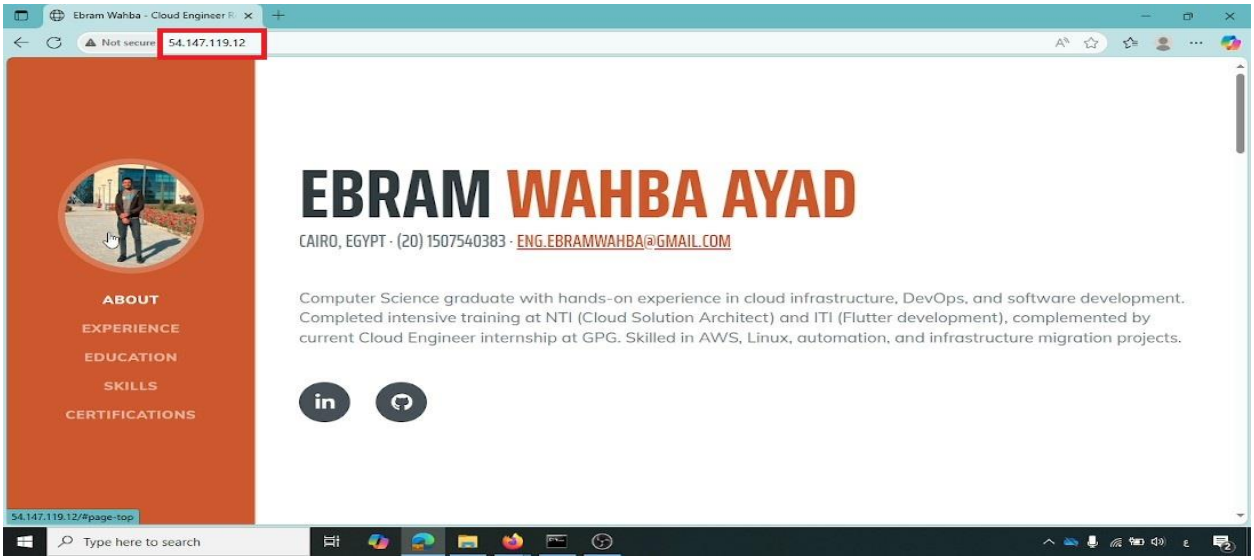
EC2 Instance Public IP Address

This screenshot shows the public IPv4 address (54.147.119.12) of the static-web-demo instance, which is used for direct access to the web application



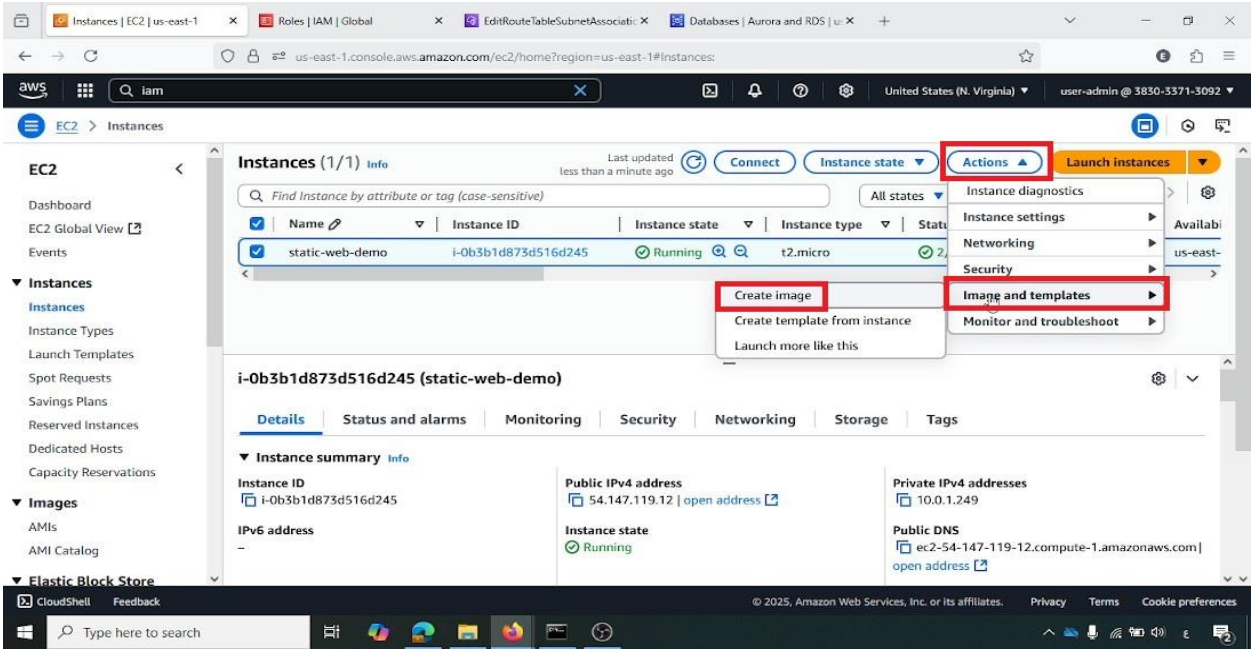
Accessing Web Application via Public IP

A web browser displays the personal portfolio website, confirming that the web application is successfully deployed and accessible via the EC2 instance's public IP address (54.147.119.12)



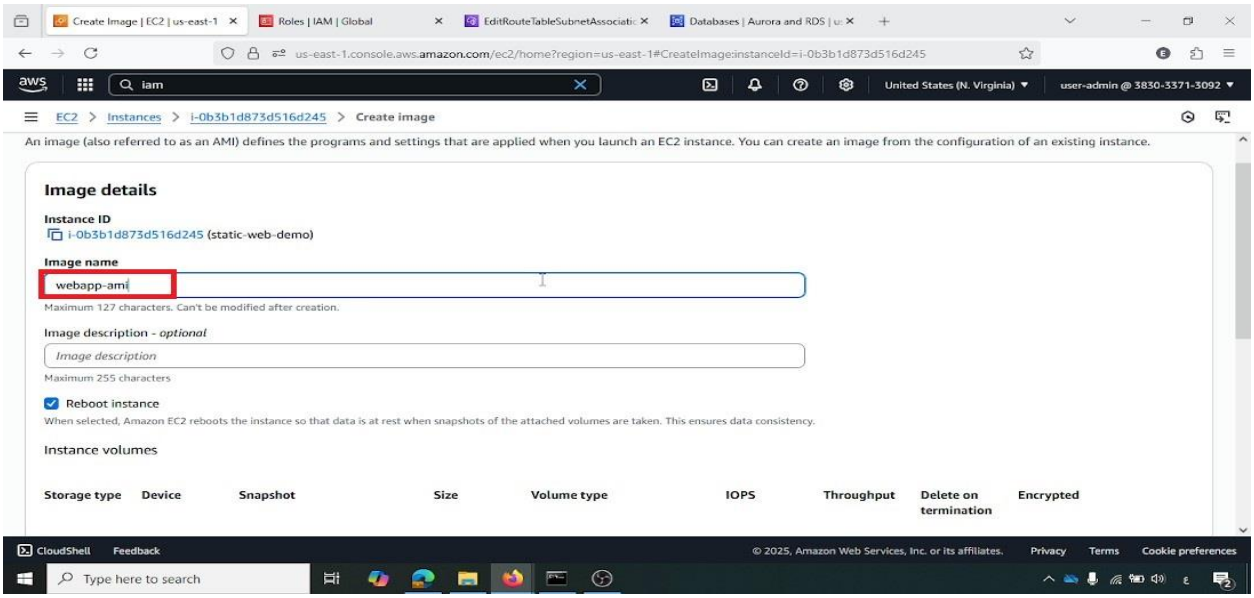
Creating AMI from Instance

From the EC2 instance details, the "Actions" dropdown, then "Image and templates," and finally "Create image" are highlighted. This sequence initiates the creation of a custom Amazon Machine Image (AMI) from the configured instance



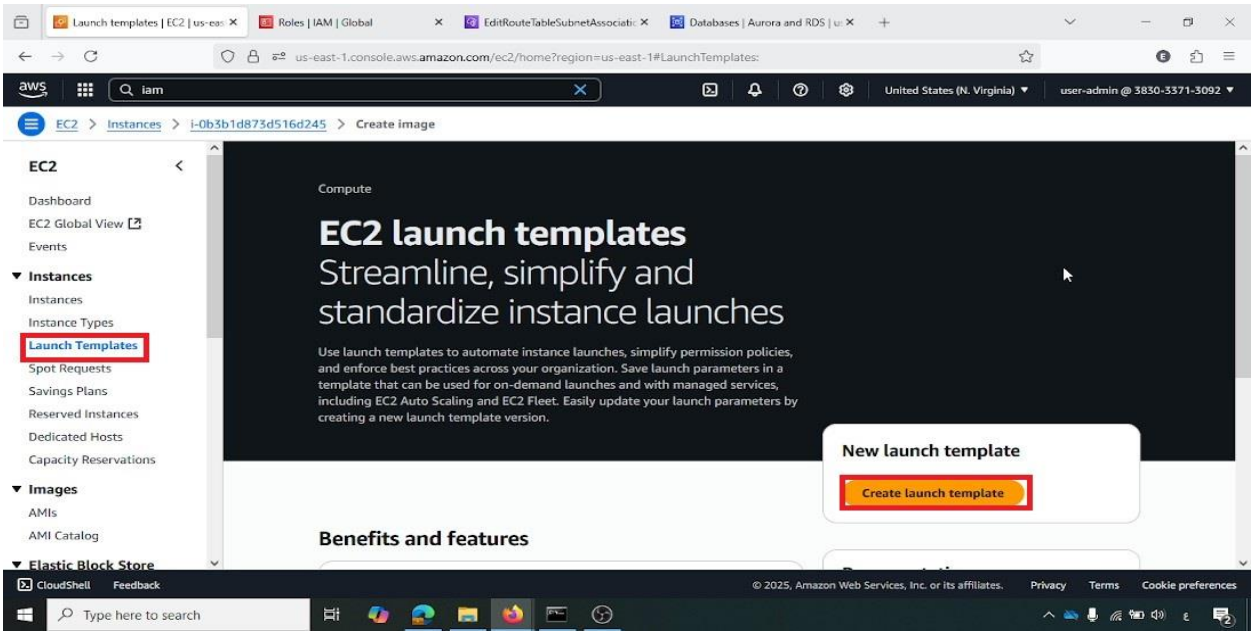
Naming AMI

The AMI is named webapp-ami. This custom AMI will serve as the standardized image for all future instances launched by the Auto Scaling group



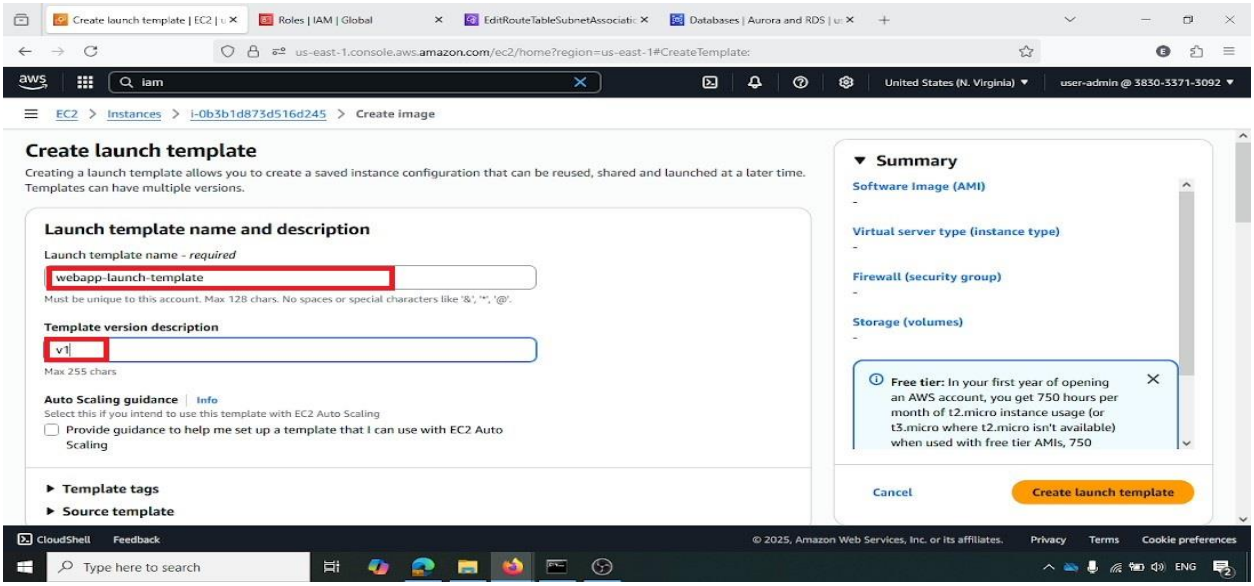
Initiating Launch Template Creation

From the AWS EC2 console, under "Launch Templates," the "Create launch template" button is highlighted, starting the process of defining instance configurations for Auto Scaling



