

## Overview of Task: Deploying a Scalable Web Application on AWS

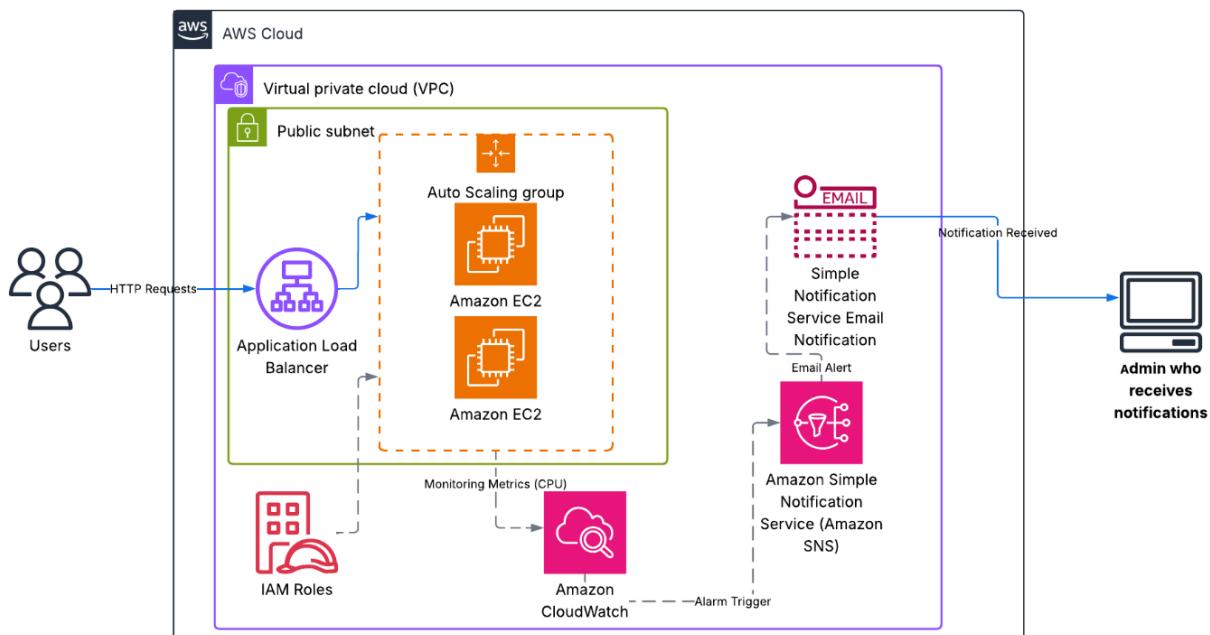
This document provides a comprehensive, step-by-step guide for deploying a highly available, fault-tolerant, and scalable web application on Amazon Web Services (AWS). The primary goal is to establish an infrastructure that automatically adapts to varying traffic loads, efficiently distributes incoming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

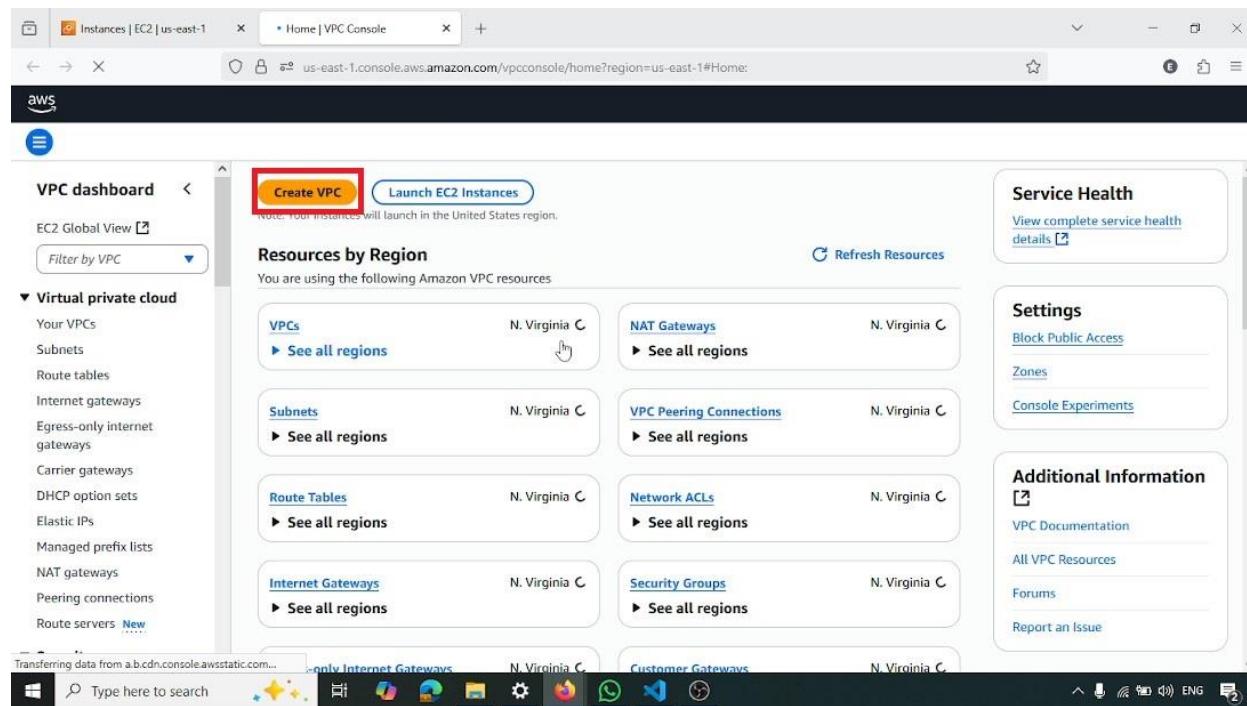


## Step 1: VPC and Network Infrastructure Setup

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

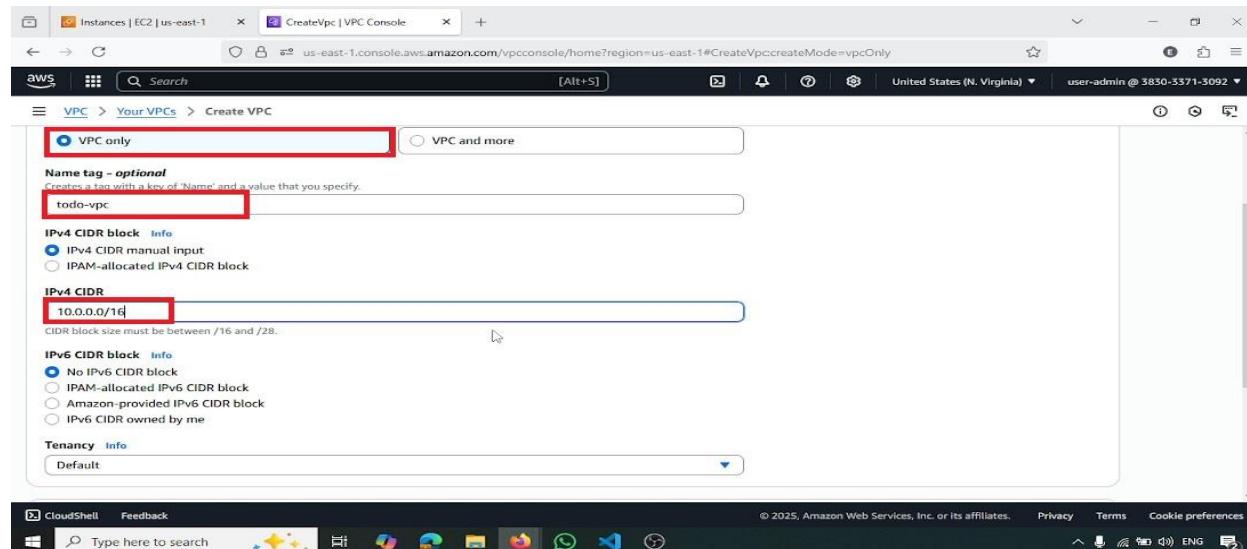
### Initiating VPC Creation

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



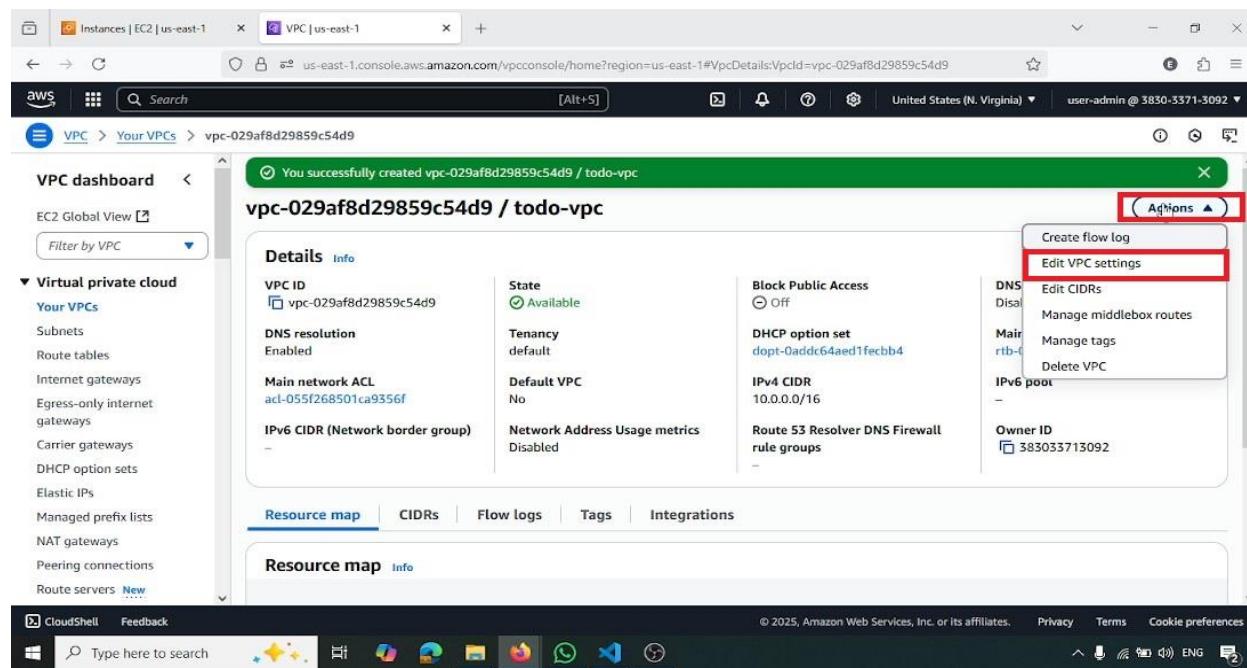
## Creating VPC with CIDR Block

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



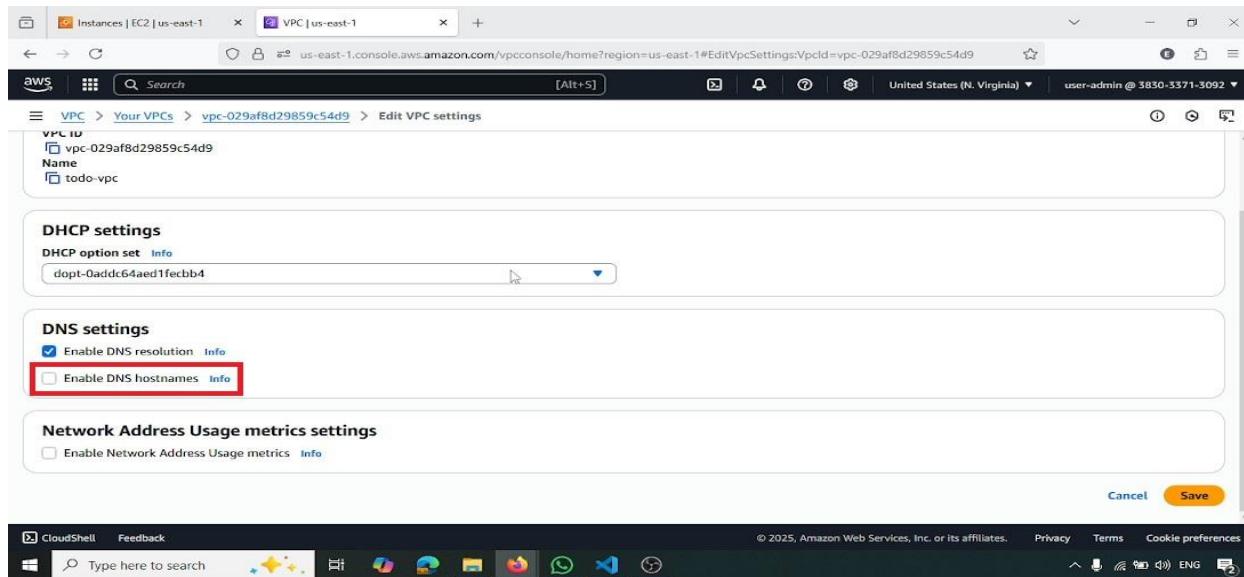
## Editing VPC Settings

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

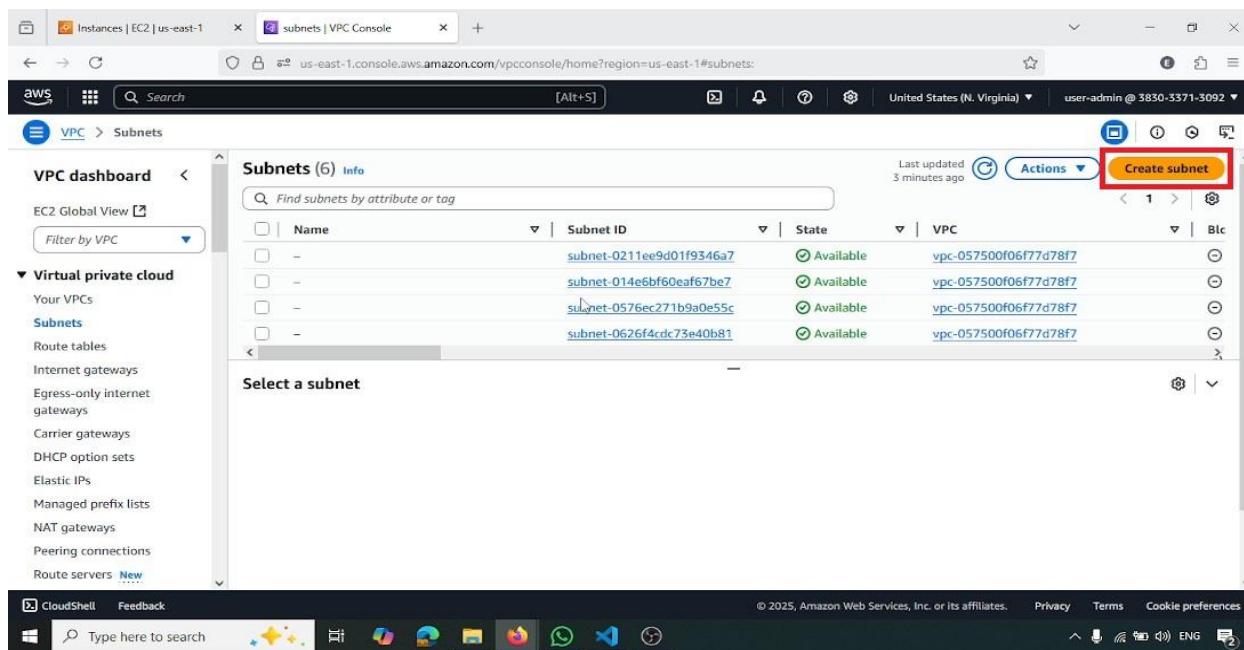
Within the VPC settings, the "Enable DNS hostnames" checkbox is highlighted and selected. This ensures that instances launched within this VPC will automatically receive public DNS hostnames, facilitating easier access and management



The screenshot shows the 'Edit VPC settings' page for a VPC named 'todo-vpc'. In the 'DNS settings' section, the 'Enable DNS hostnames' checkbox is checked and highlighted with a red box. Other options like 'Enable DNS resolution' and 'Network Address Usage metrics' are also present.

## 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



The screenshot shows the 'Subnets' page with 6 subnets listed. The 'Actions' dropdown menu at the top right has a 'Create subnet' button highlighted with a red box. The subnets table includes columns for Name, Subnet ID, State, and VPC.