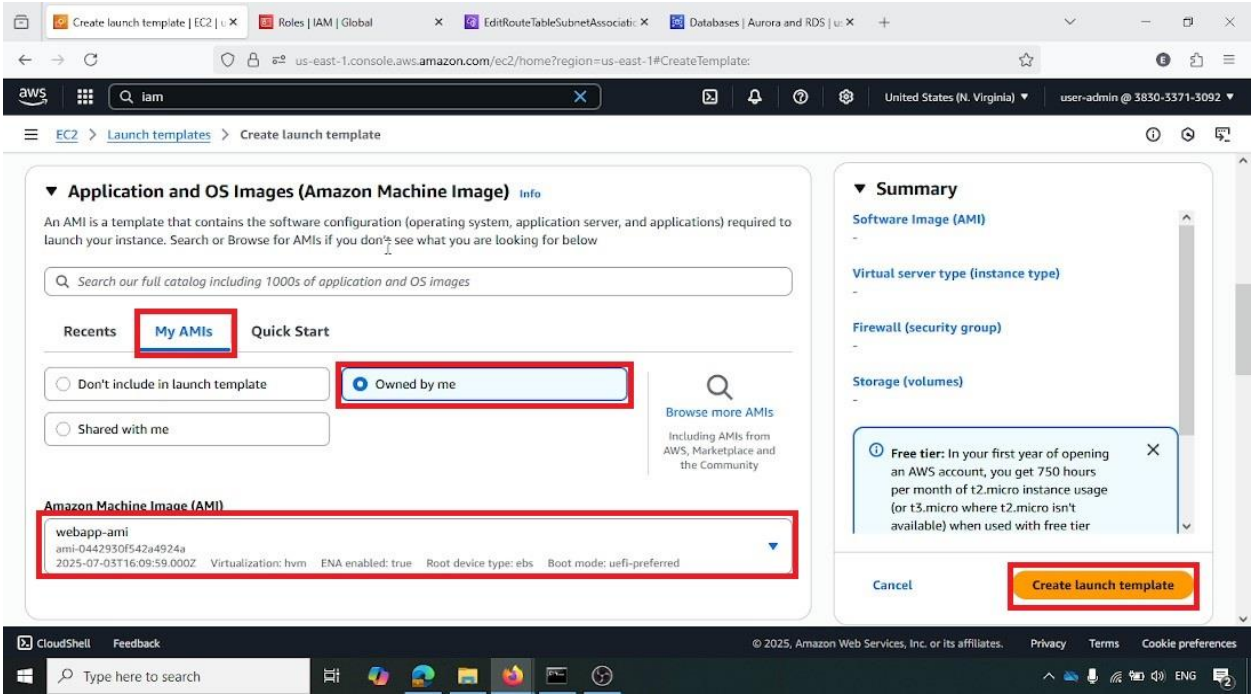
Naming Launch Template and Version

The launch template is named webapp-launch-template with a "Template version description" of v1. This provides a clear identifier for the template



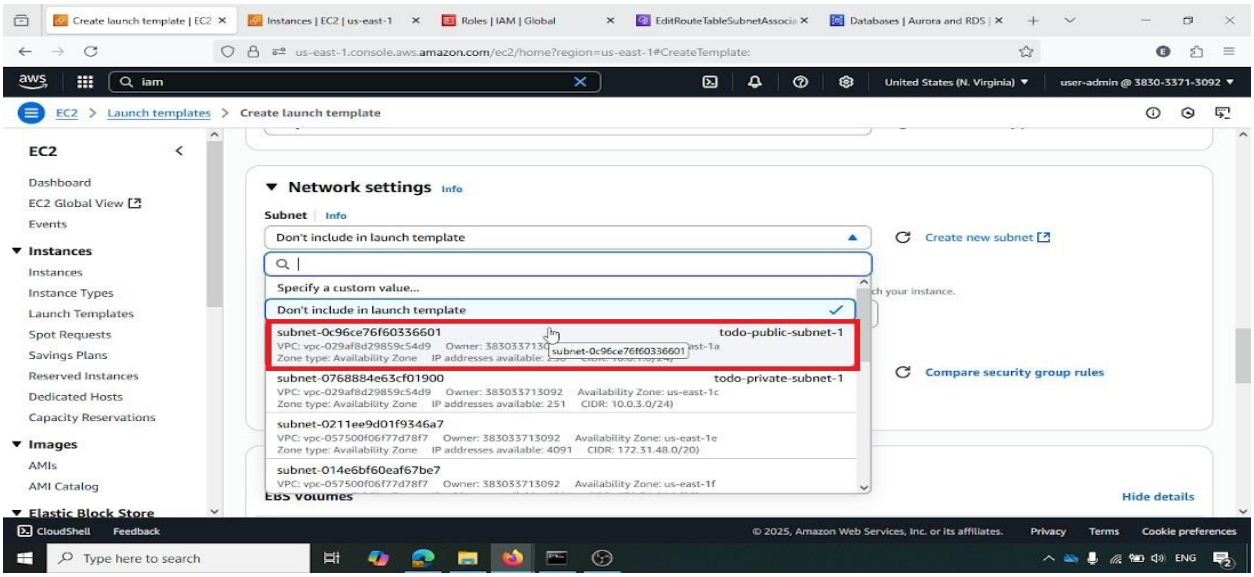
Selecting AMI and Creating Launch Template

The webapp-ami is selected as the Amazon Machine Image for the launch template. The "Create launch template" button is highlighted, finalizing the creation of the template



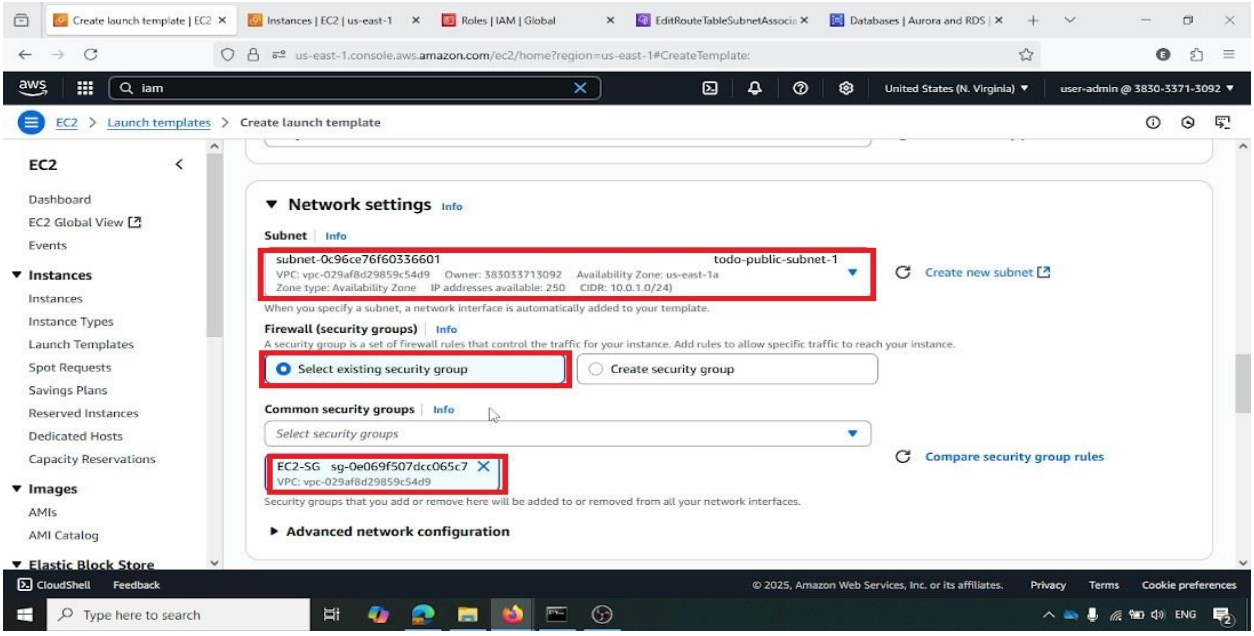
Selecting Subnet for Launch Template

The todo-public-subnet-1 (subnet-0c96ce76f60336601) is highlighted as the selected subnet for the launch template, specifying where instances launched from this template will reside



Configuring Network Settings for Launch Template

The network settings for the launch template are configured, selecting todo-public-subnet-1 and the EC2-SG security group. This ensures that instances launched with this template have the correct network configuration

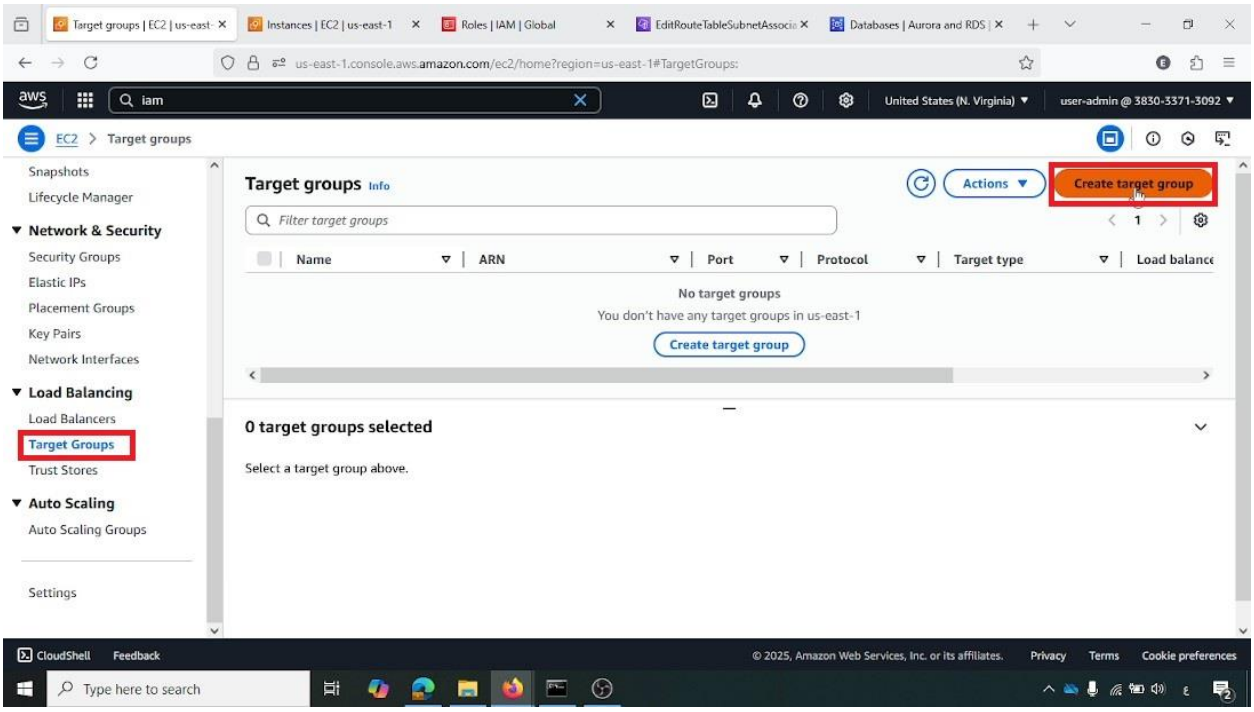


Step 3: Load Balancer Configuration

This section details the setup of the Application Load Balancer (ALB) and its associated target group, which are crucial for distributing incoming traffic across multiple instances and ensuring high availability.