Name	Subnet ID	State	VPC
-	subnet-0211ee9d01f9346a7	Available	vpc-057500f06f77d78f7
-	subnet-014ebf60eaf67be7	Available	vpc-057500f06f77d78f7
-	subnet-0576ec271b9a0e55c	Available	vpc-057500f06f77d78f7
-	subnet-0626f4cdc73e40h81	Available	vpc-057500f06f77d78f7



## 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.

The screenshot shows the AWS VPC console interface. The top navigation bar includes tabs for 'Instances | EC2 | us-east-1' and 'VPC | us-east-1'. The main title is 'Create subnet'. A search bar is at the top right. Below it, the breadcrumb trail shows 'VPC > Subnets > Create subnet'. The main content area is titled 'Create subnet' with a 'Info' link. A section labeled 'VPC' has a 'VPC ID' label and a note 'Create subnets in this VPC...'. A dropdown menu is open, showing two options: 'vpc-057500f06f77d78f7 172.31.0.0/16 (default)' and 'vpc-029af8fd29859c54d9 (todo-vpc) 10.0.0.0/16'. The second option is highlighted with a red box. Below the dropdown, a message says 'Select a VPC first to create new subnets.' A 'Add new subnet' button is available. At the bottom right are 'Cancel' and 'Create subnet' buttons.

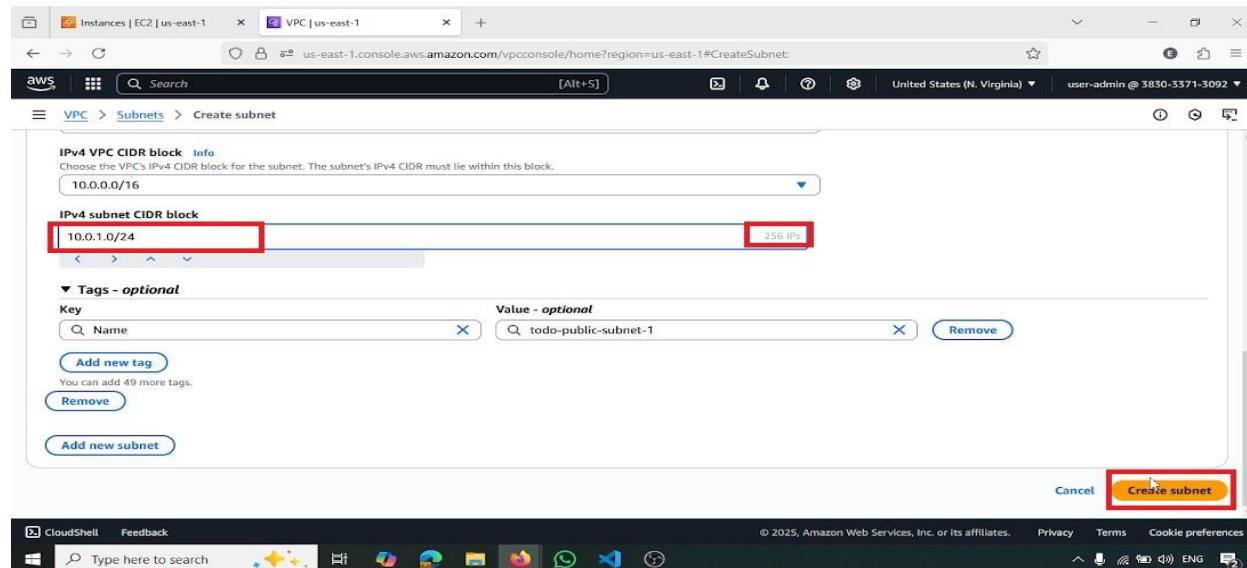
## Selecting Availability Zone for First Public Subnet

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.

The screenshot shows the AWS VPC console interface for creating a new subnet. The top navigation bar includes tabs for Instances, VPC, and the current page, VPC | us-east-1. The URL in the address bar is [us-east-1.console.aws.amazon.com/vpcconsole/home?region=us-east-1#CreateSubnet](https://us-east-1.console.aws.amazon.com/vpcconsole/home?region=us-east-1#CreateSubnet). The user is signed in as 'user-admin @ 3850-3371-3092'. The main content area shows the 'Create subnet' wizard with the first step, 'Specify the CIDR blocks and availability zone for the subnet'. The 'Subnet 1 of 1' section has a 'Subnet name' field containing 'todo-public-subnet-1', which is highlighted with a red box. Below it, a note says 'The name can be up to 256 characters long.' The 'Availability Zone' section shows a dropdown menu with several options. One option, 'United States (N. Virginia) / us-east-1d', is also highlighted with a red box. The 'Tags - optional' section at the bottom is partially visible.

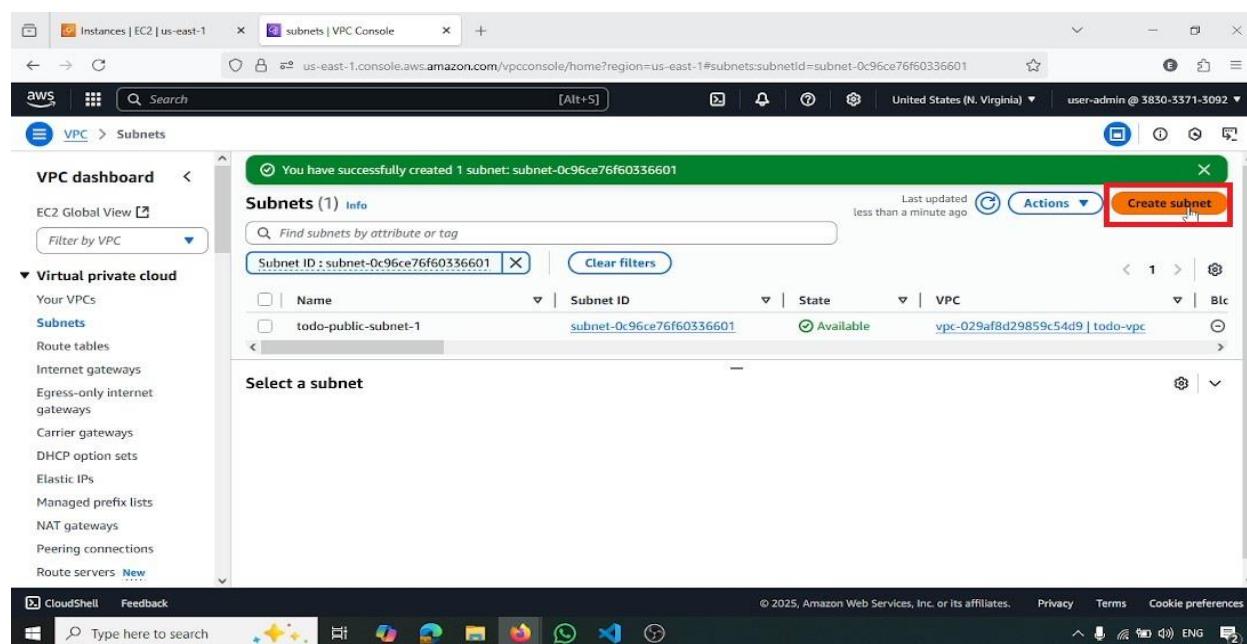
## Creating First Public Subnet

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



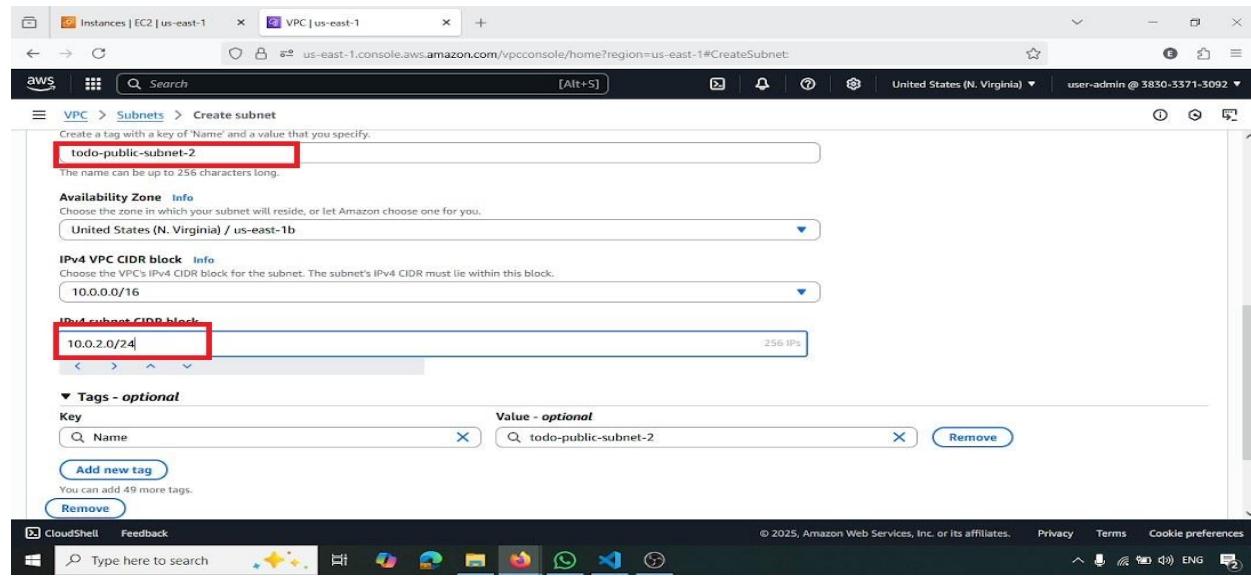
## Creating Second Public Subnet

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



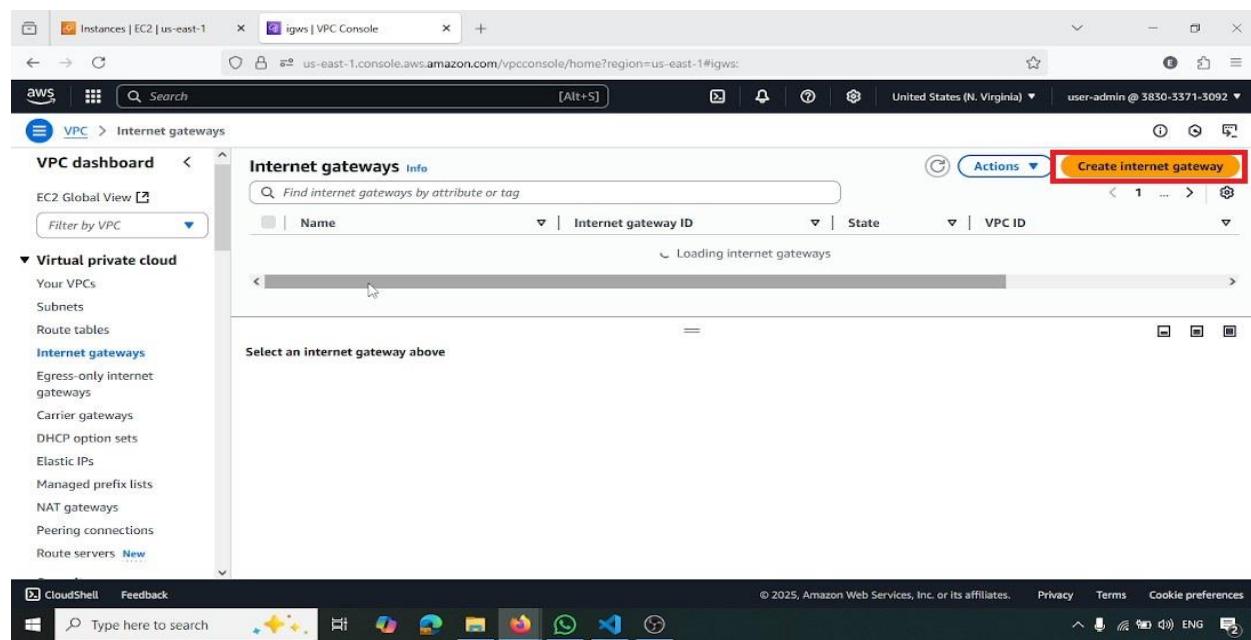
## Creating Second Public Subnet

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



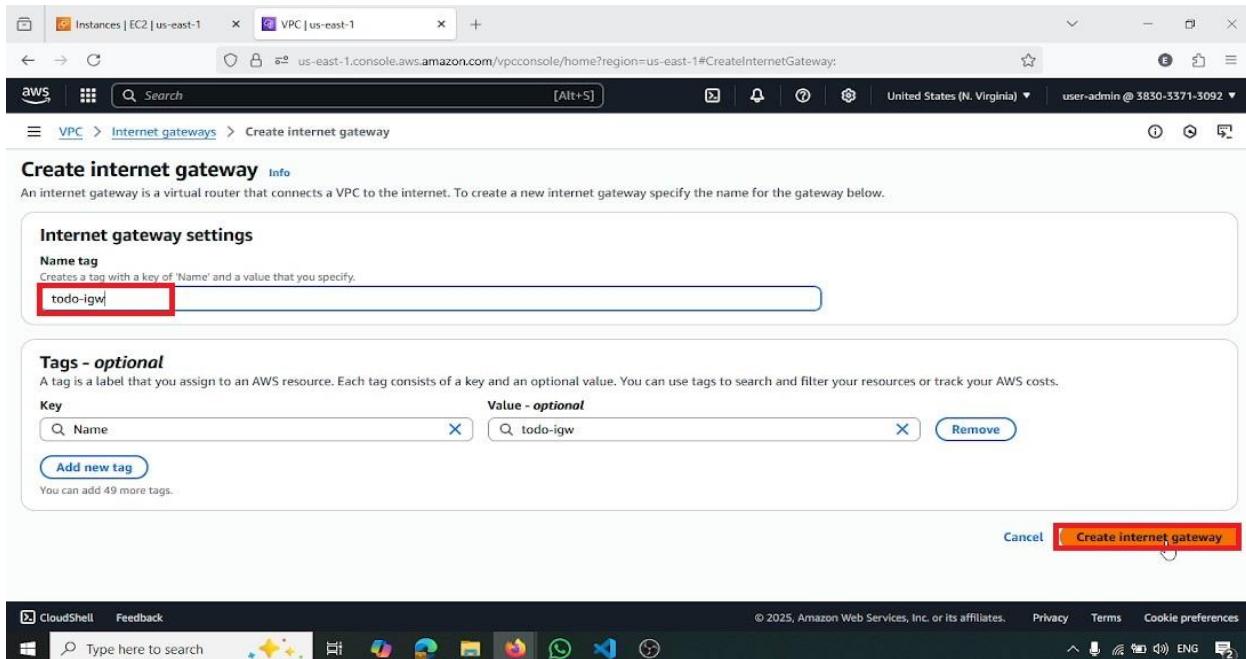
## Initiating Internet Gateway Creation

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

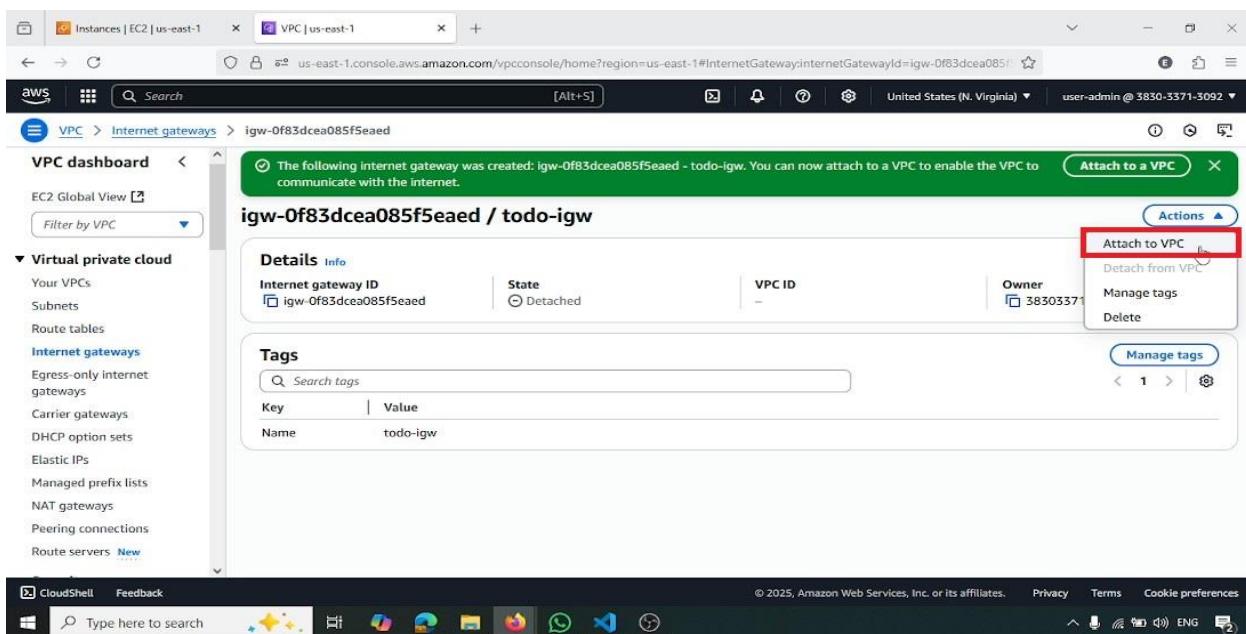
The "Create internet gateway" wizard shows the "Name tag" entered as todo-igw. This step assigns a descriptive name to the internet gateway



The screenshot shows the 'Create internet gateway' wizard. In the 'Internet gateway settings' section, the 'Name tag' field contains 'todo-igw'. Below it, the 'Tags - optional' section shows a single tag 'Name: todo-igw'. At the bottom right, there is a red box around the 'Create internet gateway' button.

## Attaching Internet Gateway Action

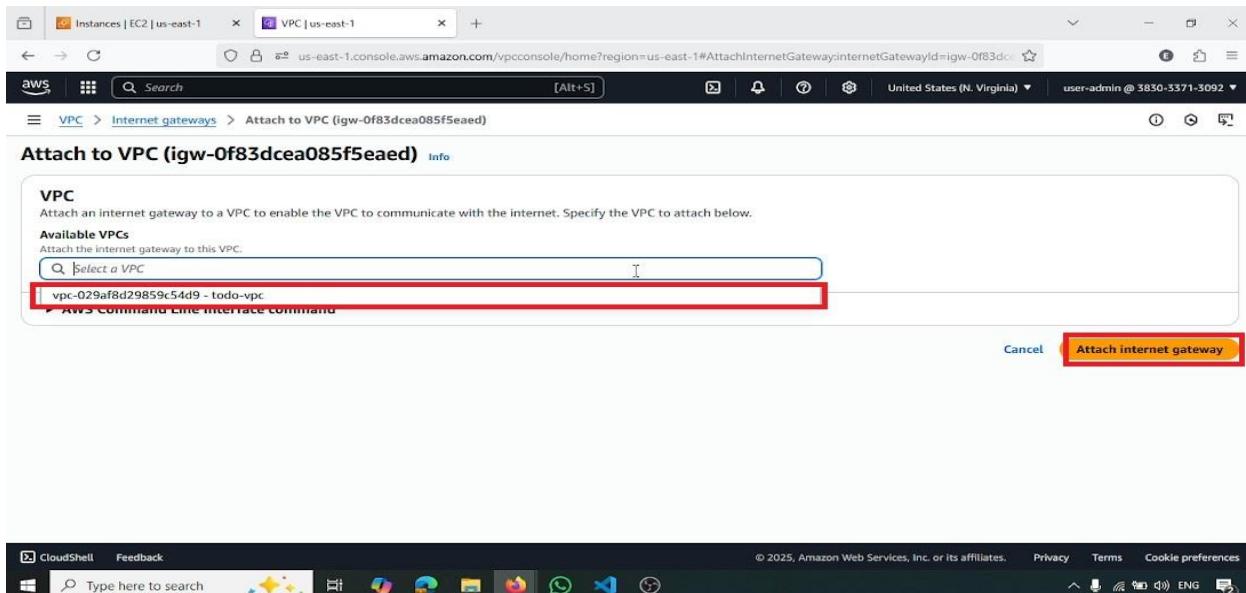
This image shows the "Actions" dropdown for the newly created internet gateway, with "Attach to VPC" highlighted. This action is taken to link the internet gateway to the todo-vpc



The screenshot shows the 'Internet Gateways' page. A green banner at the top says 'The following internet gateway was created: igw-0f83dcea085f5eaed - todo-igw. You can now attach to a VPC to enable the VPC to communicate with the internet.' Below it, the 'igw-0f83dcea085f5eaed / todo-igw' card has an 'Actions' dropdown with a red box around the 'Attach to VPC' option. Other options in the dropdown include 'Detach from VPC', 'Manage tags', and 'Delete'.

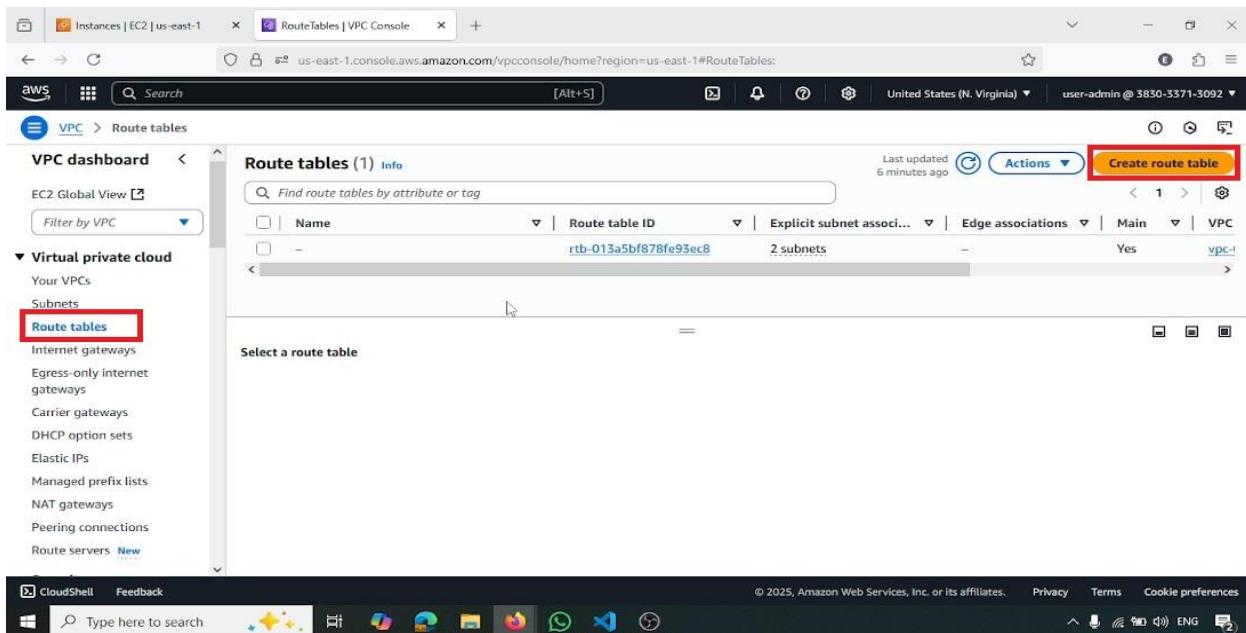
## Attaching Internet Gateway to VPC

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



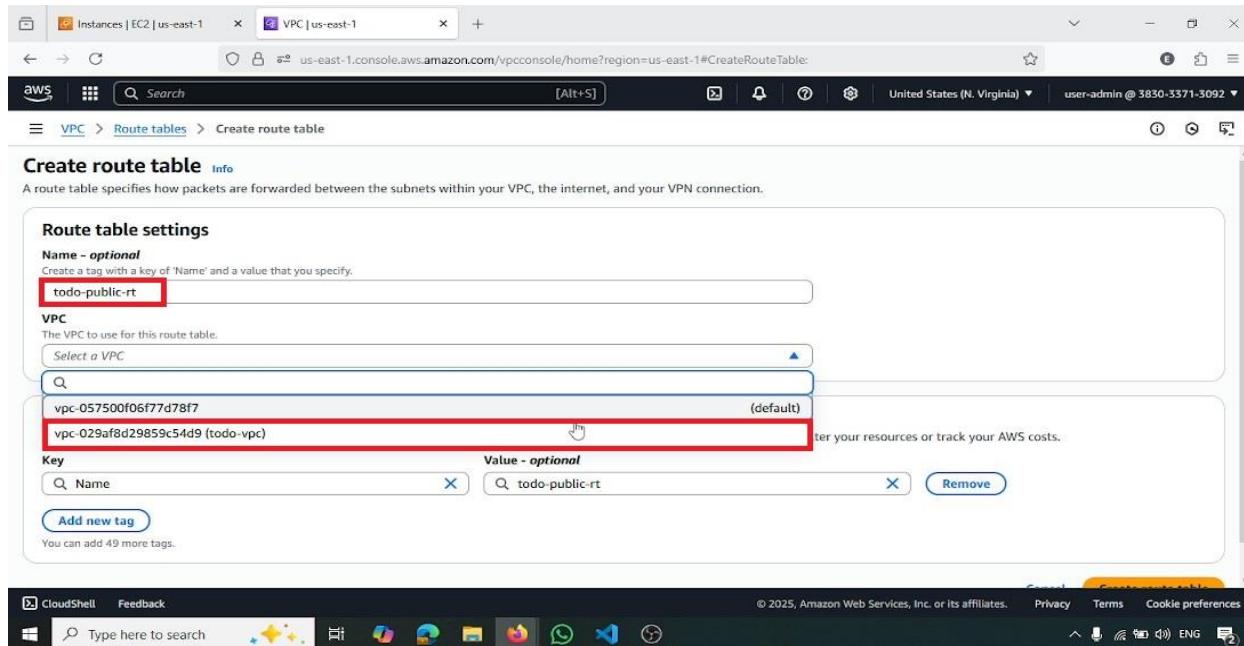
## Initiating Route Table Creation

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



**Create route table** Info

A route table specifies how packets are forwarded between the subnets within your VPC, the internet, and your VPN connection.

**Route table settings**

**Name - optional**  
Create a tag with a key of 'Name' and a value that you specify.  
**todo-public-rt**

**VPC**  
The VPC to use for this route table.  
**Select a VPC**

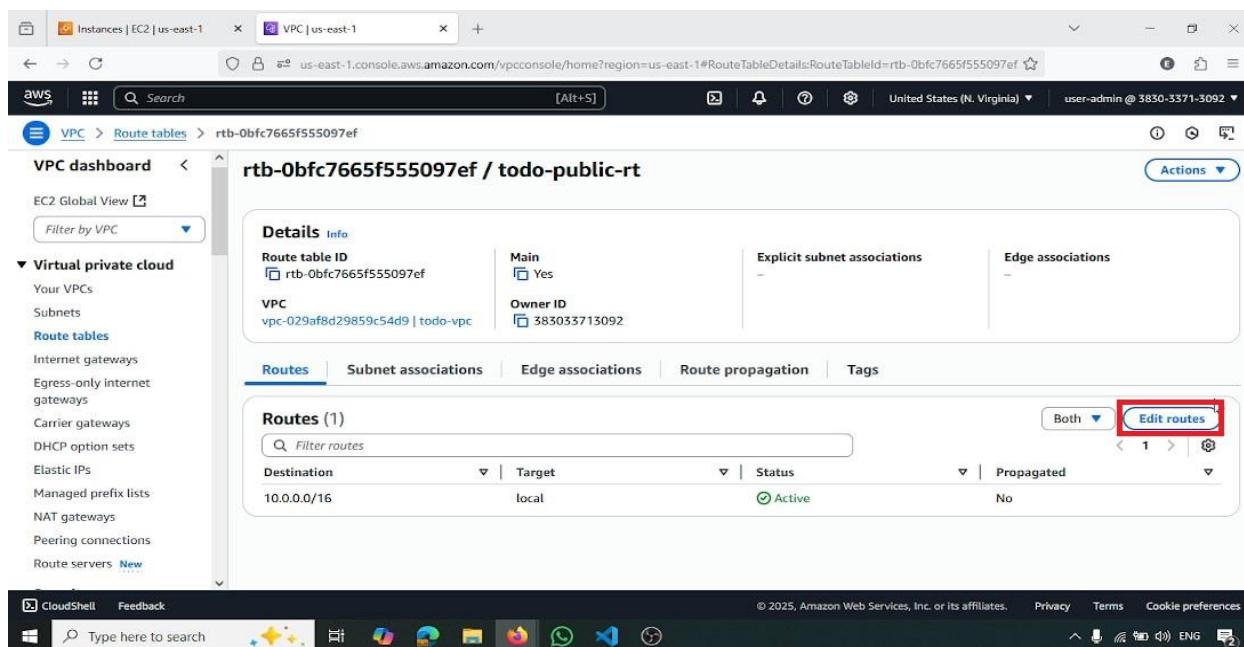
vpc-057500f06f77d78f7 (default)  
vpc-029af8d29859c54d9 (todo-vpc)

**Key** **Value - optional**  
Name todo-public-rt

**Add new tag**  
You can add 49 more tags.

## 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



**rtb-0bf7665f555097ef / todo-public-rt**

**Details** Info

**Route table ID**: rtb-0bf7665f555097ef  
**Main**: Yes  
**Owner ID**: 383033713092

**Explicit subnet associations**:  
**Edge associations**:

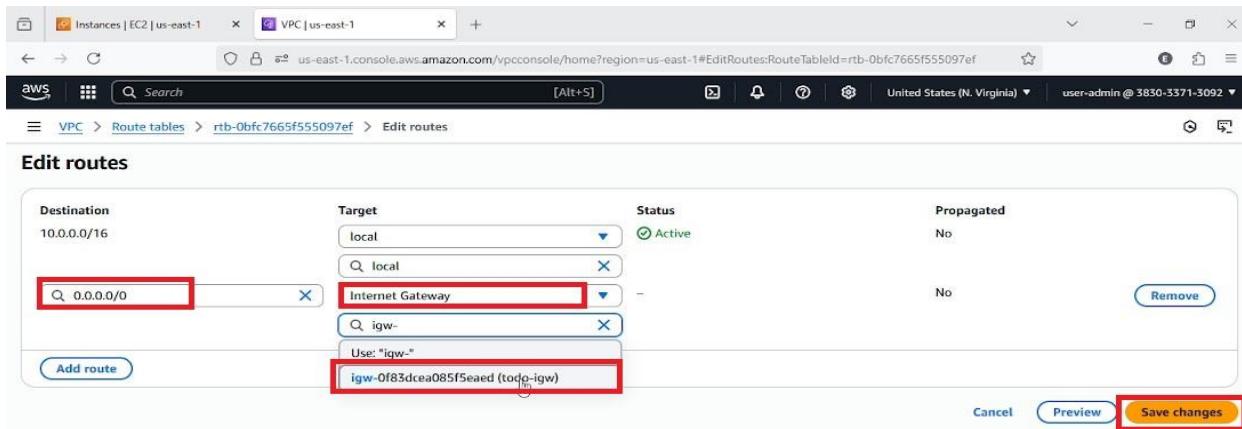
**Routes** Subnet associations Edge associations Route propagation Tags

**Routes (1)**

Destination	Target	Status	Propagated
10.0.0.0/16	local	Active	No

## 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



The screenshot shows the AWS VPC console with the URL <https://us-east-1.console.aws.amazon.com/vpcconsole/home?region=us-east-1#EditRoutes:RouteTableId=rtb-0bfc7665f555097ef>. The 'Edit routes' page is displayed for route table `rtb-0bfc7665f555097ef`. A new route is being added with the following details:

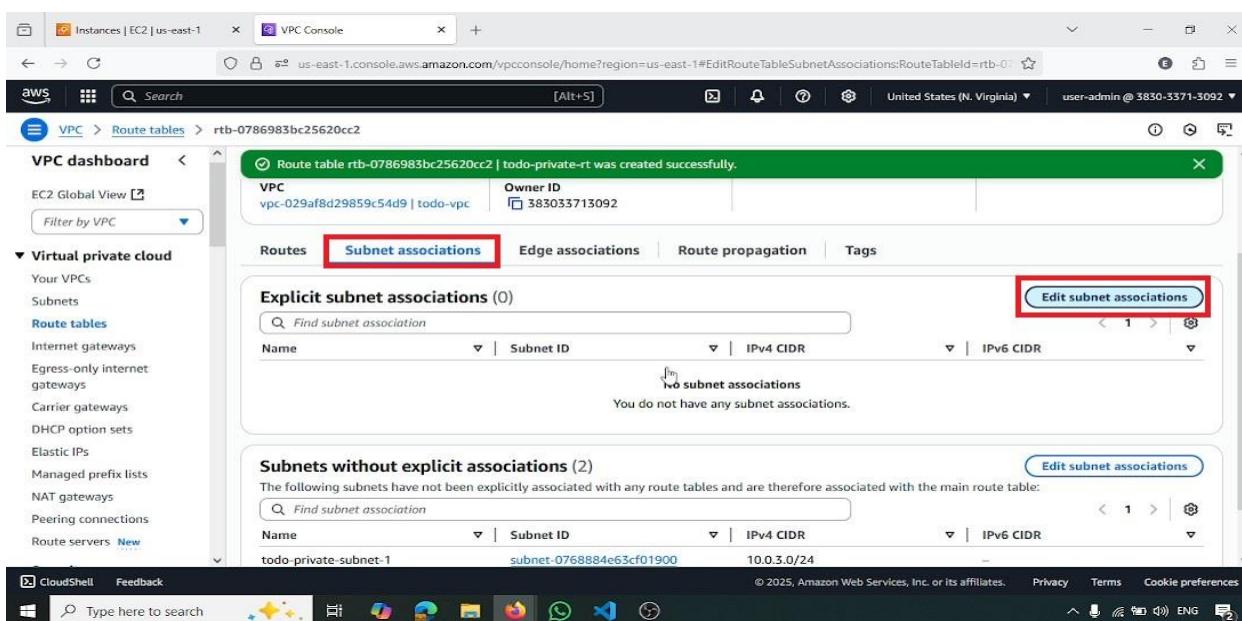
Destination	Target	Status	Propagated
10.0.0.0/16	local	Active	No
0.0.0.0/0	Internet Gateway	-	No
	igw-		
	Use: "igw-"		
	igw-0f83dcea085f5eaed (todo-igw)		

The 'Save changes' button at the bottom right is highlighted with a red box.