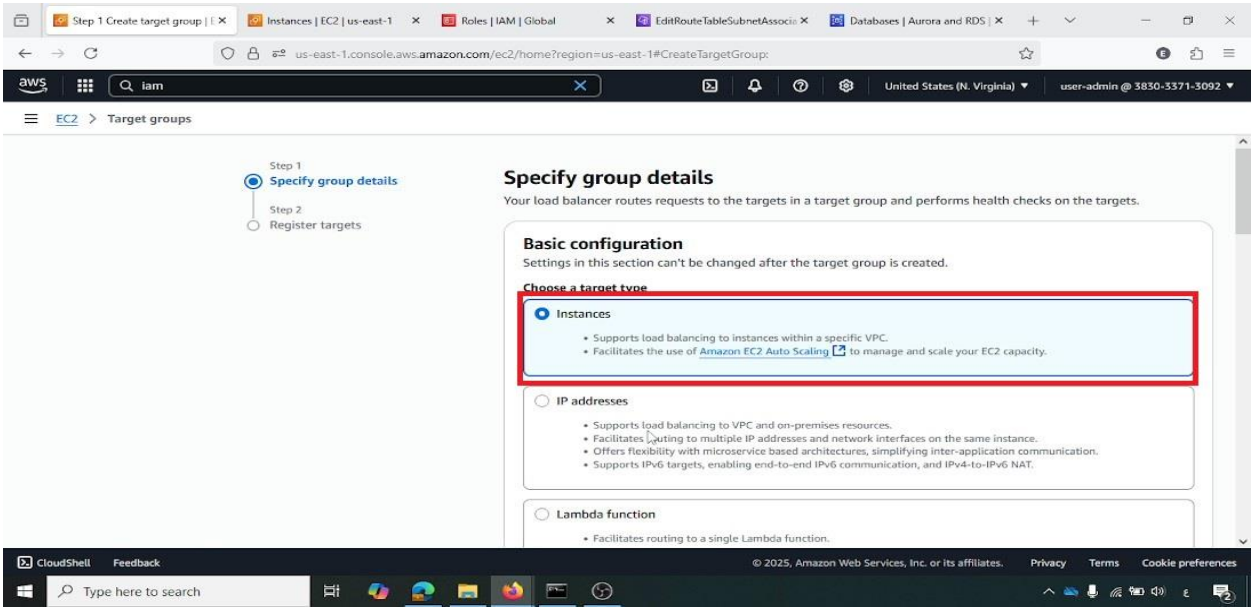
Initiating Target Group Creation

From the AWS EC2 console, under "Target Groups," the "Create target group" button is highlighted, beginning the process of defining a group of targets for the load balancer



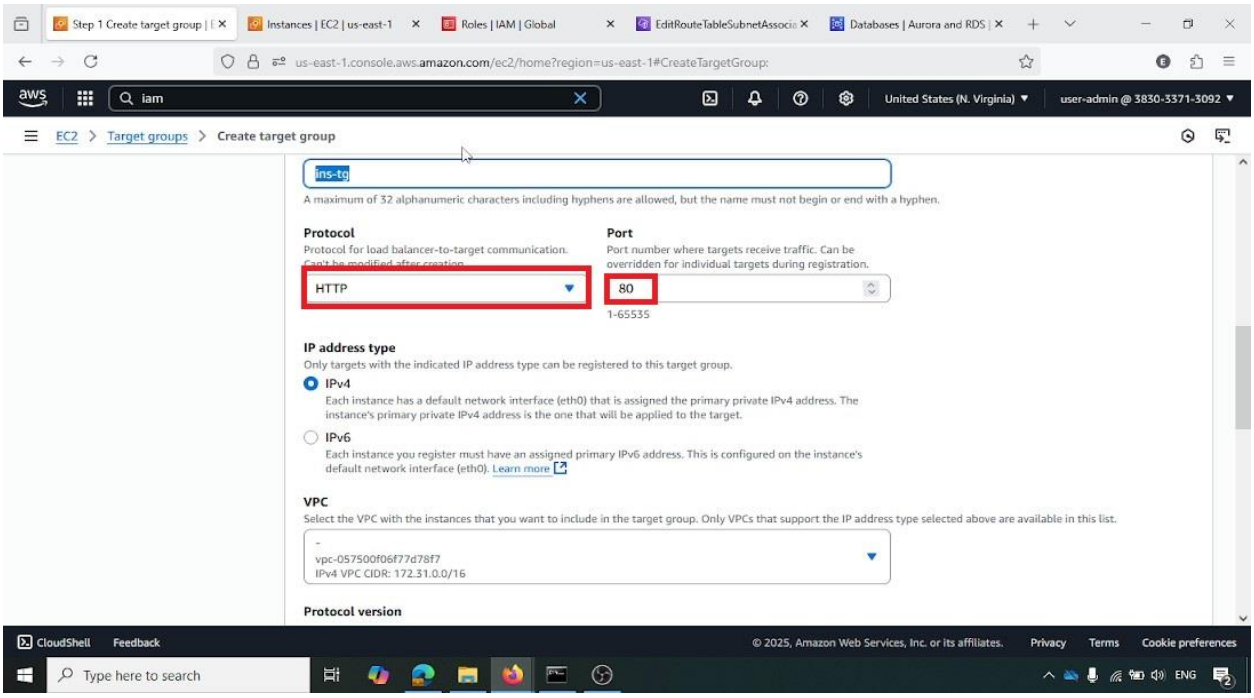
Choosing Target Type for Target Group

In the "Create target group" wizard, "Instances" is selected as the target type. This specifies that the target group will register EC2 instances as its targets



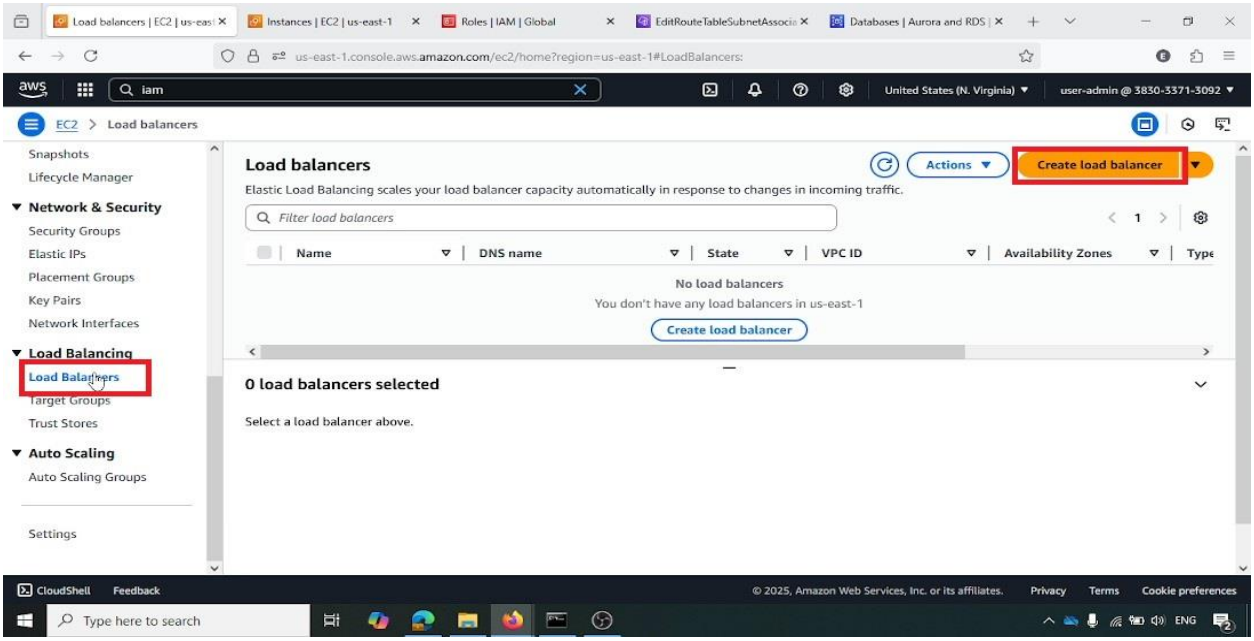
Configuring Target Group Protocol and Port

The target group is configured to use "HTTP" protocol on "Port 80." This defines how the load balancer will communicate with the instances in this group



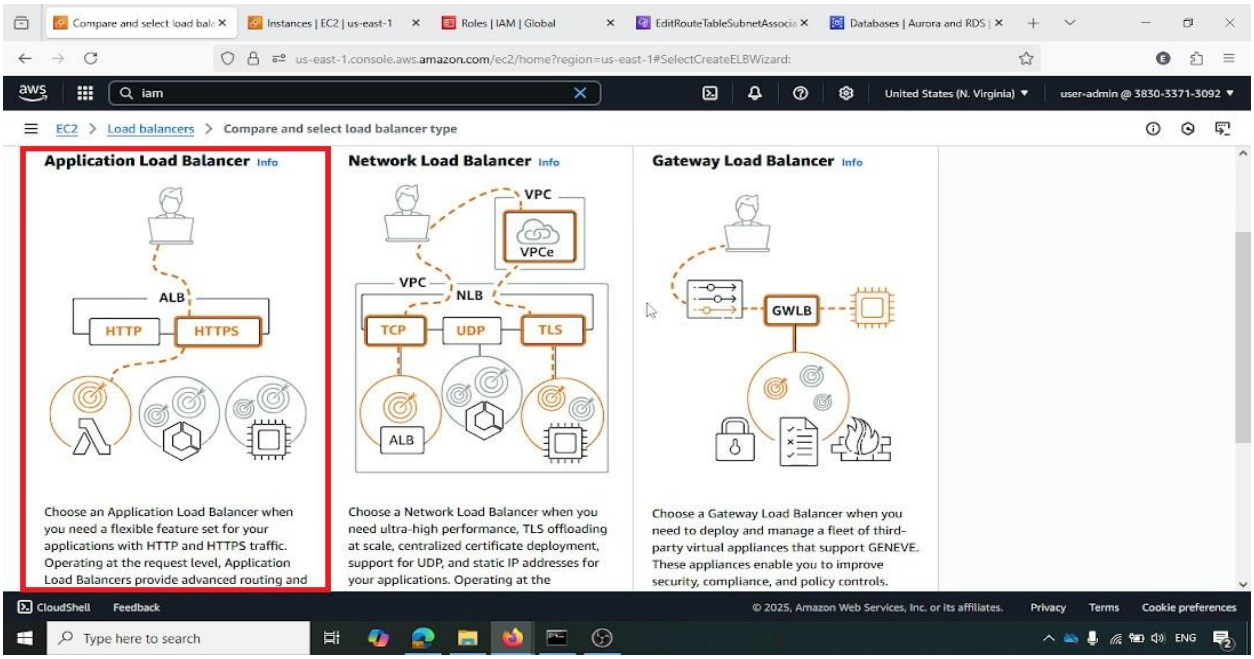
Initiating Load Balancer Creation

From the AWS EC2 console, under "Load Balancers," the "Create load balancer" button is highlighted, starting the process of setting up a new load balancer



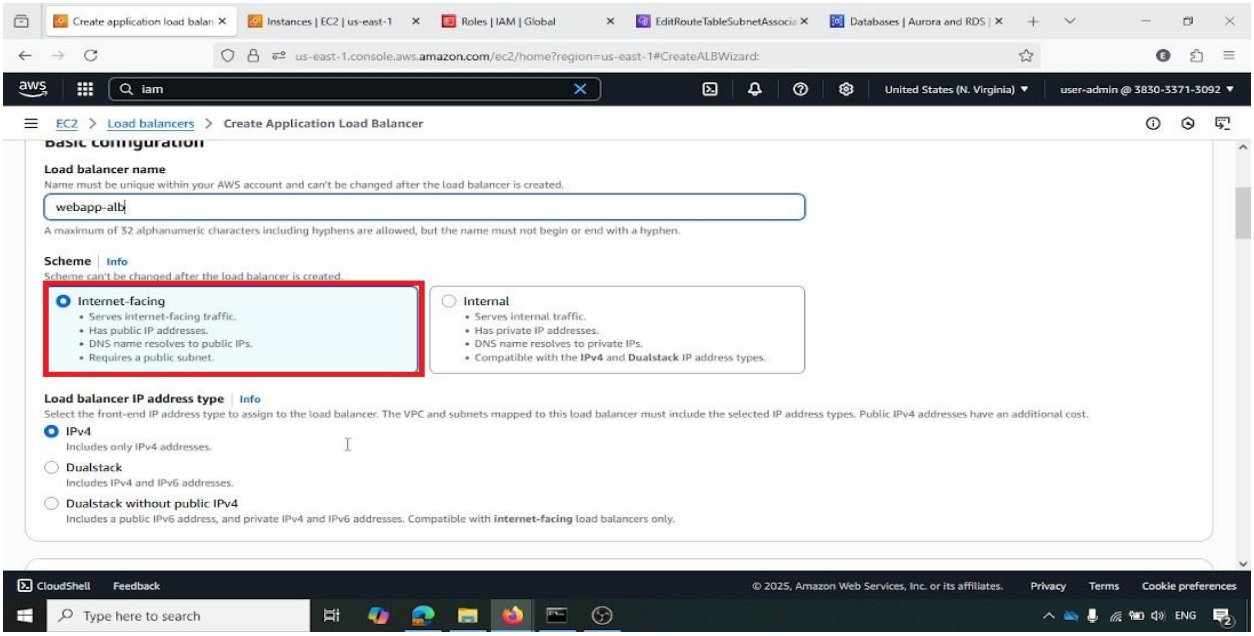
Choosing Application Load Balancer Type

"Application Load Balancer" is highlighted and selected. This type is chosen for its ability to handle HTTP/HTTPS traffic and provide advanced routing features



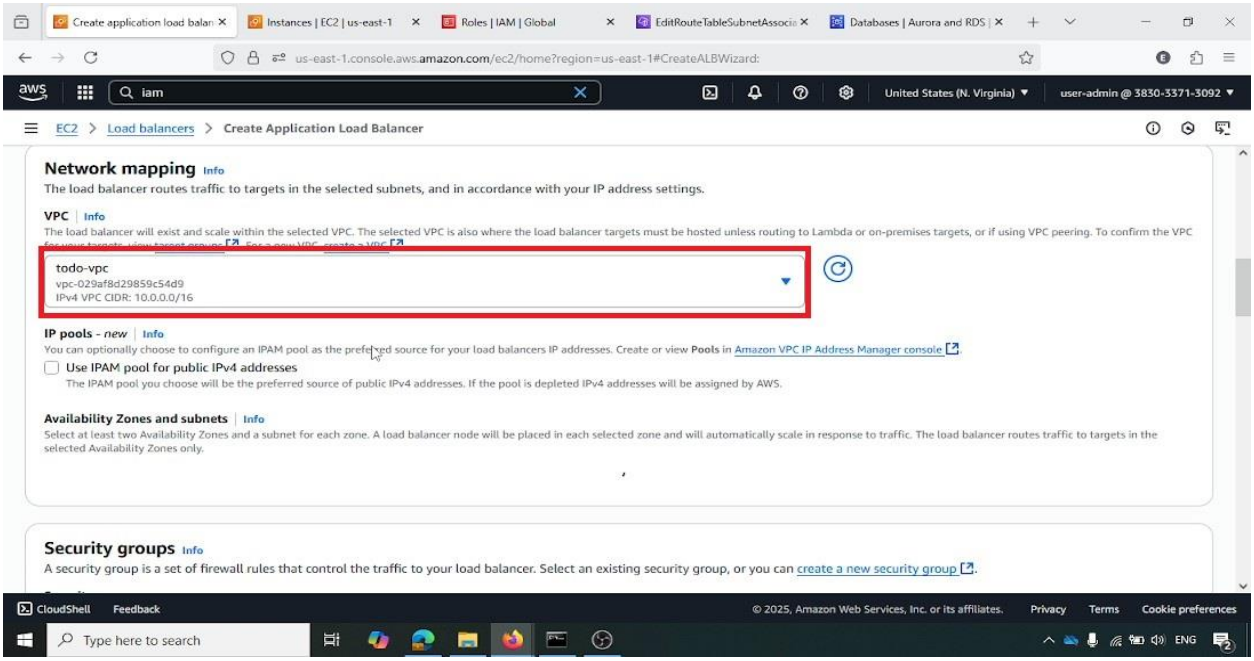
Configuring Basic Load Balancer Settings

The load balancer is named webapp-alb, and its scheme is set to "Internet-facing," making it publicly accessible. This defines the fundamental properties of the ALB



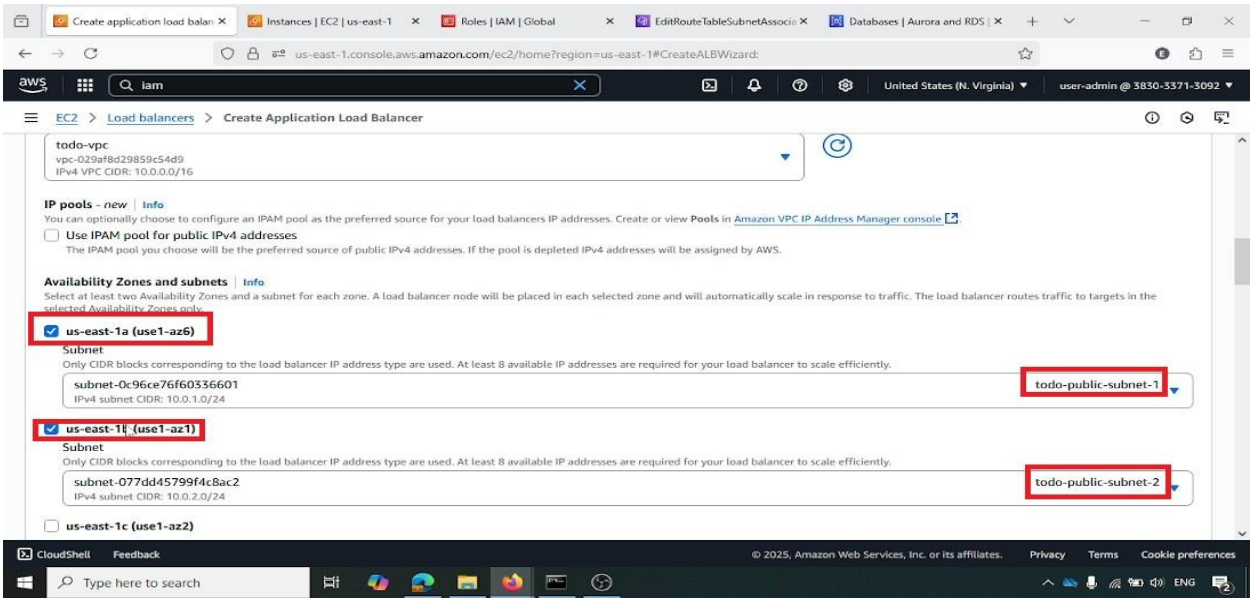
Selecting VPC for Load Balancer

The todo-vpc (vpc-029af8d29859c54d9) is selected as the VPC for the load balancer, ensuring it operates within the defined network environment



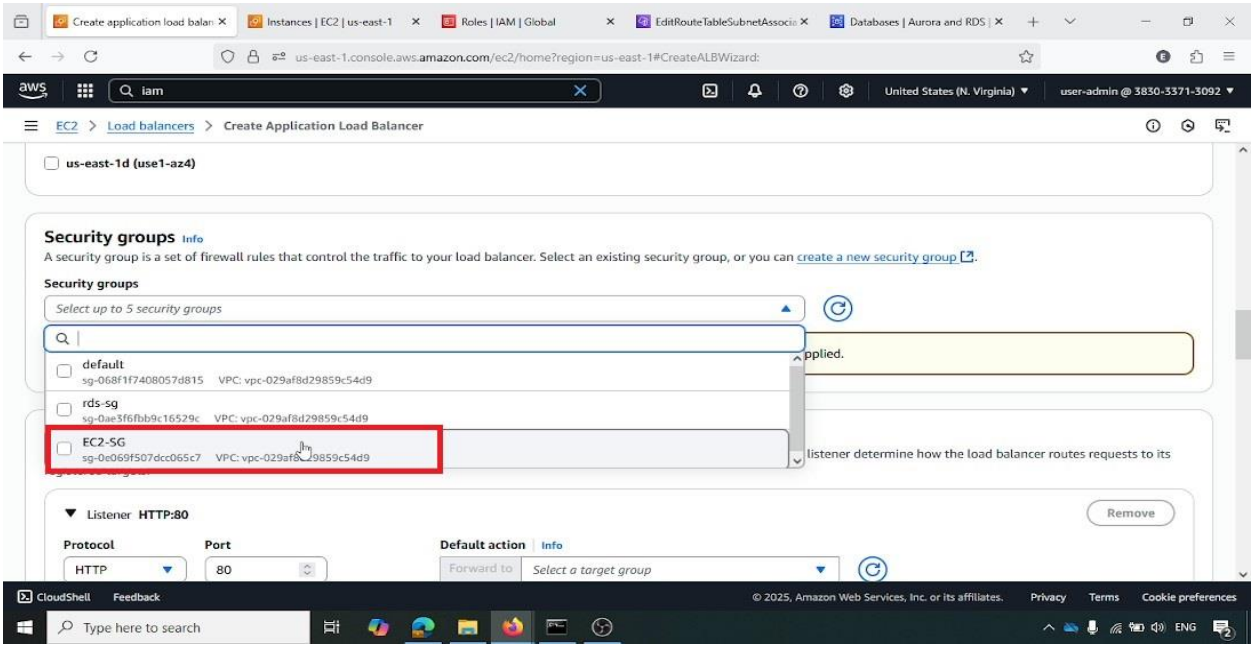
Selecting Availability Zones and Subnets for Load Balancer

Two public subnets, us-east-1a (todo-public-subnet-1) and us-east-1b (todo-public-subnet-2), are selected. This deploys the load balancer across multiple Availability Zones for improved fault tolerance



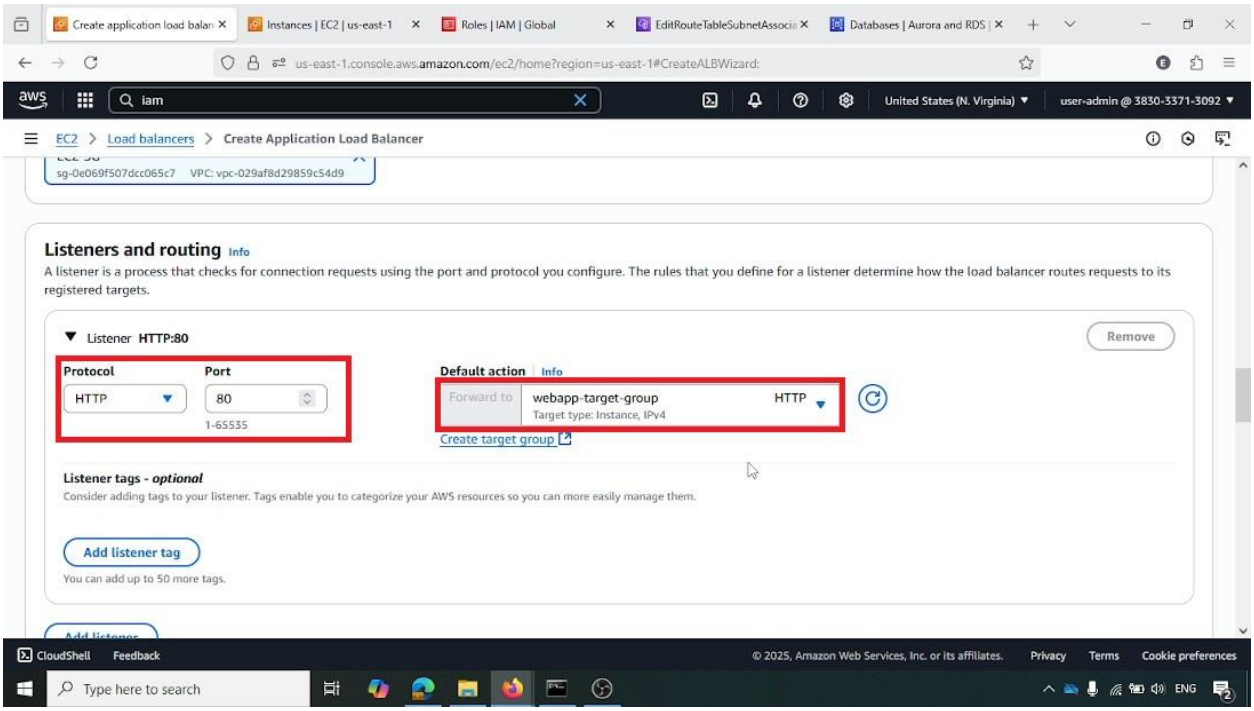
Selecting Security Groups for Load Balancer