## Editing Subnet Associations for Route Table

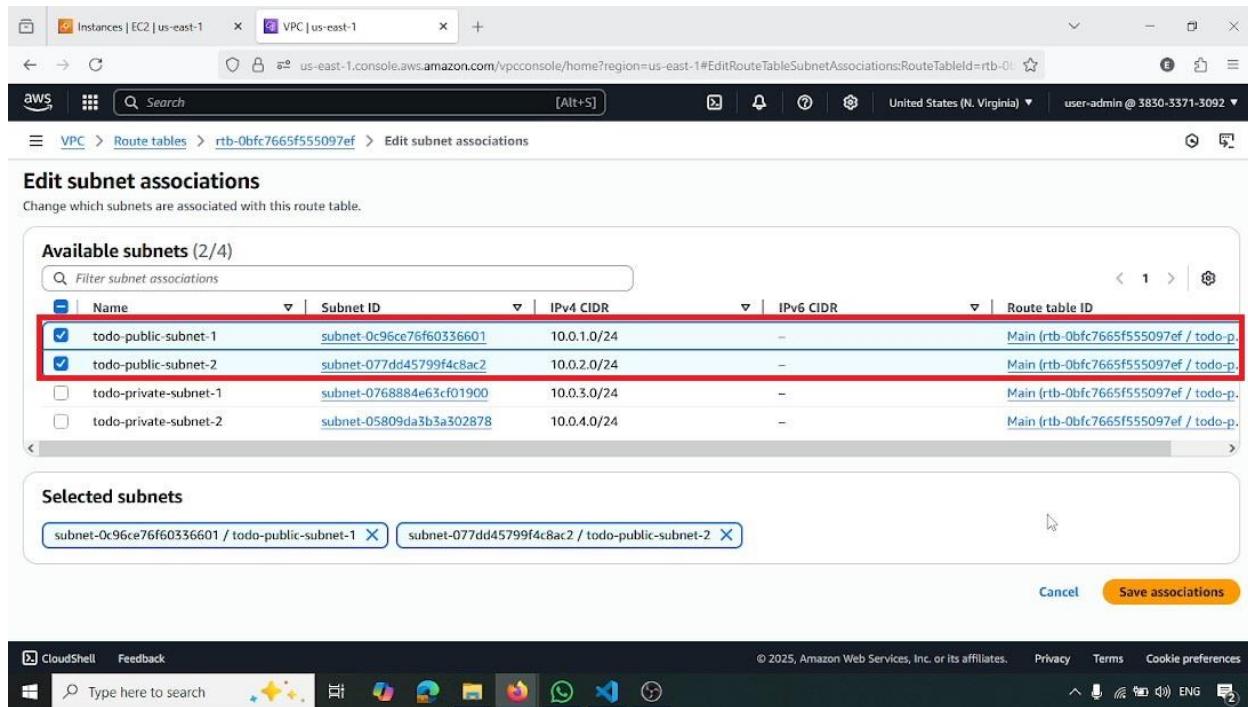
This image shows the "Edit subnet associations" button highlighted within the todo-public-rt details. This action is taken to link the public subnets to this route table



The screenshot shows the AWS VPC console with the URL <https://us-east-1.console.aws.amazon.com/vpcconsole/home?region=us-east-1>EditRouteTableSubnetAssociations:RouteTableId=rtb-0786983bc25620cc2>. The 'Subnet associations' tab is selected for route table `rtb-0786983bc25620cc2`. The 'Edit subnet associations' button is highlighted with a red box.

## 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



**Available subnets (2/4)**

Name	Subnet ID	IPv4 CIDR	IPv6 CIDR	Route table ID
<input checked="" type="checkbox"/> todo-public-subnet-1	subnet-0c96ce76f60336601	10.0.1.0/24	-	Main (rtb-0bf7665f555097ef / todo-p.)
<input checked="" type="checkbox"/> todo-public-subnet-2	subnet-077dd45799f4c8ac2	10.0.2.0/24	-	Main (rtb-0bf7665f555097ef / todo-p.)
<input type="checkbox"/> todo-private-subnet-1	subnet-0768884e63cf01900	10.0.3.0/24	-	Main (rtb-0bf7665f555097ef / todo-p.)
<input type="checkbox"/> todo-private-subnet-2	subnet-05809da3b3a302878	10.0.4.0/24	-	Main (rtb-0bf7665f555097ef / todo-p.)

**Selected subnets**

- subnet-0c96ce76f60336601 / todo-public-subnet-1
- subnet-077dd45799f4c8ac2 / todo-public-subnet-2

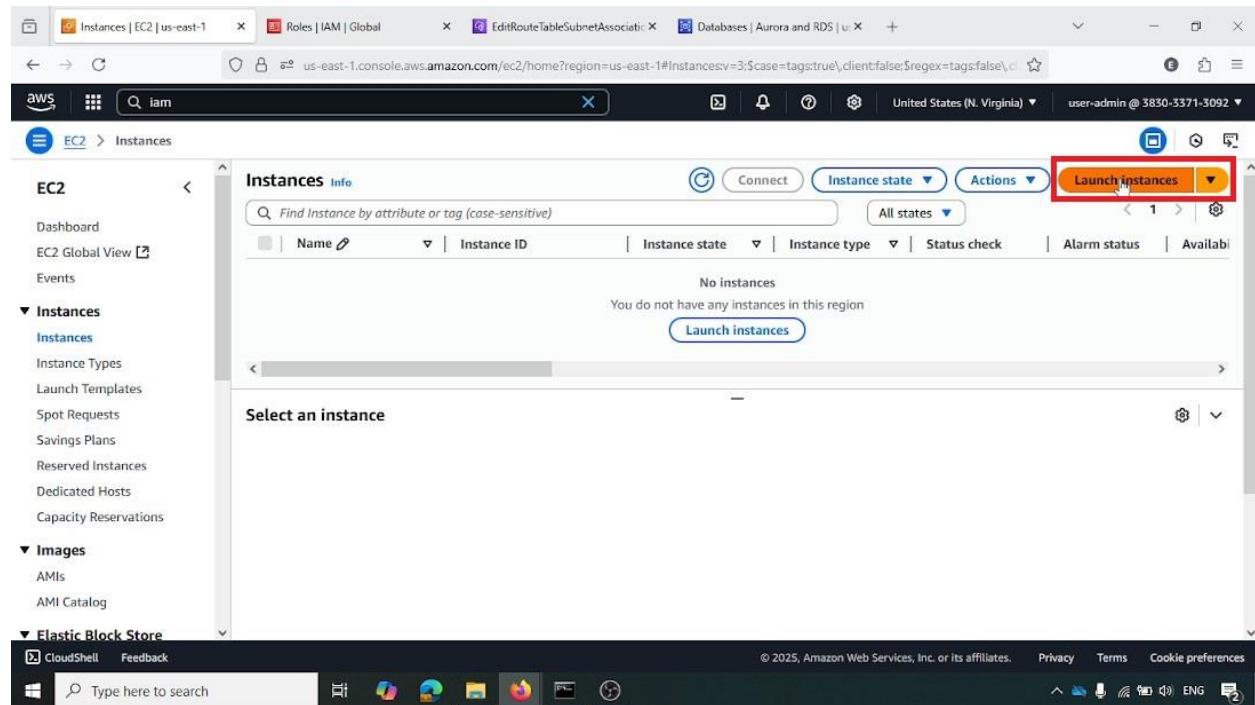
**Buttons:** Cancel, Save associations

## 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.

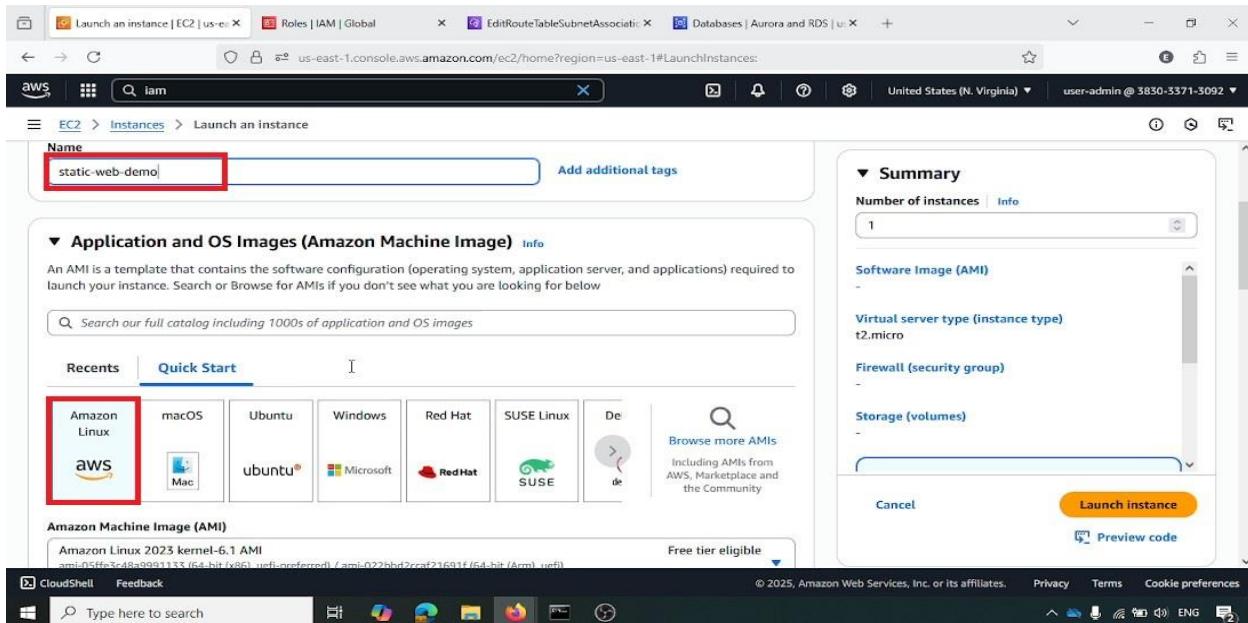
### Initiating EC2 Instance Launch

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

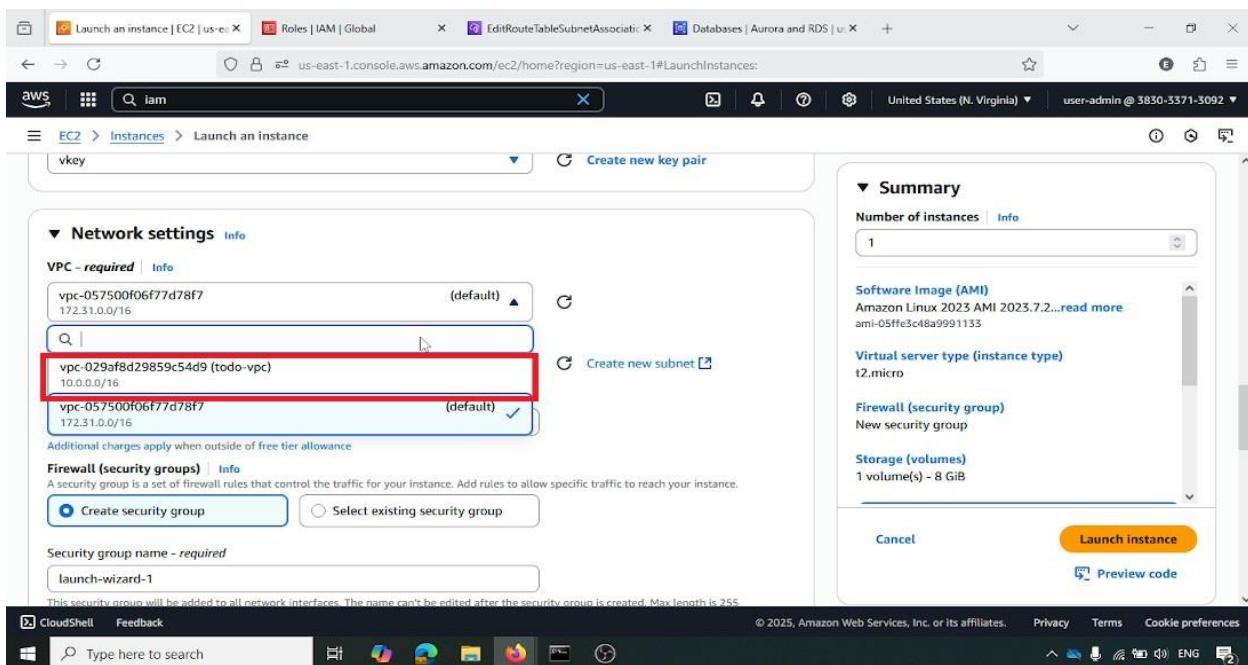
The "Launch an instance" wizard shows the instance named static-web-demo and "Amazon Linux" selected as the base operating system image



The screenshot shows the AWS Lambda console with the "Launch an instance" wizard open. The "Name" field contains "static-web-demo". In the "Application and OS Images (Amazon Machine Image)" section, "Amazon Linux" is selected, highlighted with a red box. The "Virtual server type (instance type)" is set to "t2.micro". The "Launch Instance" button is visible at the bottom right.

## Selecting VPC for Instance Launch

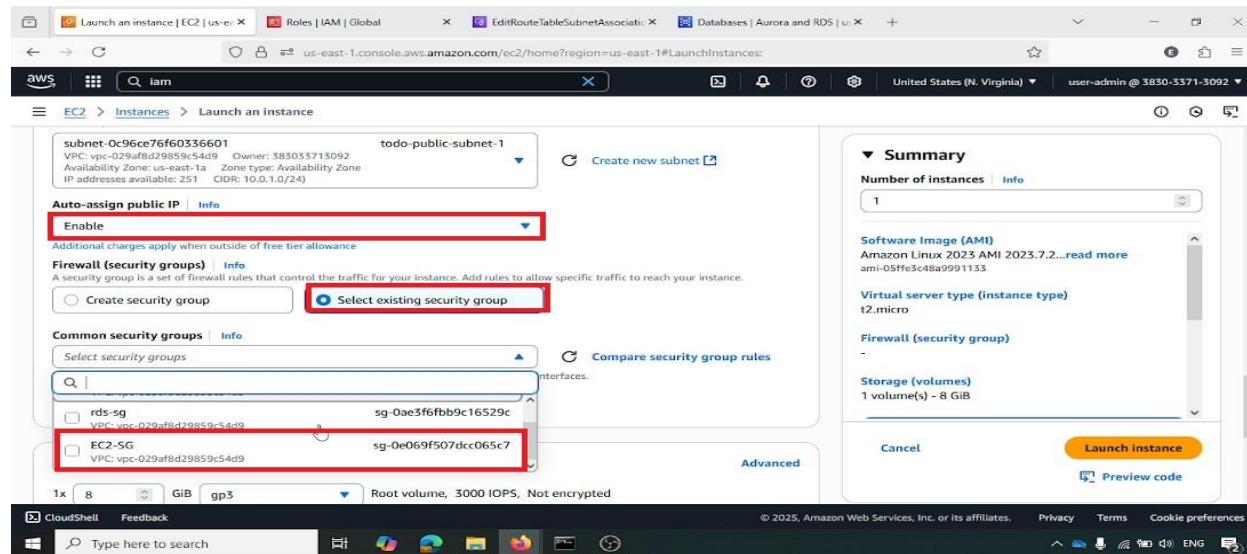
The todo-vpc (vpc-029af8d29859c54d9) is selected from the "VPC" dropdown, ensuring the instance is launched within the correct network



The screenshot shows the AWS Lambda console with the "Launch an instance" wizard open. In the "Network settings" section, "vpc-029af8d29859c54d9 (todo-vpc)" is selected from the "VPC" dropdown, highlighted with a red box. The "Virtual server type (instance type)" is set to "t2.micro". The "Launch Instance" button is visible at the bottom right.