The EC2-SG security group (sg-0e069f507c0c065c7) is highlighted and selected for the load balancer. This security group governs the network access to the ALB



Configuring Listener and Routing for Load Balancer

A listener is configured for "HTTP" on "Port 80," with the default action set to "Forward to" the webapp-target-group. This ensures that incoming HTTP requests are directed to the instances within the target group

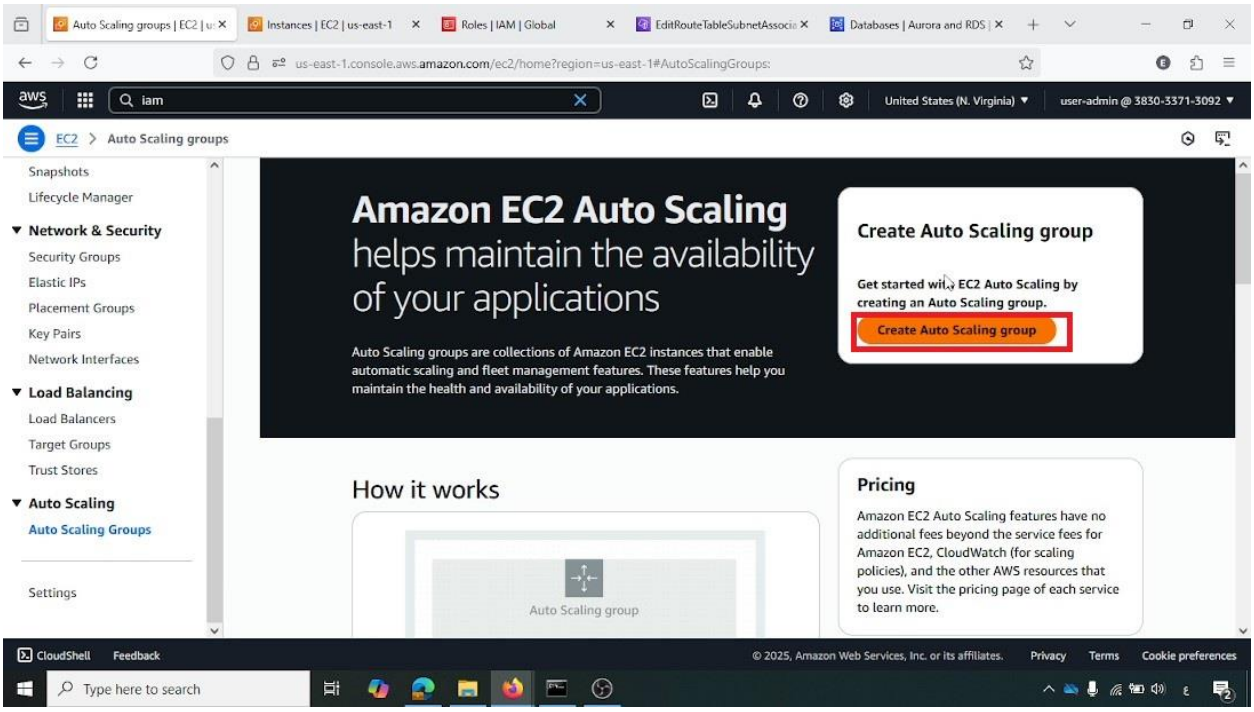


Step 4: Auto Scaling Group Setup

This section outlines the configuration of the Auto Scaling Group, which automatically manages the number of EC2 instances based on demand, ensuring application scalability and high availability.

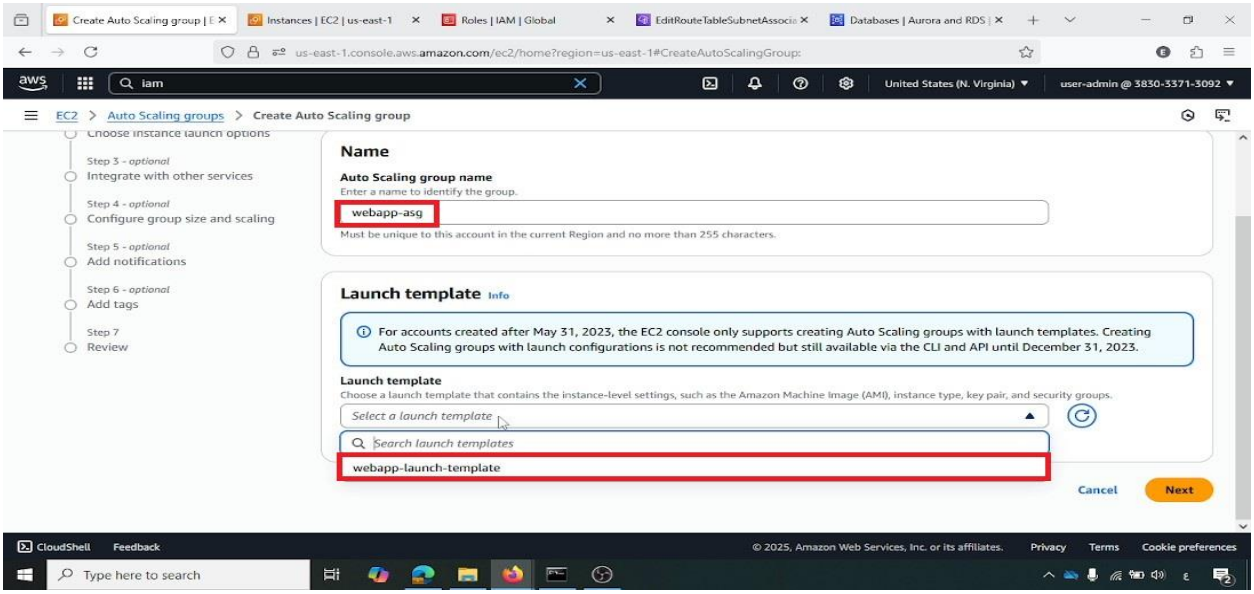
Initiating Auto Scaling Group Creation

From the AWS EC2 console, under "Auto Scaling groups," the "Create Auto Scaling group" button is highlighted, starting the process of creating a new Auto Scaling group



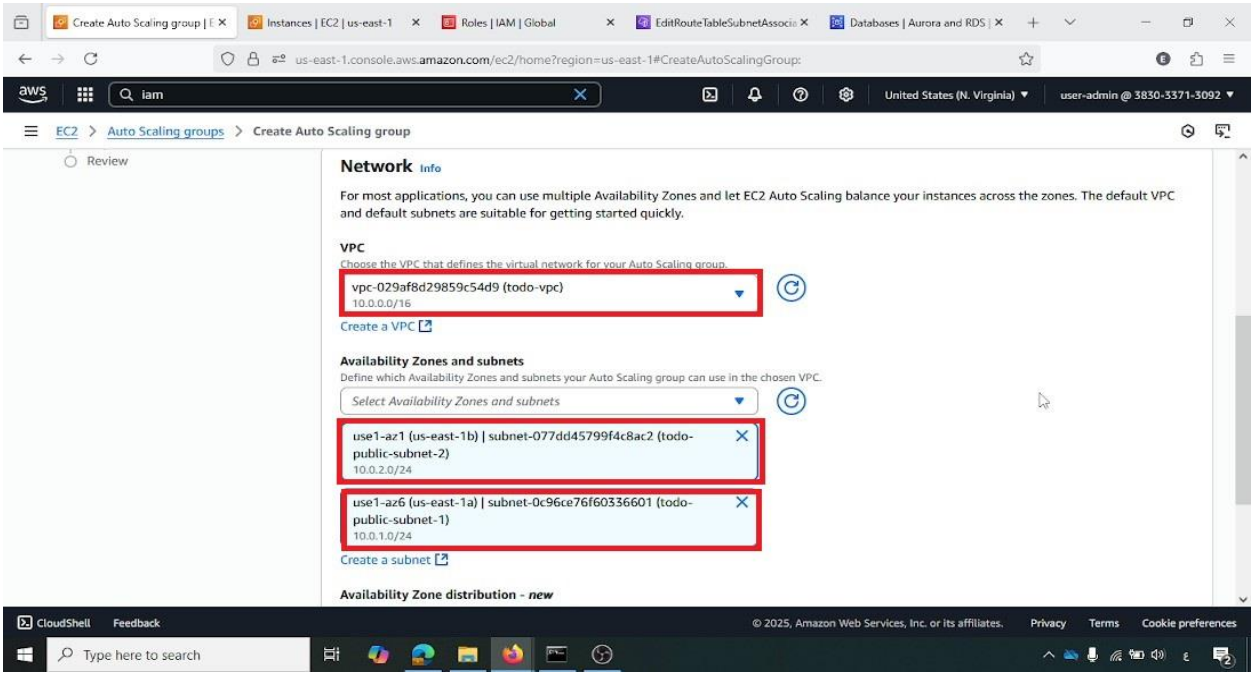
Specifying Auto Scaling Group Name and Launch Template

The Auto Scaling group is named webapp-asg, and the webapp-launch-template is selected. This links the Auto Scaling group to the predefined instance configuration



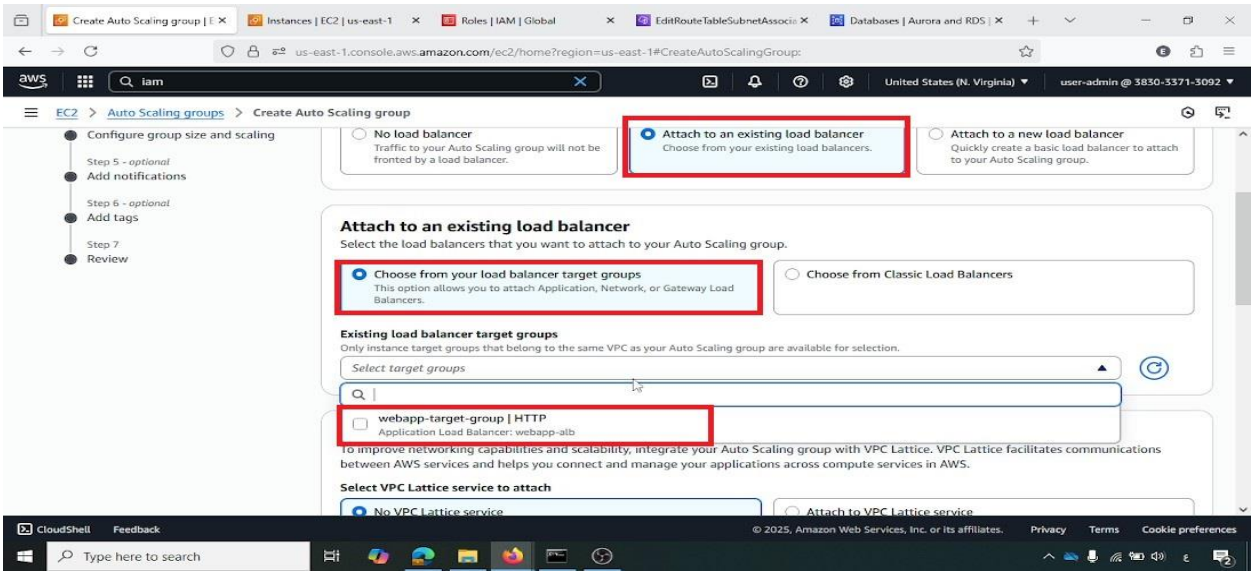
Selecting VPC and Subnets for Auto Scaling Group

The todo-vpc and the two public subnets (use1-az1 and use1-az6) are selected for the Auto Scaling group. This ensures instances are launched in the correct network and across multiple Availability Zones



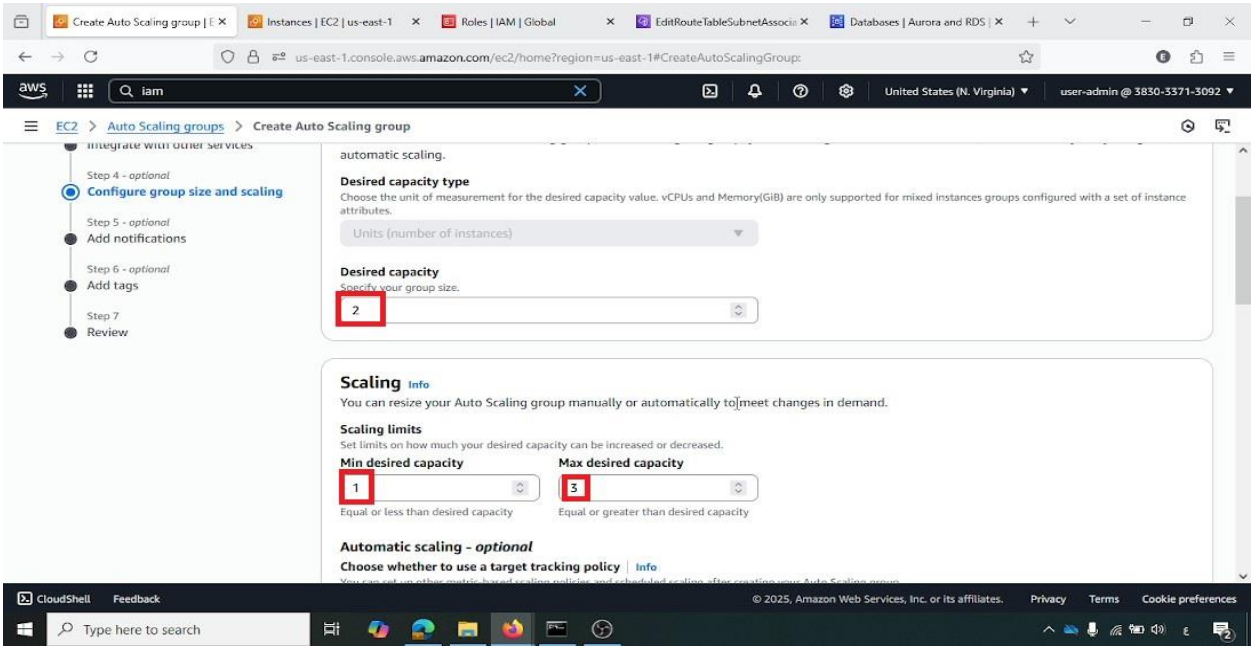
Attaching to Existing Load Balancer

The option "Attach to an existing load balancer" is selected, and the webapp-target-group is chosen. This integrates the Auto Scaling group with the Application Load Balancer, allowing traffic to be distributed to its instances



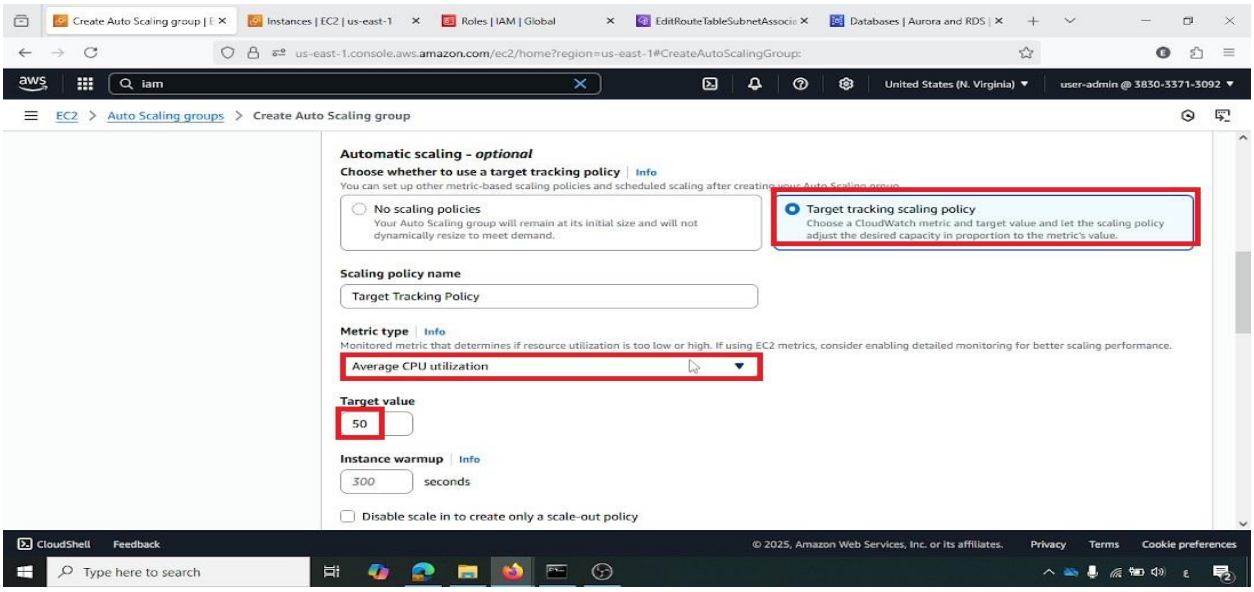
Setting Desired Capacity and Scaling Limits

The "Desired capacity" is set to "2" instances, with a "Min desired capacity" of "1" and a "Max desired capacity" of "3." These settings define the initial and boundary scaling behavior of the group



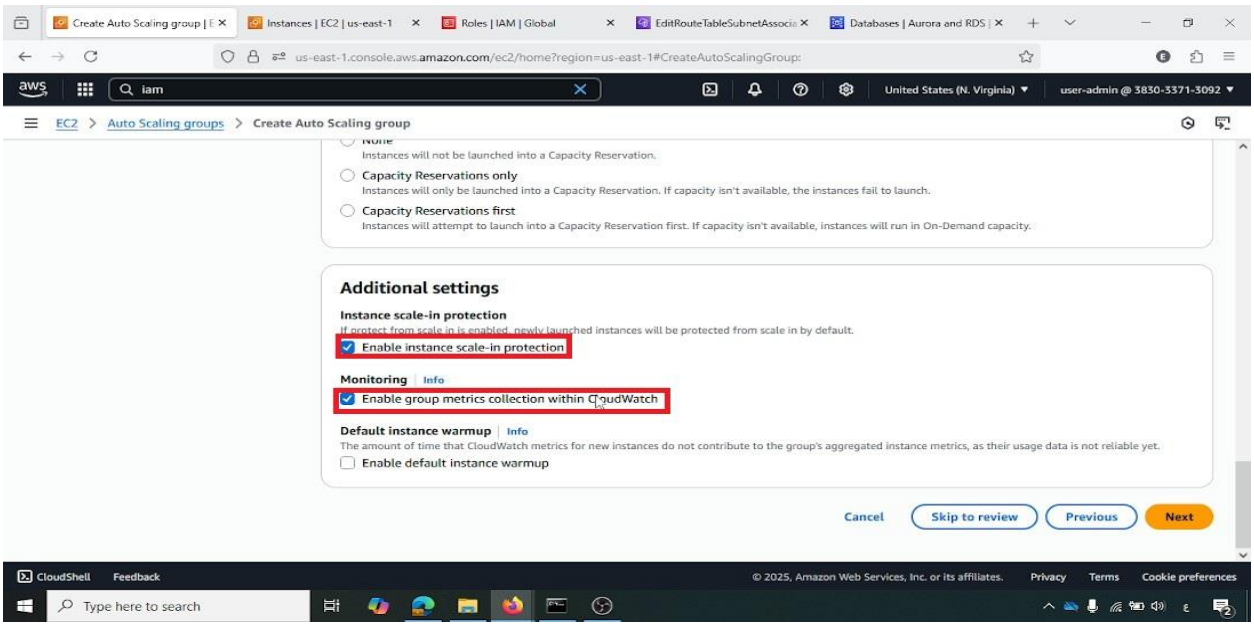
Configuring Target Tracking Scaling Policy

A "Target tracking scaling policy" is configured based on "Average CPU utilization" with a "Target value" of "50." This policy will automatically scale the number of instances to maintain an average CPU utilization around 50%



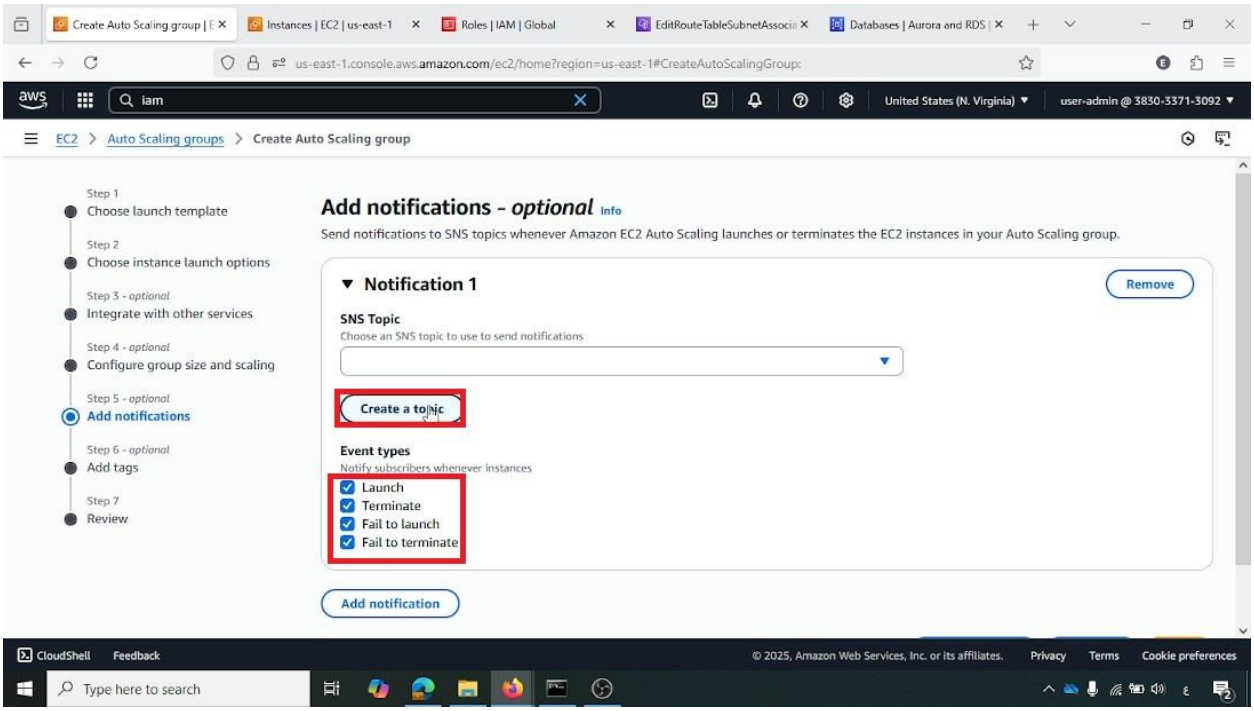
Enabling CloudWatch Group Metrics Collection

The "Enable group metrics collection within CloudWatch" checkbox is selected. This ensures that detailed performance metrics for the Auto Scaling group are sent to CloudWatch for monitoring



Adding Notifications to Auto Scaling Group

The "Add notifications" step is shown, with "Create a topic" highlighted. Event types like "Launch," "Terminate," "Fail to launch," and "Fail to terminate" are selected. This configures the Auto Scaling group to send notifications for these lifecycle events