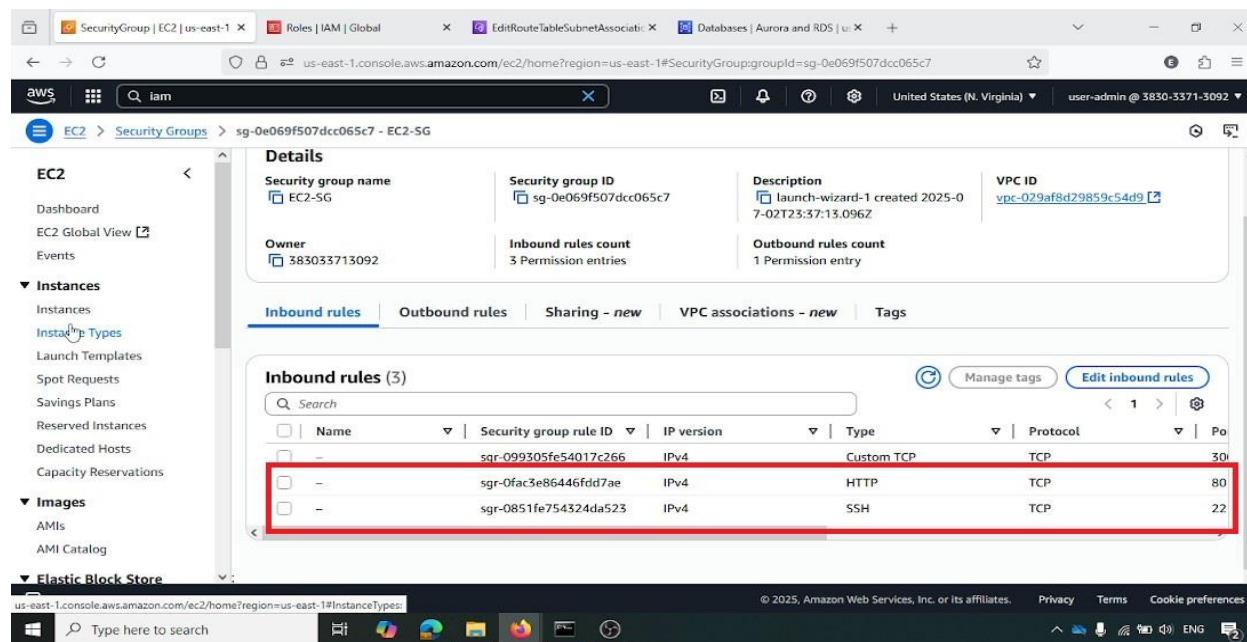
## Selecting Security Group for Instance Launch

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



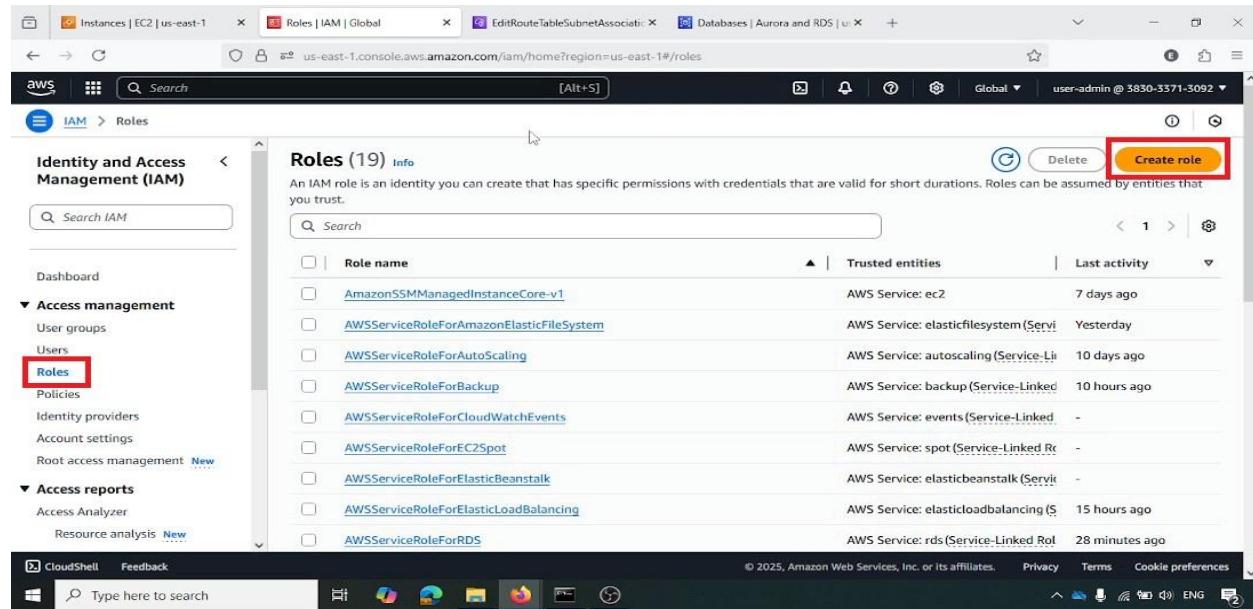
## Security Group Inbound Rules

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

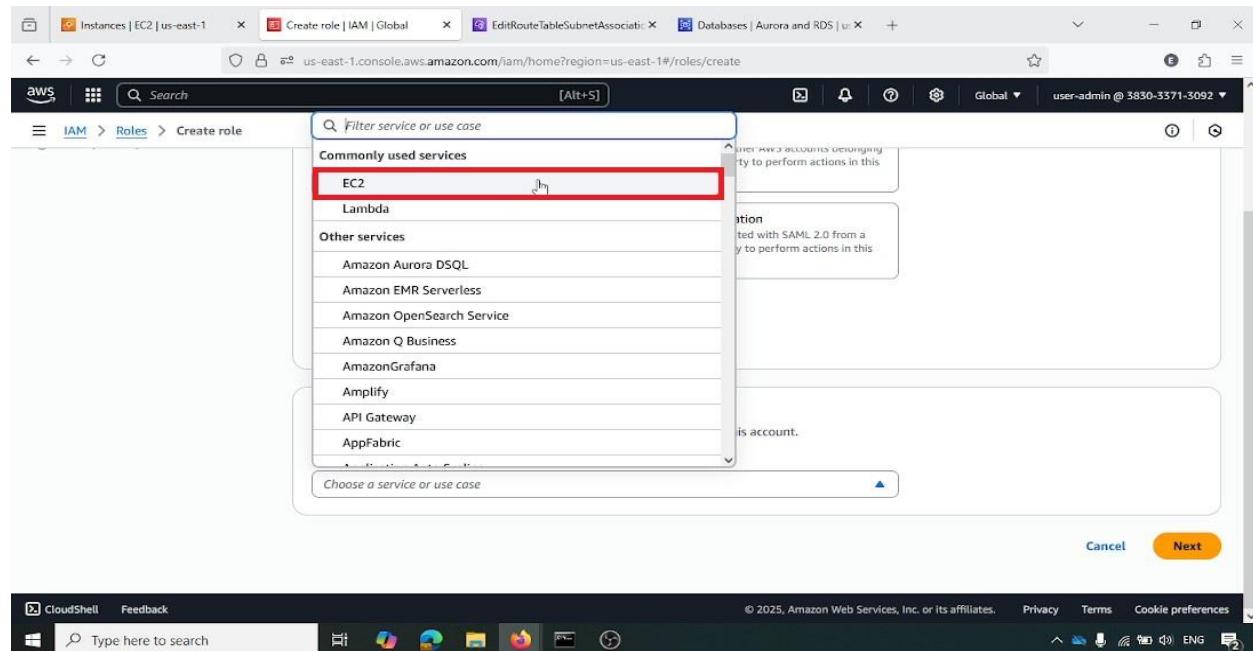


The screenshot shows the AWS IAM Roles page. On the left sidebar, under 'Access management', the 'Roles' option is selected and highlighted with a red box. At the top right, there is a yellow 'Create role' button, which is also highlighted with a red box. The main area displays a table of existing IAM roles, each with a checkbox, role name, trusted entity, and last activity.

Role name	Trusted entities	Last activity
AmazonSSManagedInstanceCore-v1	AWS Service: ec2	7 days ago
AWSServiceRoleForAmazonElasticFileSystem	AWS Service: elasticfilesystem (Service-Linked)	Yesterday
AWSServiceRoleForAutoScaling	AWS Service: autoscaling (Service-Linked)	10 days ago
AWSServiceRoleForBackup	AWS Service: backup (Service-Linked)	10 hours ago
AWSServiceRoleForCloudWatchEvents	AWS Service: events (Service-Linked)	-
AWSServiceRoleForEC2Spot	AWS Service: spot (Service-Linked Role)	-
AWSServiceRoleForElasticBeanstalk	AWS Service: elasticbeanstalk (Service-Linked)	-
AWSServiceRoleForElasticLoadBalancing	AWS Service: elasticloadbalancing (Service-Linked)	15 hours ago
AWSServiceRoleForRDS	AWS Service: rds (Service-Linked Role)	28 minutes ago

## Selecting EC2 as Trusted Entity

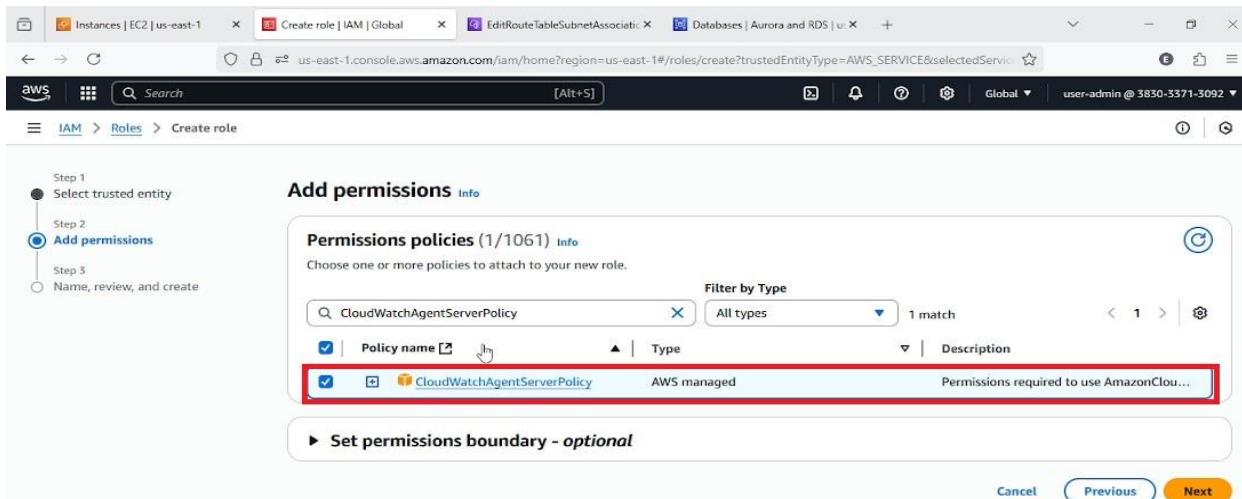
In the "Create role" wizard, "EC2" is selected as the trusted entity, allowing EC2 instances to assume this role and its associated permissions



The screenshot shows the 'Create role' wizard. In the 'Commonly used services' section, 'EC2' is listed and highlighted with a red box. Below it, other services like Lambda, Amazon Aurora DSQL, and Amazon EMR Serverless are listed. At the bottom of the dropdown, there is a 'Choose a service or use case' input field. The 'Next' button at the bottom right is highlighted with a yellow box.

## 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

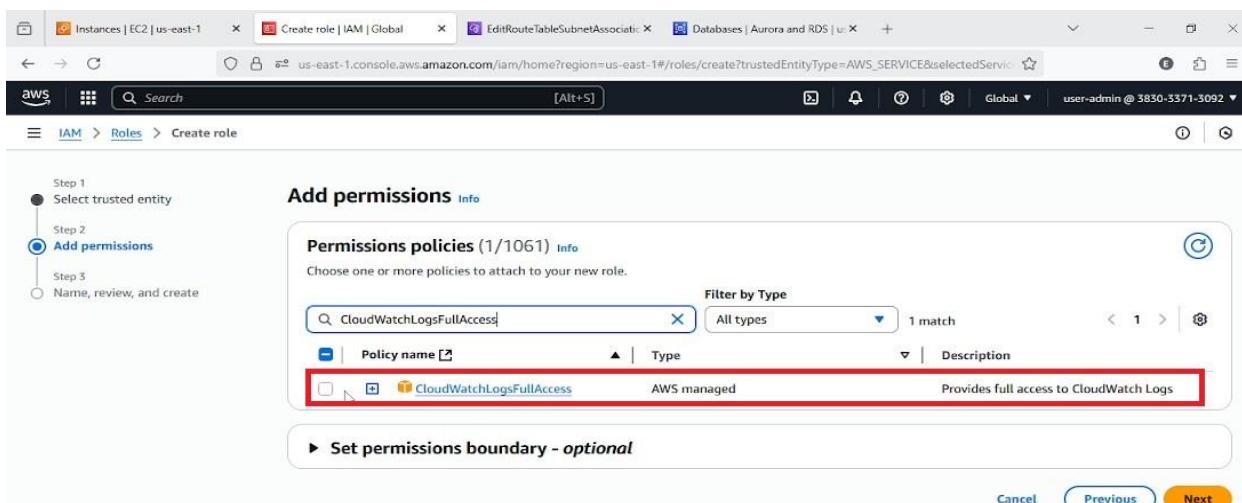


The screenshot shows the AWS IAM 'Create role' wizard. The user is on Step 2: Add permissions. In the search bar, 'CloudWatchAgentServerPolicy' is typed. The results show one policy: 'CloudWatchAgentServerPolicy' (AWS managed). This policy is highlighted with a red box. Below the search bar, there's a section titled 'Set permissions boundary - optional'.



## Attaching CloudWatchLogsFullAccess Policy

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

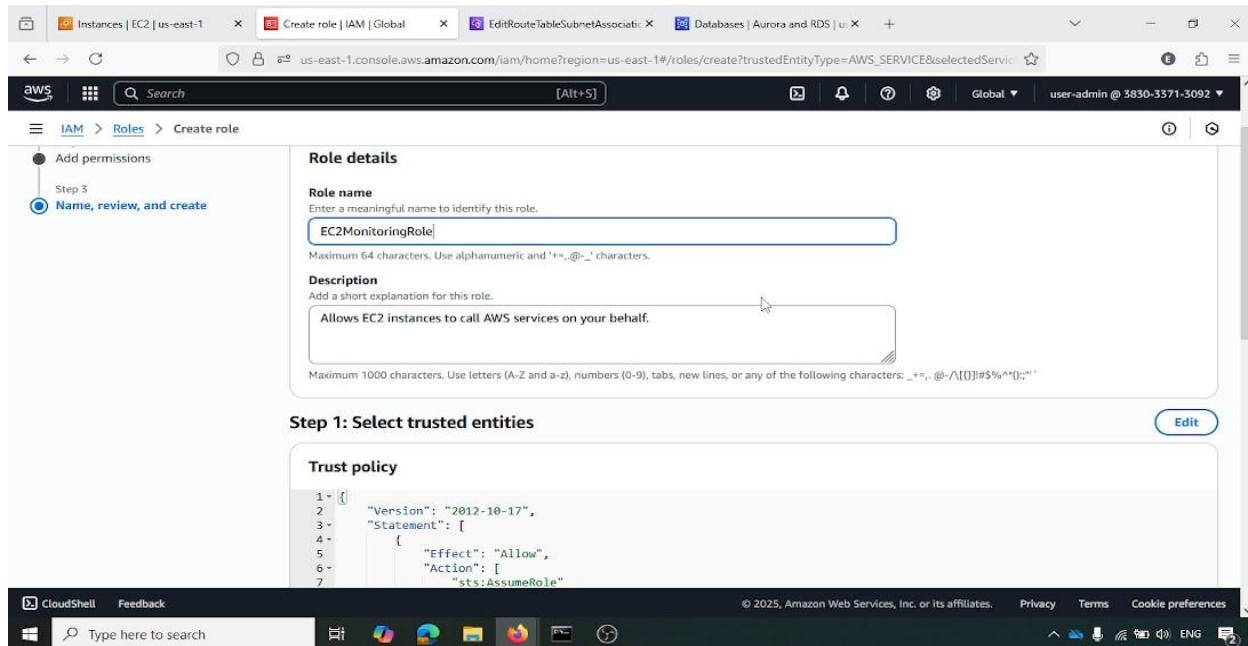


The screenshot shows the AWS IAM 'Create role' wizard. The user is on Step 2: Add permissions. In the search bar, 'CloudWatchLogsFullAccess' is typed. The results show one policy: 'CloudWatchLogsFullAccess' (AWS managed). This policy is highlighted with a red box. Below the search bar, there's a section titled 'Set permissions boundary - optional'.



## 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



**Role details**

**Role name**  
Enter a meaningful name to identify this role.  
**EC2MonitoringRole**

**Description**  
Add a short explanation for this role.  
Allows EC2 instances to call AWS services on your behalf.

**Step 1: Select trusted entities**

**Trust policy**

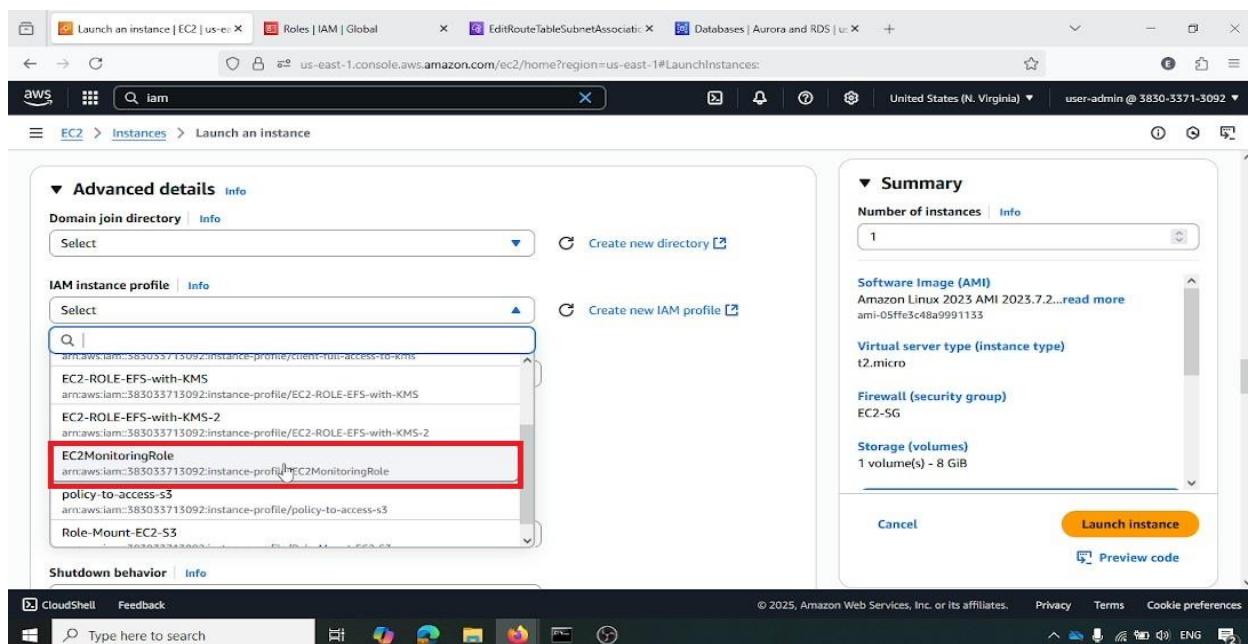
```

1 "Version": "2012-10-17",
2 "Statement": [
3     {
4         "Effect": "Allow",
5         "Action": [
6             "sts:AssumeRole"
7         ]
}

```

## 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



**Advanced details**

**IAM instance profile**

- Select
- Create new IAM profile

**EC2MonitoringRole** (highlighted with a red box)

**Summary**

Number of instances: 1

Software Image (AMI): Amazon Linux 2023 AMI 2023.7.2...read more

Virtual server type (instance type): t2.micro

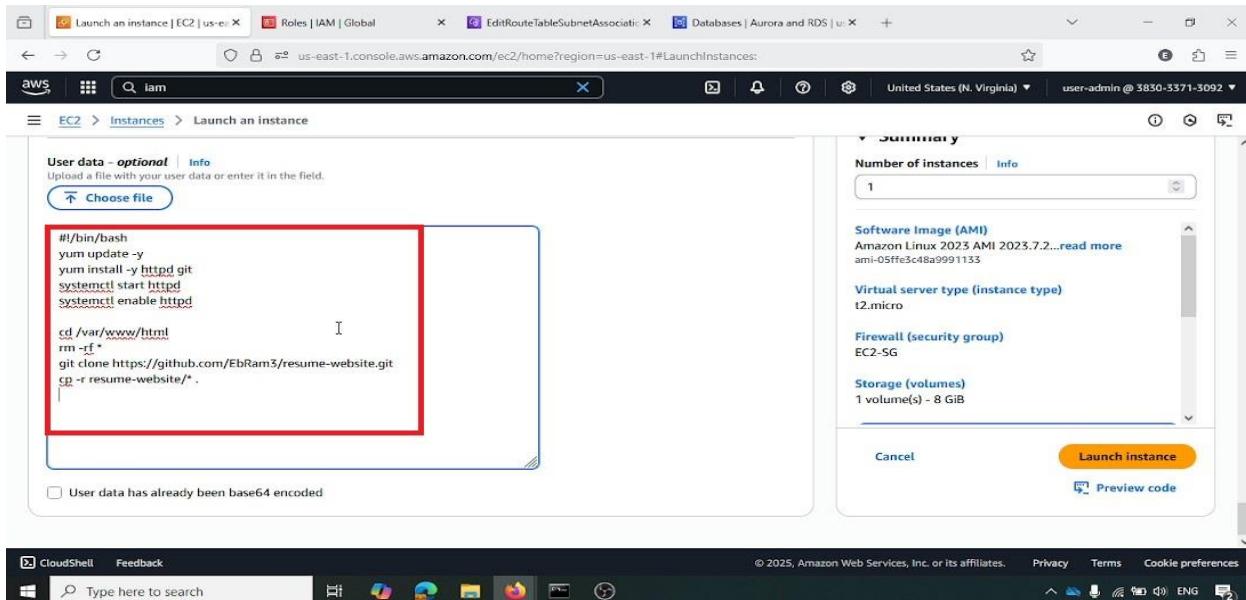
Firewall (security group): EC2-SG

Storage (volumes): 1 volume(s) - 8 GiB

**Launch instance**

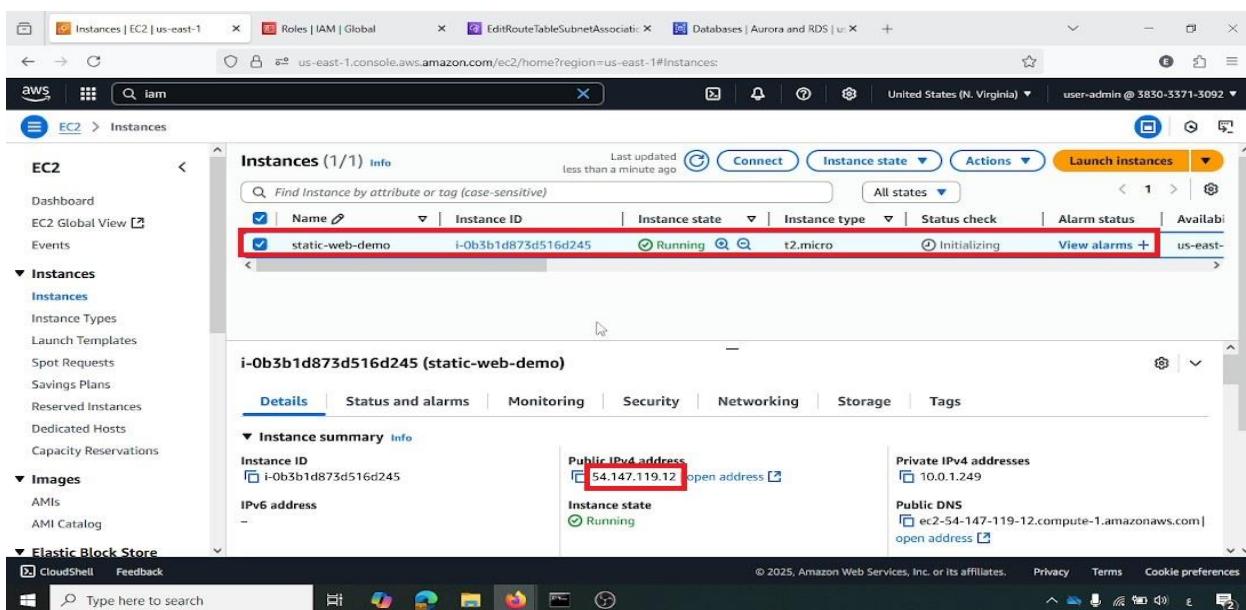
## User Data Script for Instance Launch

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



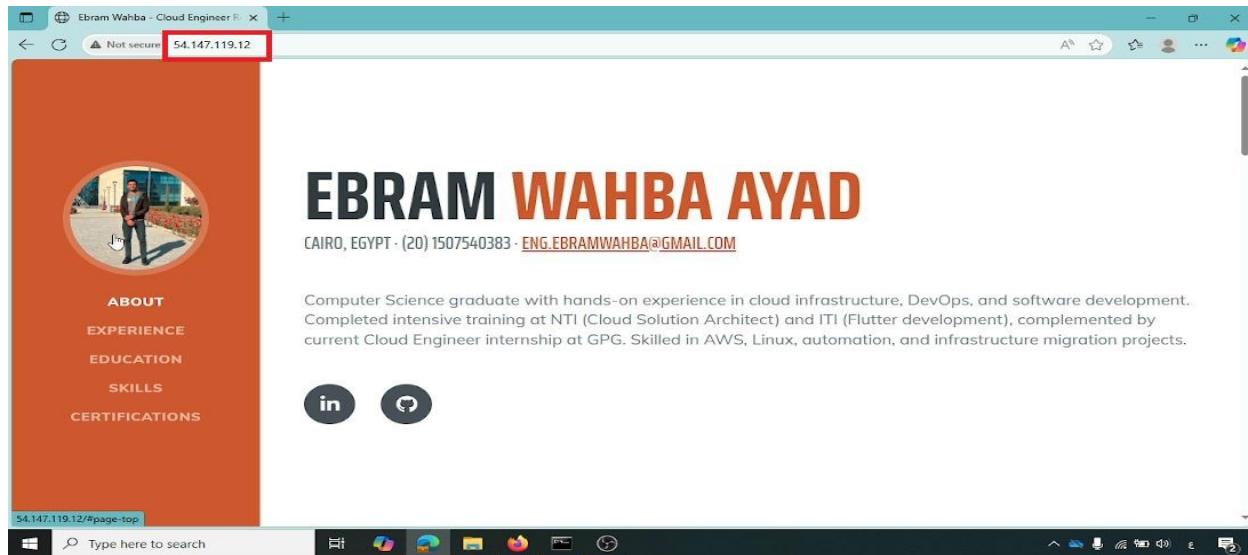
## EC2 Instance Public IP Address

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



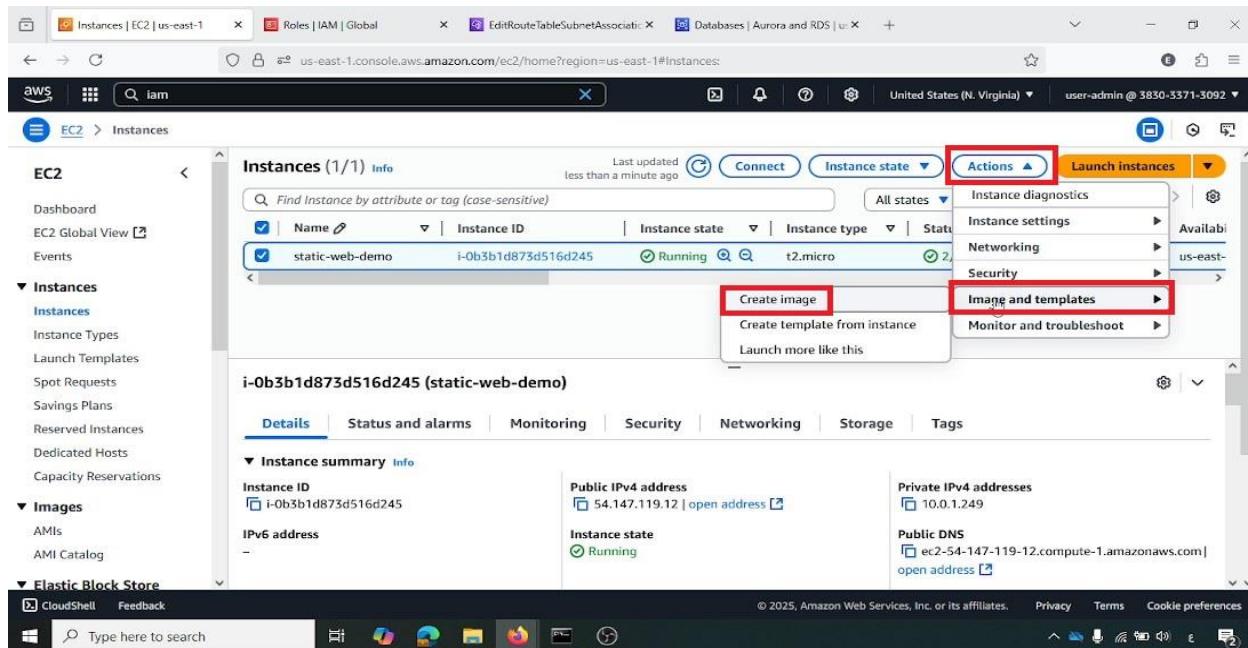
## Accessing Web Application via Public IP

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)



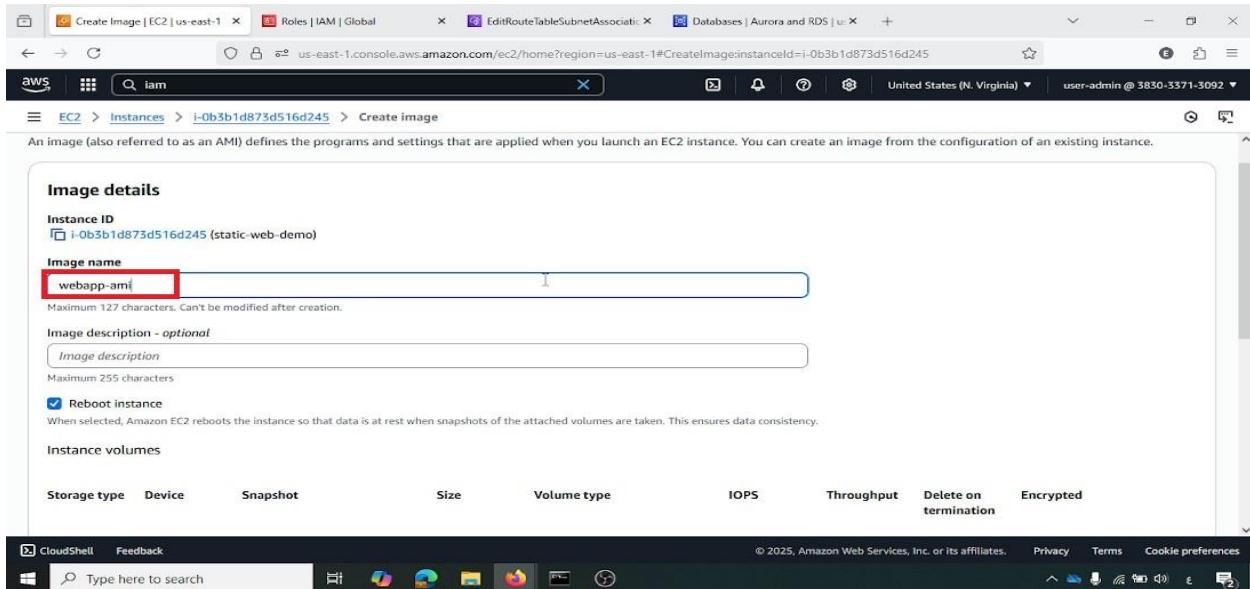
## Creating AMI from Instance

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



An image (also referred to as an AMI) defines the programs and settings that are applied when you launch an EC2 instance. You can create an image from the configuration of an existing instance.