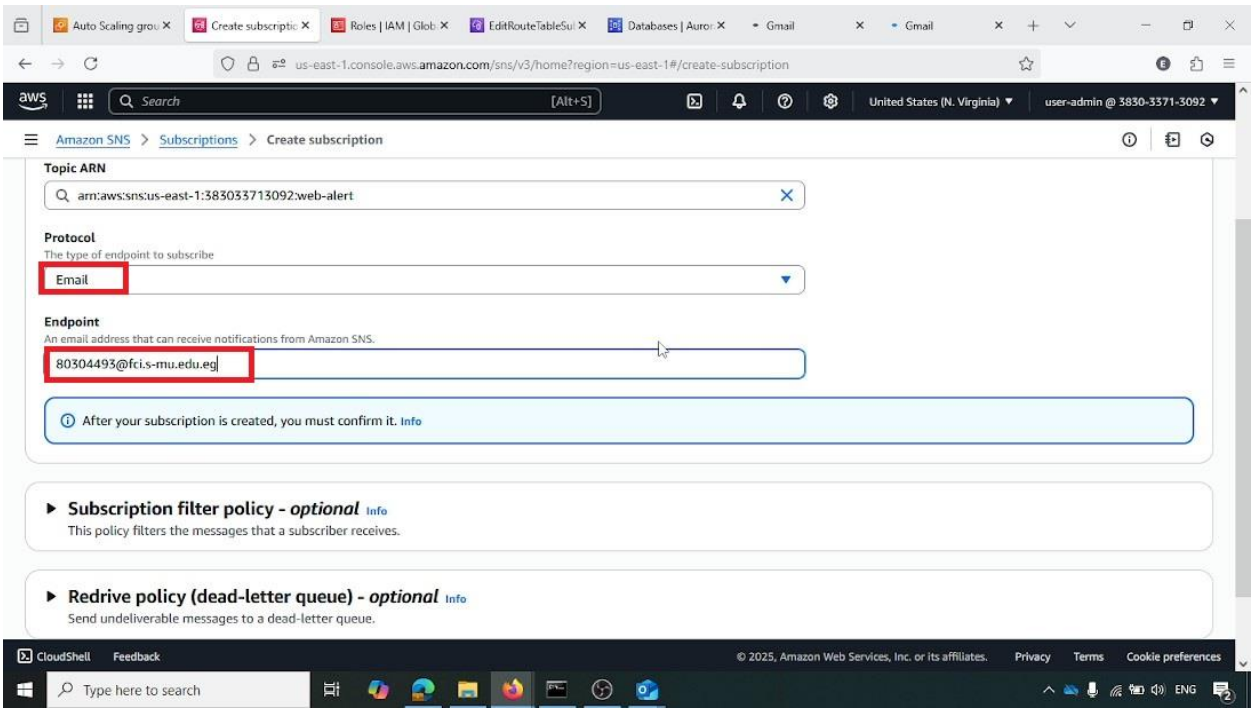


Step 5: Monitoring and Notifications

This phase details the setup of Amazon SNS for email notifications and the configuration of CloudWatch alarms to provide real-time alerts for critical events

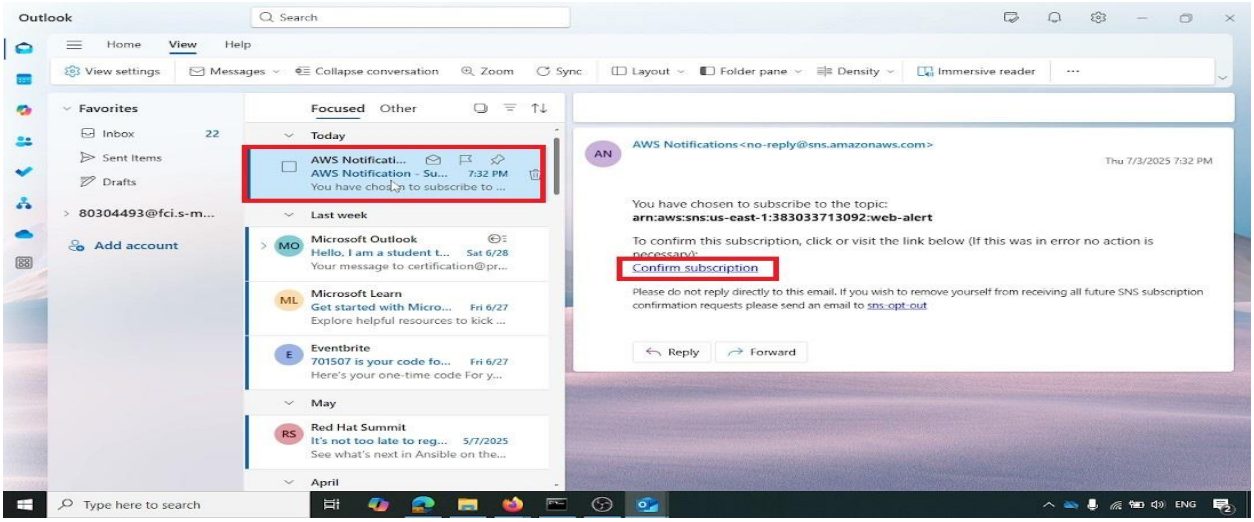
Creating SNS Subscription

In the AWS SNS console, a new subscription is created with "Protocol" set to "Email" and the "Endpoint" specified as 80304493@fci-s.mu-edu.eg. This configures the email address to receive notifications from the SNS topic



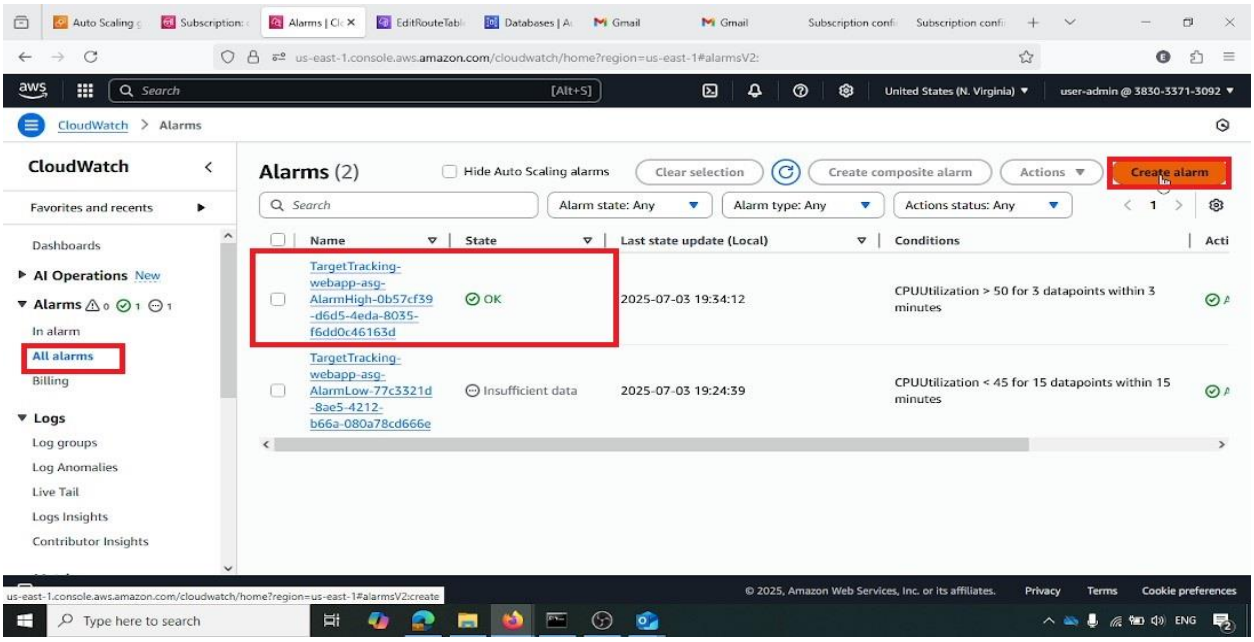
Confirming SNS Subscription

This screenshot shows an email received in Outlook, prompting confirmation of the SNS subscription. The "Confirm subscription" link is highlighted, which is clicked to activate the notification delivery



CloudWatch Alarms Overview

This image provides an overview of the CloudWatch alarms. It shows two alarms configured for the webapp-asg: one for high CPU utilization (status "OK") and another for low CPU utilization (status "Insufficient data"). The "Create alarm" button is also visible, indicating where new alarms can be configured

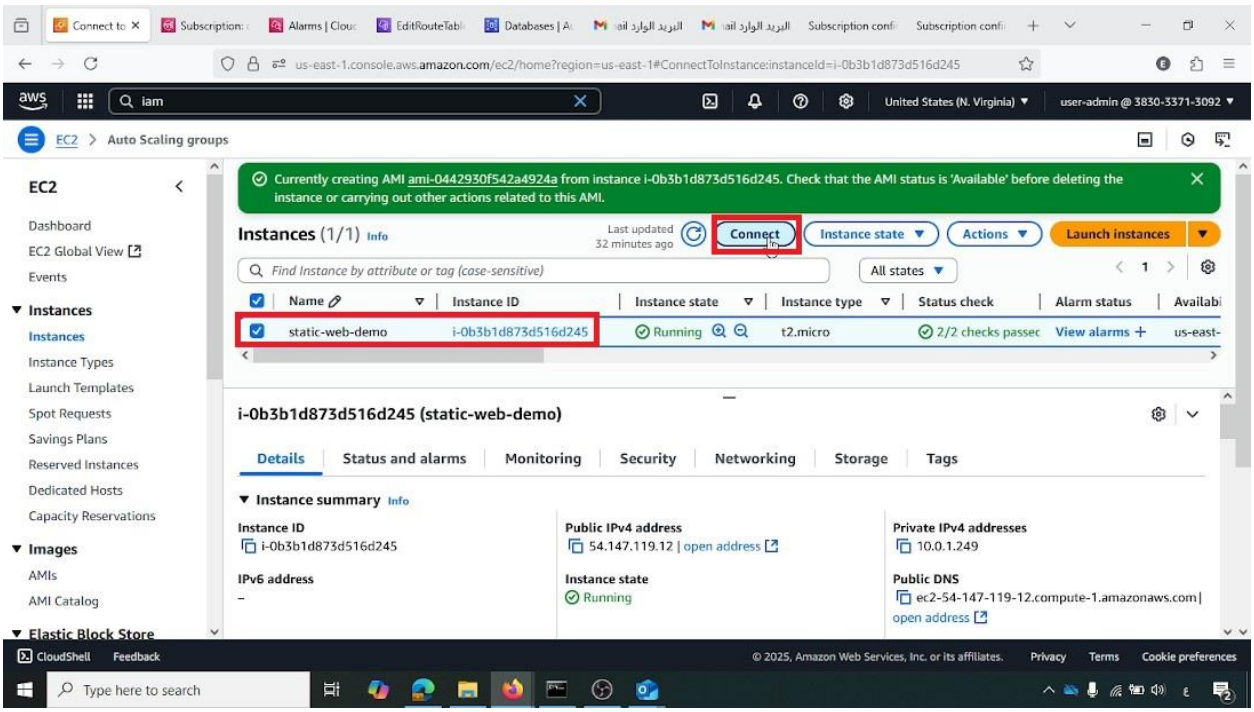


Step 6: Testing and Validation

This final phase involves testing the auto-scaling functionality by simulating load and verifying that the web application is accessible through the load balancer

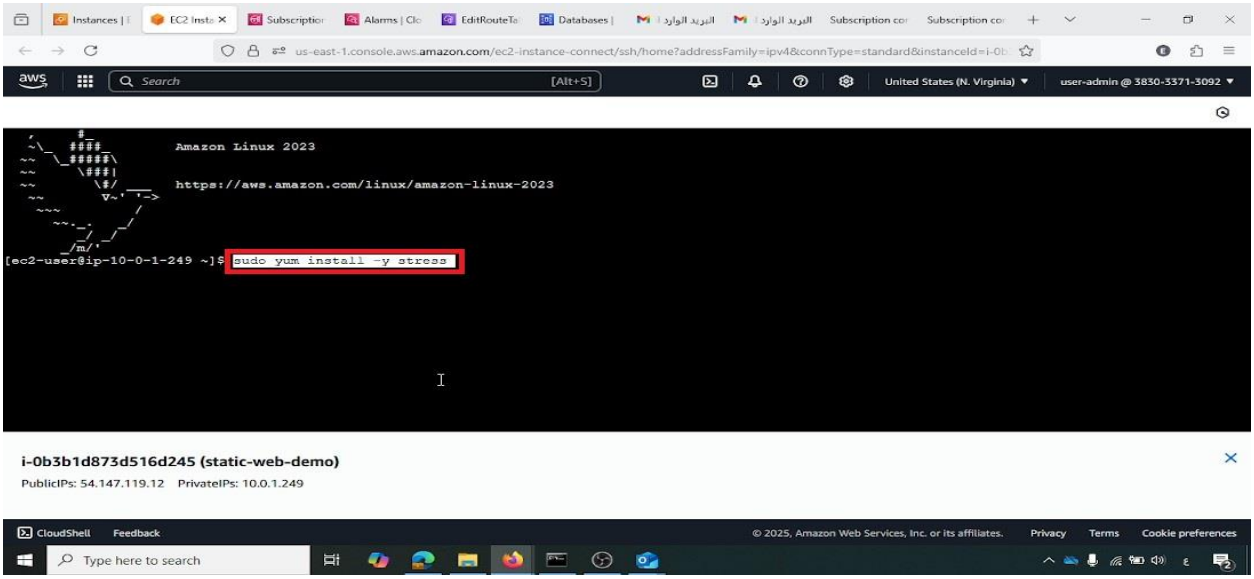
Connecting to EC2 Instance

This screenshot from the AWS EC2 console shows the "Connect" button highlighted for the static-web-demo instance. This action is taken to establish an SSH connection to the instance for testing purposes