**Image details**

Instance ID: i-0b3b1d873d516d245 (static-web-demo)

**Image name:**  Maximum 127 characters. Can't be modified after creation.

**Image description - optional:**  Maximum 255 characters

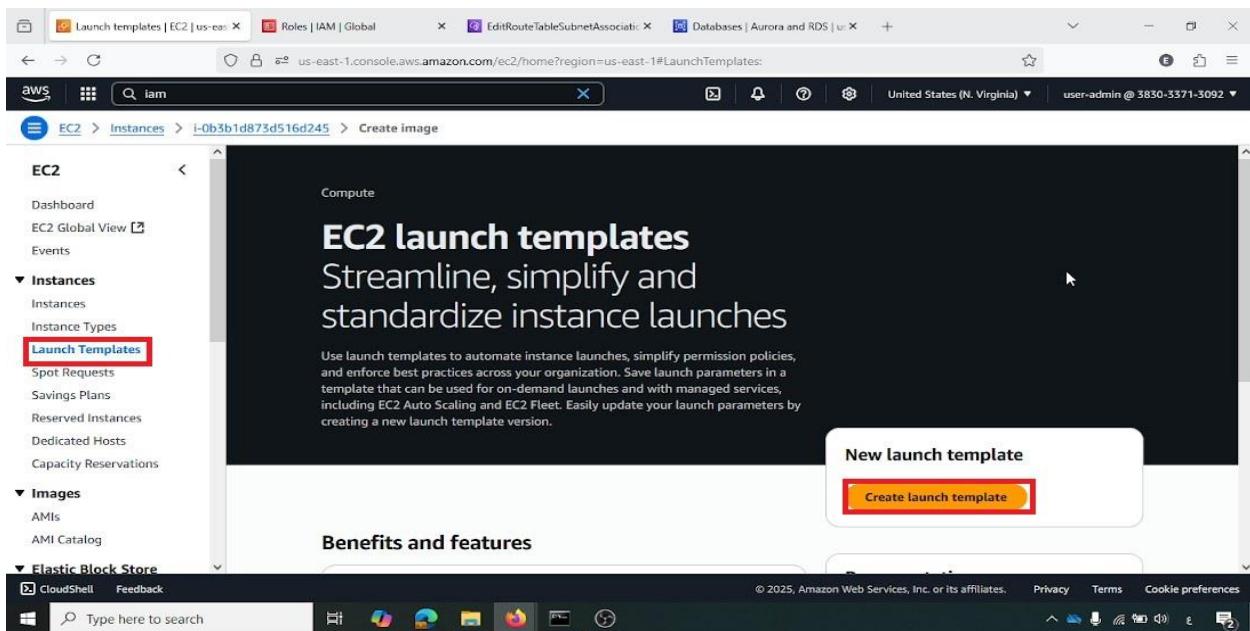
**Reboot instance**  
When selected, Amazon EC2 reboots the instance so that data is at rest when snapshots of the attached volumes are taken. This ensures data consistency.

**Instance volumes**

Storage type	Device	Snapshot	Size	Volume type	IOPS	Throughput	Delete on termination	Encrypted

## 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



Compute

## EC2 launch templates

Streamline, simplify and standardize instance launches

Use launch templates to automate instance launches, simplify permission policies, and enforce best practices across your organization. Save launch parameters in a template that can be used for on-demand launches and with managed services, including EC2 Auto Scaling and EC2 Fleet. Easily update your launch parameters by creating a new launch template version.

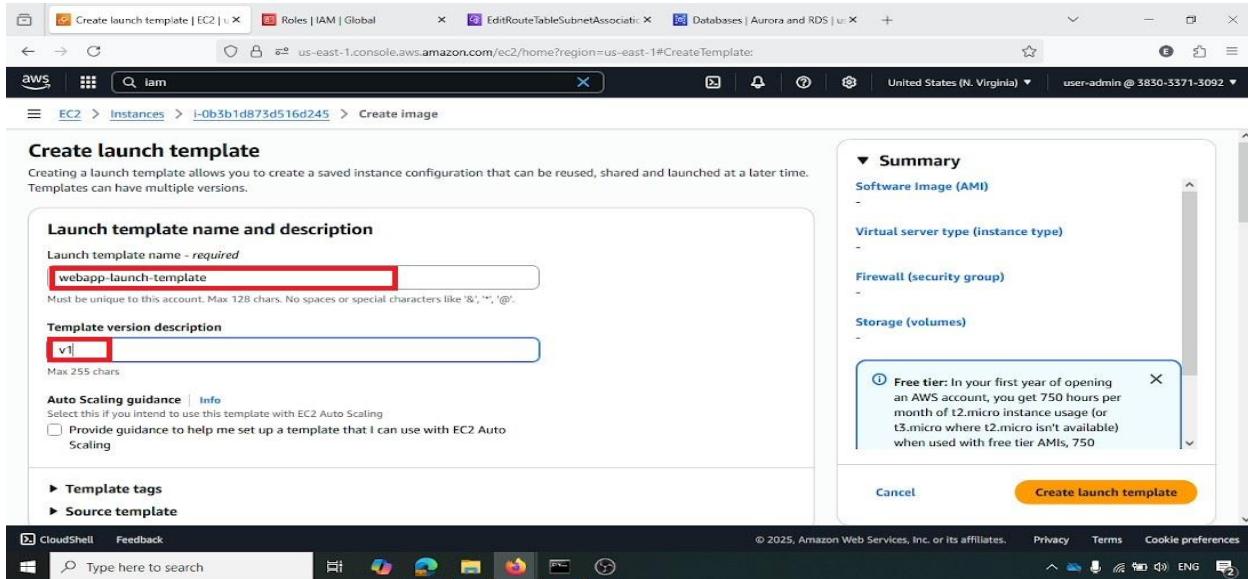
**New launch template**

**Create launch template**

**Benefits and features**

## 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



**Create launch template**

Creating a launch template allows you to create a saved instance configuration that can be reused, shared and launched at a later time. Templates can have multiple versions.

**Launch template name and description**

Launch template name - required  
webapp-launch-template

Must be unique to this account. Max 128 chars. No spaces or special characters like '&', '/', '@'.

Template version description  
v1

Max 255 chars

**Auto Scaling guidance** | Info  
Select this if you intend to use this template with EC2 Auto Scaling  
 Provide guidance to help me set up a template that I can use with EC2 Auto Scaling

**Summary**

Software Image (AMI)

Virtual server type (instance type)

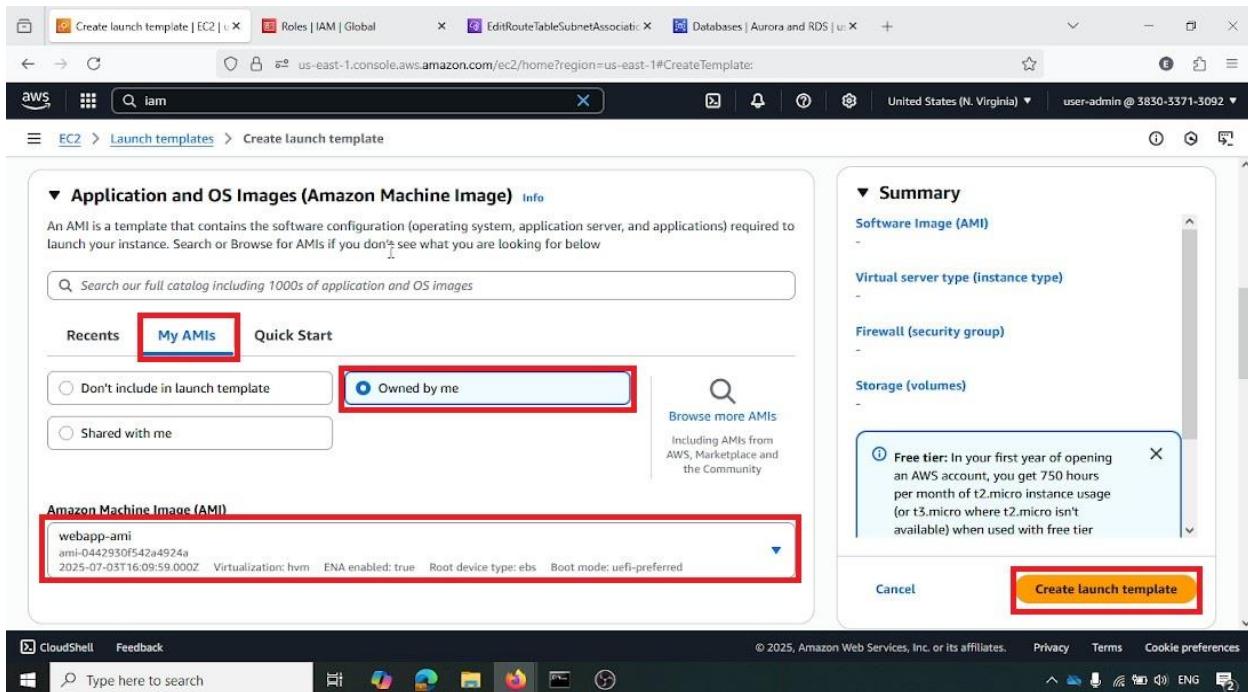
Firewall (security group)

Storage (volumes)

**Create launch 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



**Application and OS Images (Amazon Machine Image)** | Info

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Q Search our full catalog including 1000s of application and OS images

Recents    My AMIs    Quick Start

Don't include in launch template     Owned by me     Shared with me

Browse more AMIs  
Including AMIs from AWS, Marketplace and the Community

**Amazon Machine Image (AMI)**

webapp-ami  
ami-0442930f542a4924a  
2025-07-03T16:09:59.000Z Virtualization: hvm ENA enabled: true Root device type: ebs Boot mode: uefi-preferred

**Summary**

Software Image (AMI)

Virtual server type (instance type)

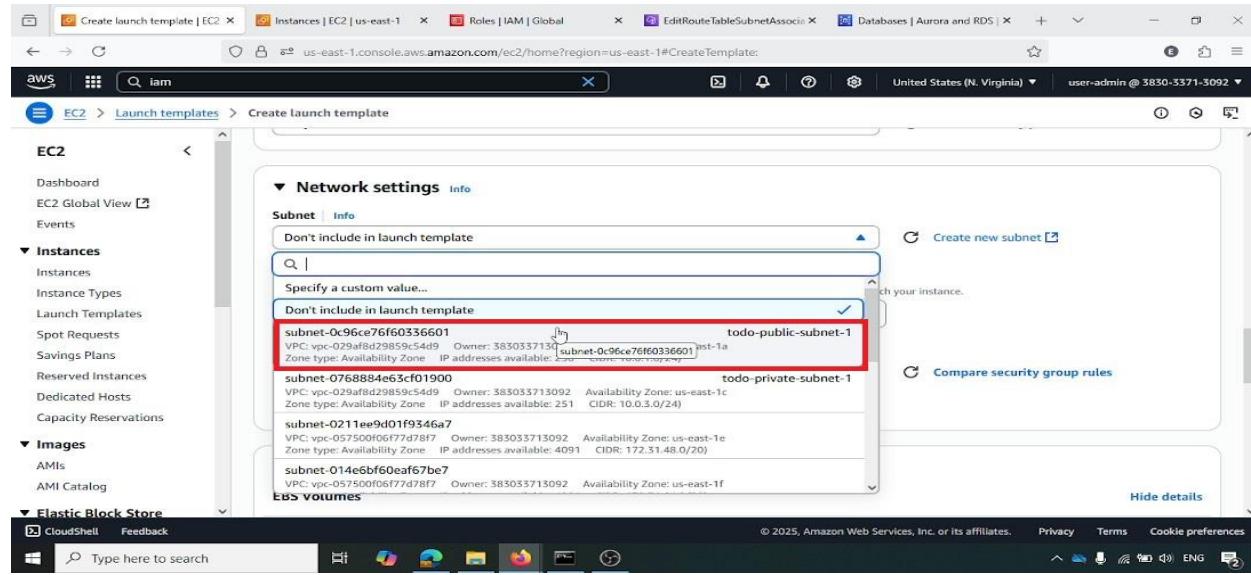
Firewall (security group)

Storage (volumes)

**Create launch template**

## Selecting Subnet for Launch Template

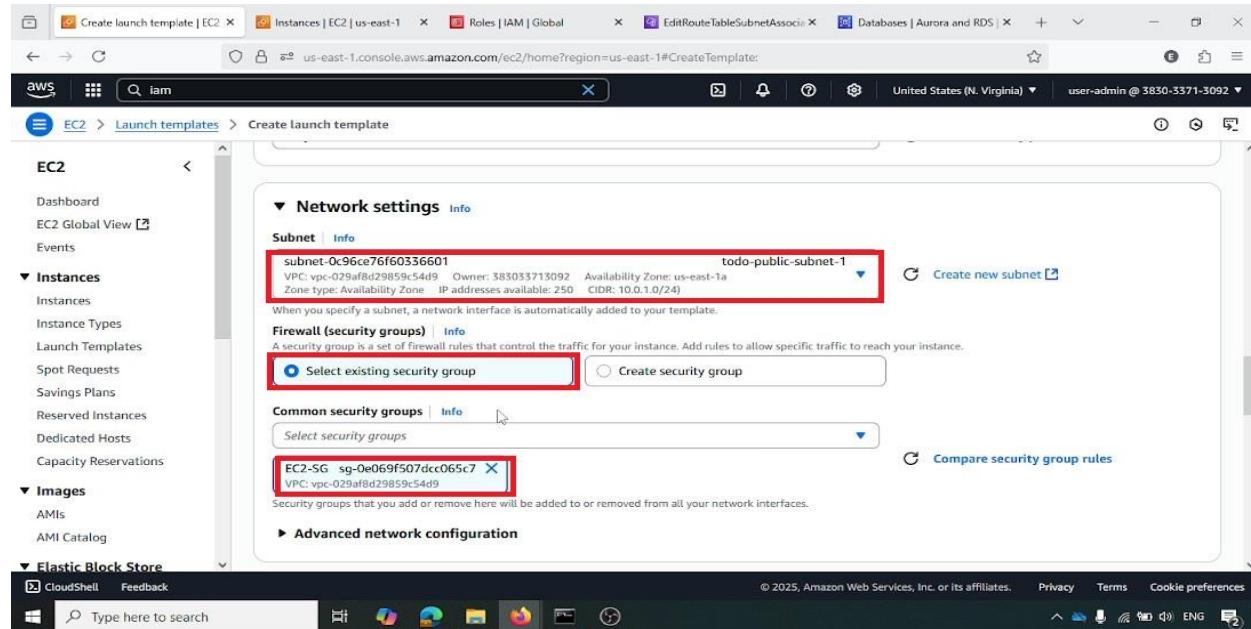
The todo-public-subnet-1 (subnet-0c96ce76f60336601) is highlighted as the selected subnet for the launch template, specifying where instances launched from this template will reside



The screenshot shows the AWS EC2 console with the 'Create launch template' wizard open. In the 'Network settings' step, the 'Subnet' dropdown is expanded, showing a list of available subnets. The subnet 'todo-public-subnet-1' (subnet-0c96ce76f60336601) is selected and highlighted with a red box. Other subnets listed include 'todo-private-subnet-1', 'subnet-0211ee9d01f9346a7', and 'subnet-014e6bf60eaef67be7'. The 'todo-public-subnet-1' entry shows details such as VPC ID, owner, availability zone, and CIDR range.