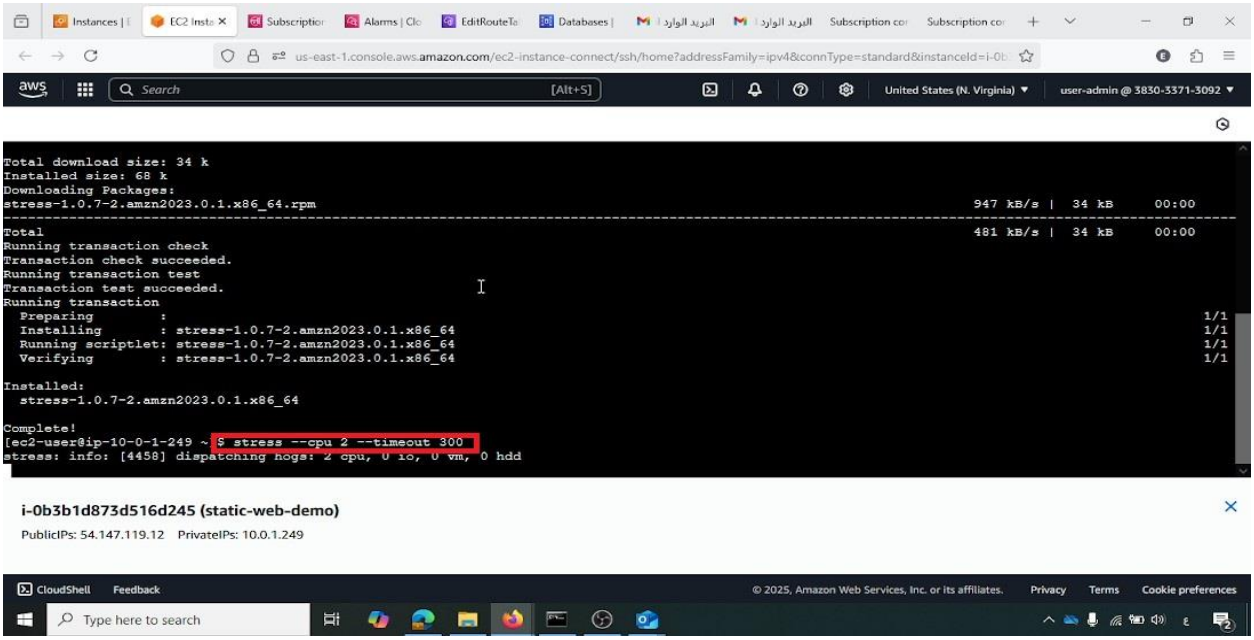
Installing Stress Tool on EC2 Instance

Inside the EC2 instance's CLI, the command sudo yum install -y stress is executed to install the stress tool. This tool is used to generate artificial CPU load



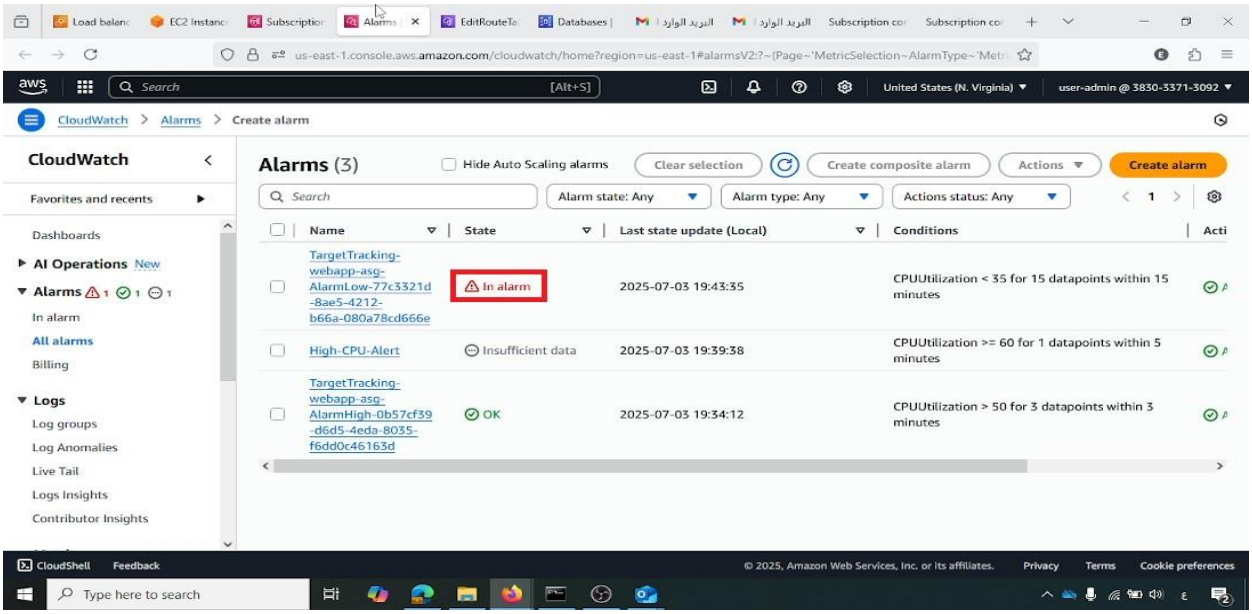
Running Stress Test on EC2 Instance

The command stress --cpu 2 --timeout 300 is executed on the EC2 instance. This command simulates high CPU utilization (2 cores for 300 seconds), which is intended to trigger the CloudWatch high CPU alarm and initiate a scale-out event in the Auto Scaling group



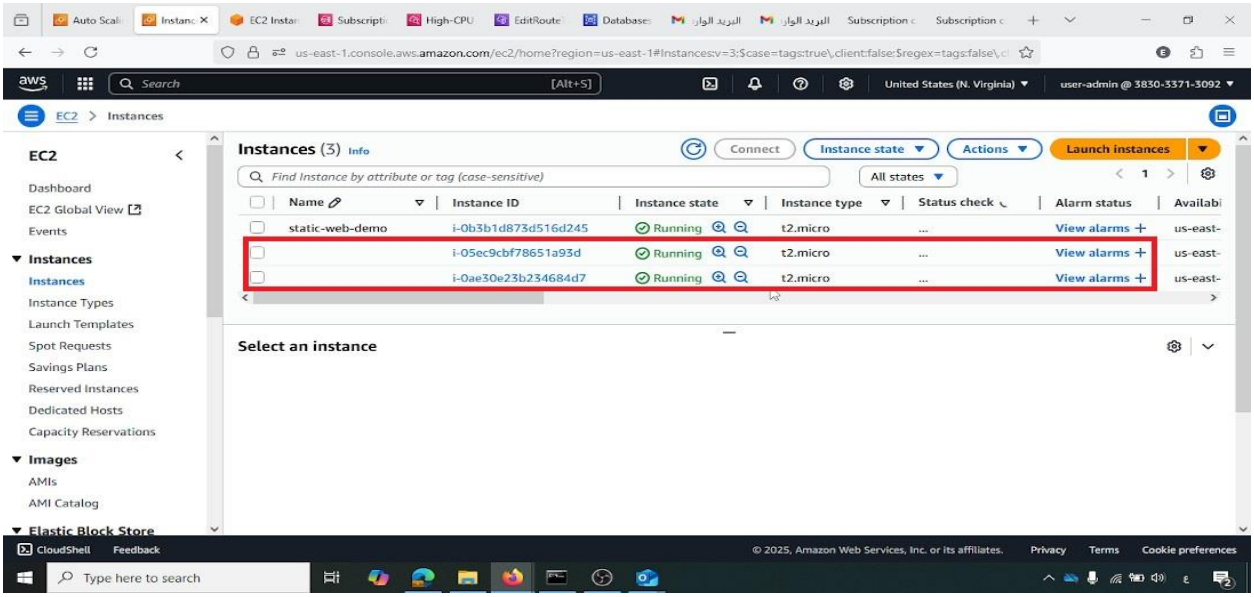
CloudWatch Alarm Status

This screenshot specifically shows the TargetTracking-webapp-asg-AlarmLow alarm in an "In alarm" state, indicating that the CPU utilization has dropped below the set threshold. This demonstrates the active monitoring and alerting capabilities of CloudWatch



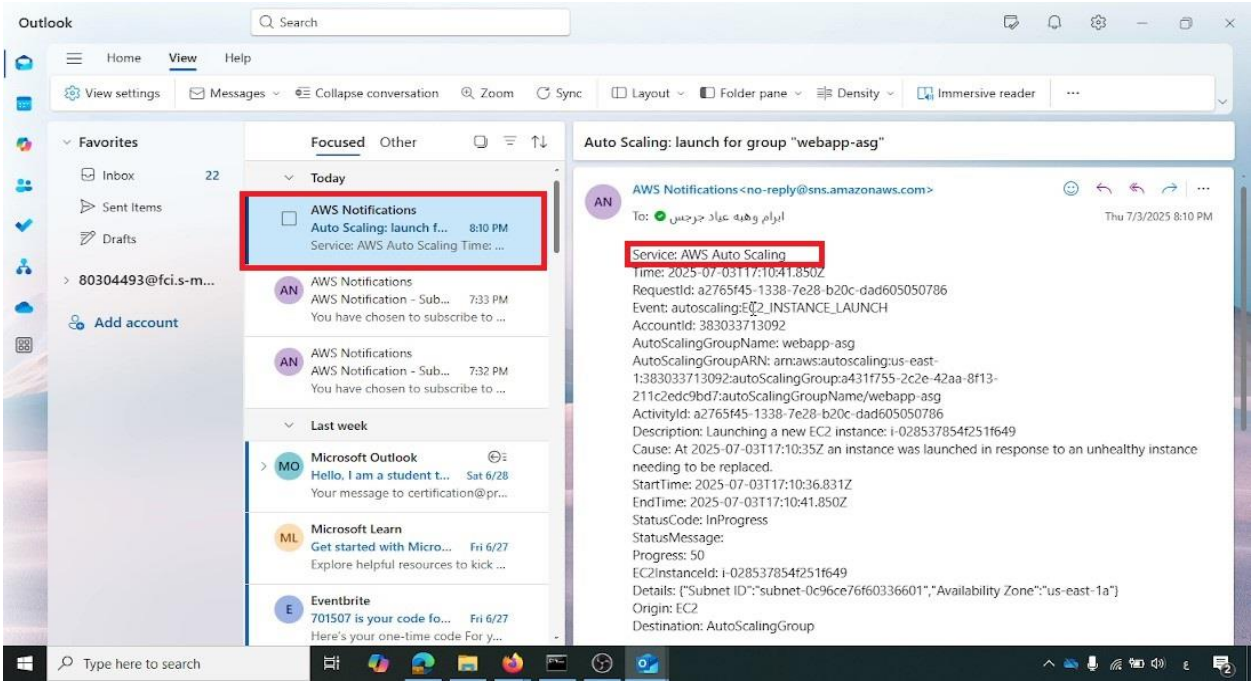
EC2 Instances Running in AWS Console

This image shows three running EC2 instances in the console, with two instances (i-056c3cbf7f8651a93d and i-0ae30e23b234684d7) highlighted. This demonstrates that the Auto Scaling group has successfully scaled out by launching additional instances in response to the simulated load



AWS Auto Scaling Notification Email (Outlook)

This screenshot displays an email notification from AWS Auto Scaling, confirming that an instance was launched. This verifies that the SNS notification system is working correctly for Auto Scaling events



Web Application Accessible via Load Balancer URL

A web browser displays the personal portfolio website, with the URL webapp-alb-232317014.us-east-1.elb.amazonaws.com highlighted. This confirms that the web application is successfully accessible through the Application Load Balancer, indicating a complete and functional deployment