## 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



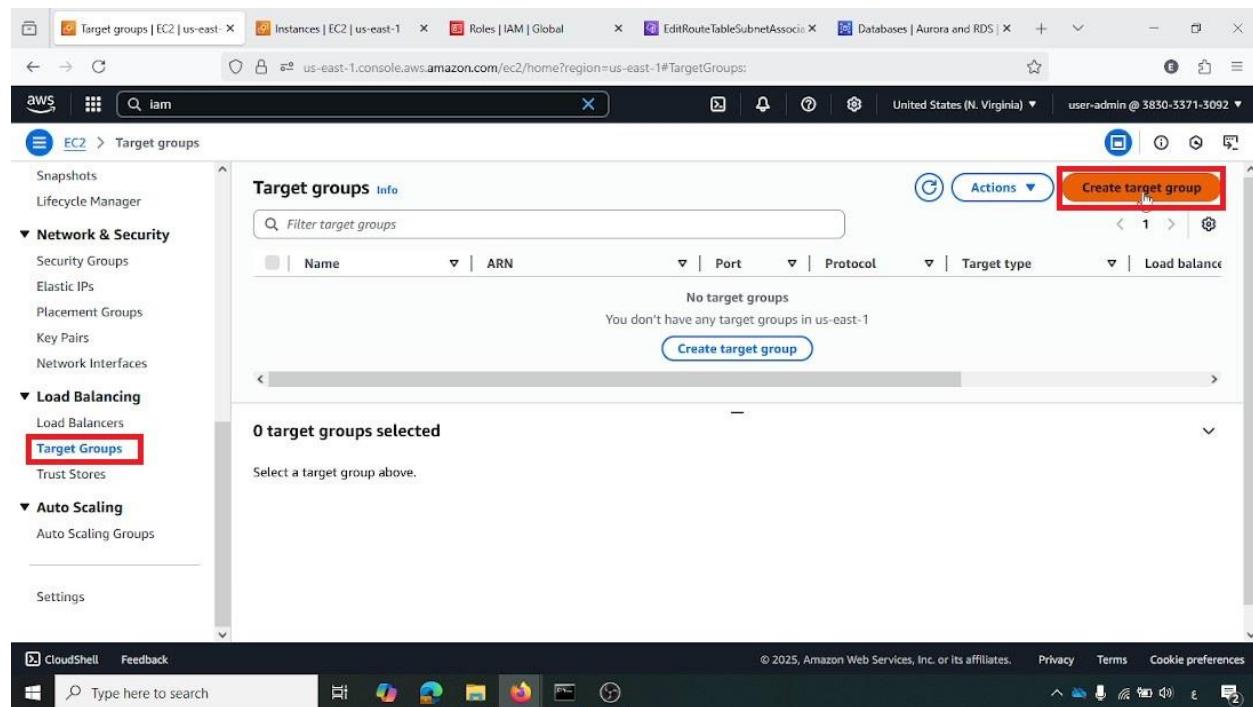
The screenshot shows the 'Network settings' step of the 'Create launch template' wizard. The 'Subnet' dropdown is set to 'todo-public-subnet-1'. In the 'Firewall (security groups)' section, the 'Select existing security group' radio button is selected, and the security group 'EC2-SG sg-0e069f507dcc065c7' is highlighted with a red box. The 'Common security groups' dropdown shows no other groups selected. The 'Advanced network configuration' section is partially visible at the bottom.

## 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

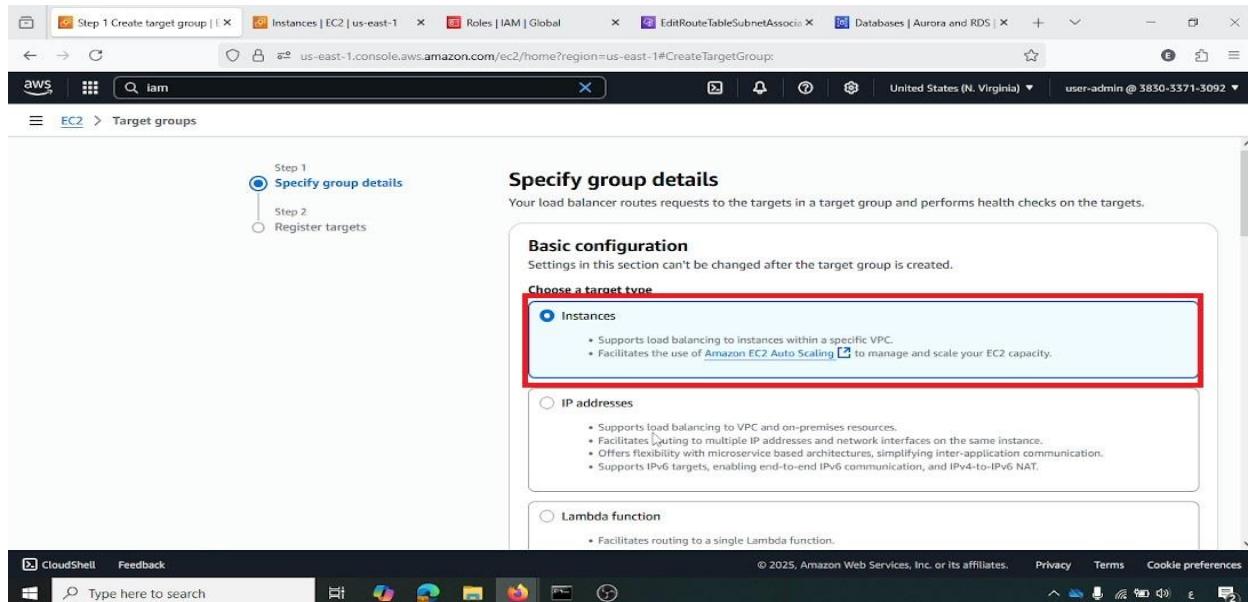
From the AWS EC2 console, under "Target Groups," the "Create target group" button is highlighted, beginning the process of defining a group of targets for the load balancer



The screenshot shows the AWS EC2 Target Groups page. On the left, there's a navigation sidebar with sections like Snapshots, Lifecycle Manager, Network & Security (Security Groups, Elastic IPs, Placement Groups, Key Pairs, Network Interfaces), Load Balancing (Load Balancers, **Target Groups**), Auto Scaling (Auto Scaling Groups), and Settings. The 'Target Groups' link in the Load Balancing section is highlighted with a red box. The main content area is titled 'Target groups' and shows a table with columns: Name, ARN, Port, Protocol, Target type, and Load balance. A message at the top says 'No target groups' and 'You don't have any target groups in us-east-1'. Below the table is a blue 'Create target group' button. At the bottom of the page, there's a search bar, CloudShell, Feedback, and a footer with links for Privacy, Terms, and Cookie preferences.

## Choosing Target Type for Target Group

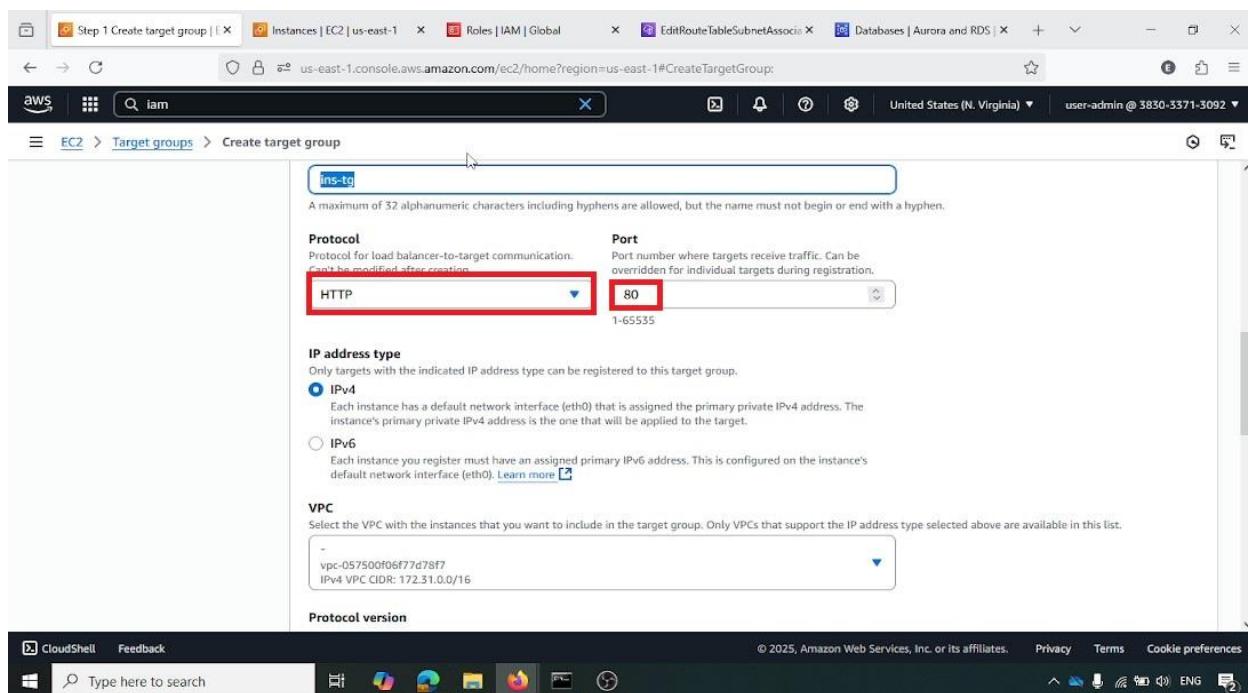
In the "Create target group" wizard, "Instances" is selected as the target type. This specifies that the target group will register EC2 instances as its targets



The screenshot shows the AWS Lambda console with the URL [us-east-1.console.aws.amazon.com/lambda/home?region=us-east-1#CreateTargetGroup](https://us-east-1.console.aws.amazon.com/lambda/home?region=us-east-1#CreateTargetGroup). The page is titled "Step 1 Specify group details". It includes a navigation bar with tabs like Step 1, Step 2, and Register targets. The main content area is titled "Specify group details" with the sub-section "Basic configuration". A red box highlights the "Choose a target type" dropdown, which has "Instances" selected. The "Instances" option is described as supporting load balancing to instances within a specific VPC and facilitating the use of Amazon EC2 Auto Scaling. Other options like "IP addresses" and "Lambda function" are also listed.

## Configuring Target Group Protocol and Port

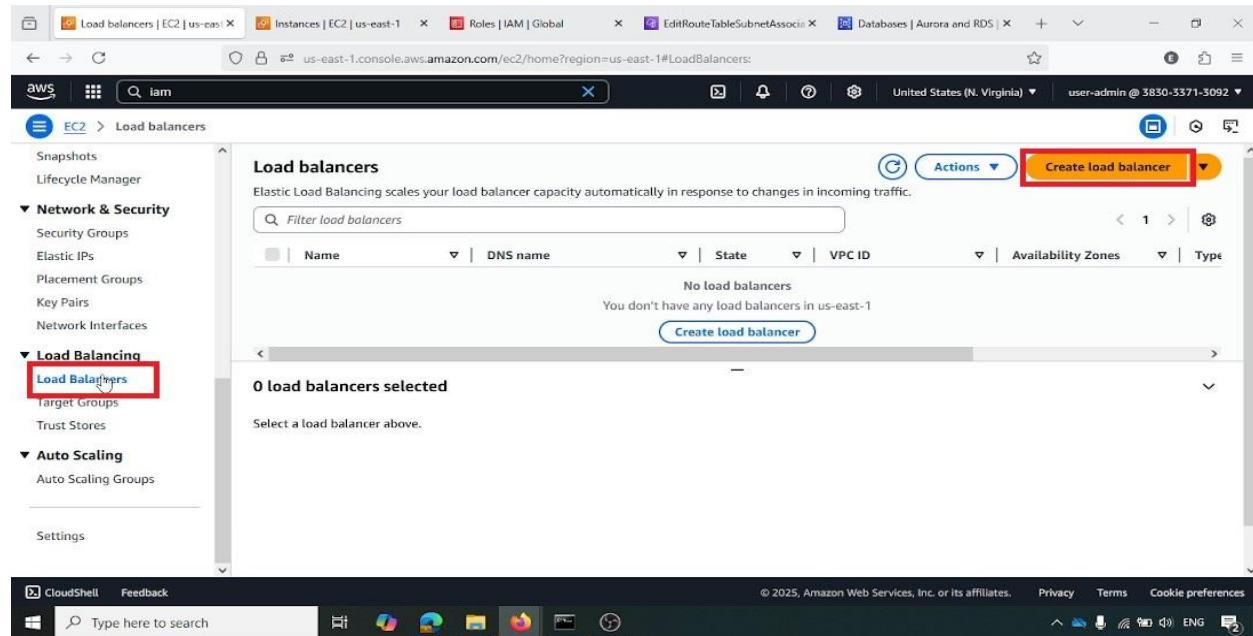
The target group is configured to use "HTTP" protocol on "Port 80." This defines how the load balancer will communicate with the instances in this group



The screenshot shows the AWS Lambda console with the URL [us-east-1.console.aws.amazon.com/lambda/home?region=us-east-1#CreateTargetGroup](https://us-east-1.console.aws.amazon.com/lambda/home?region=us-east-1#CreateTargetGroup). The page is titled "Step 1 Create target group". It includes a navigation bar with tabs like Step 1, Step 2, and Register targets. The main content area is titled "Create target group". A red box highlights the "Protocol" dropdown, which is set to "HTTP". Another red box highlights the "Port" input field, which is set to "80". The "IP address type" section shows "IPv4" selected. The "VPC" section shows a dropdown menu with one item: "vpc-057500f06f77d78f7 IPv4 VPC CIDR: 172.31.0.0/16". The bottom of the screen shows the Windows taskbar.

## Initiating Load Balancer Creation

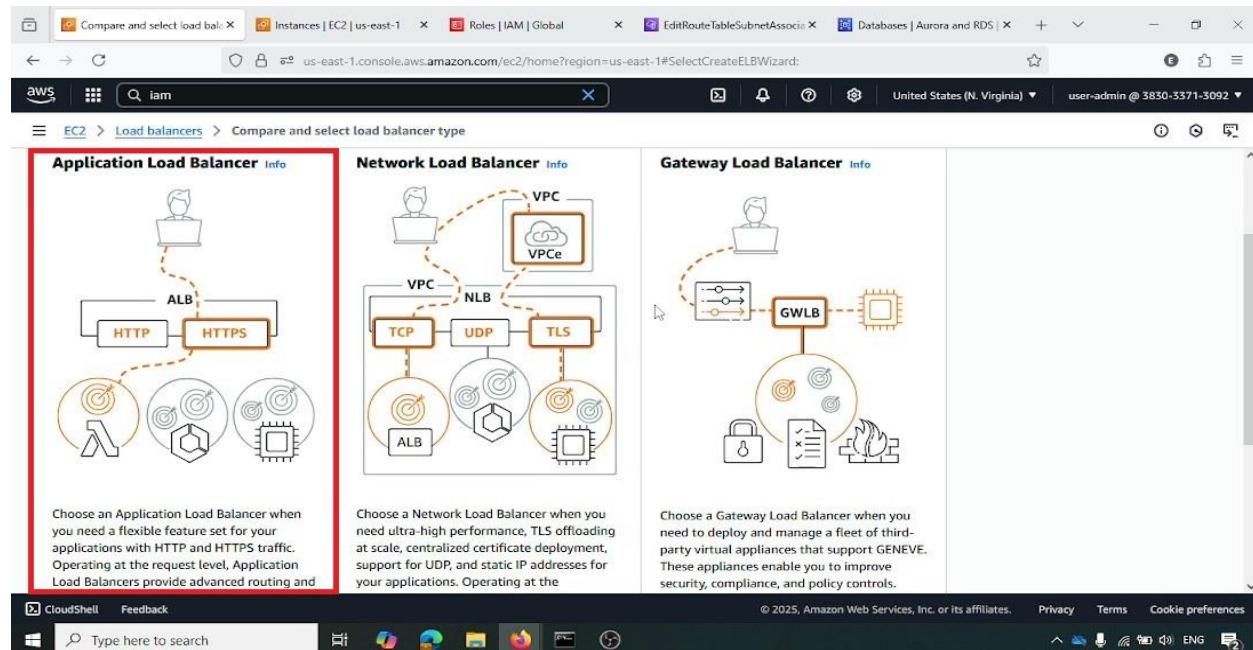
From the AWS EC2 console, under "Load Balancers," the "Create load balancer" button is highlighted, starting the process of setting up a new load balancer



The screenshot shows the AWS EC2 console with the 'Load balancers' section selected. The left sidebar includes 'Network & Security' (Snapshots, Lifecycle Manager), 'Load Balancing' (Load Balancers, Target Groups, Trust Stores), and 'Auto Scaling' (Auto Scaling Groups). The main area displays a table titled 'Load balancers' with a single row: 'No load balancers'. A message says 'You don't have any load balancers in us-east-1'. At the top right, there is a 'Create load balancer' button, which is highlighted with a red box. The browser address bar shows 'us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LoadBalancers'.

## Choosing Application Load Balancer Type

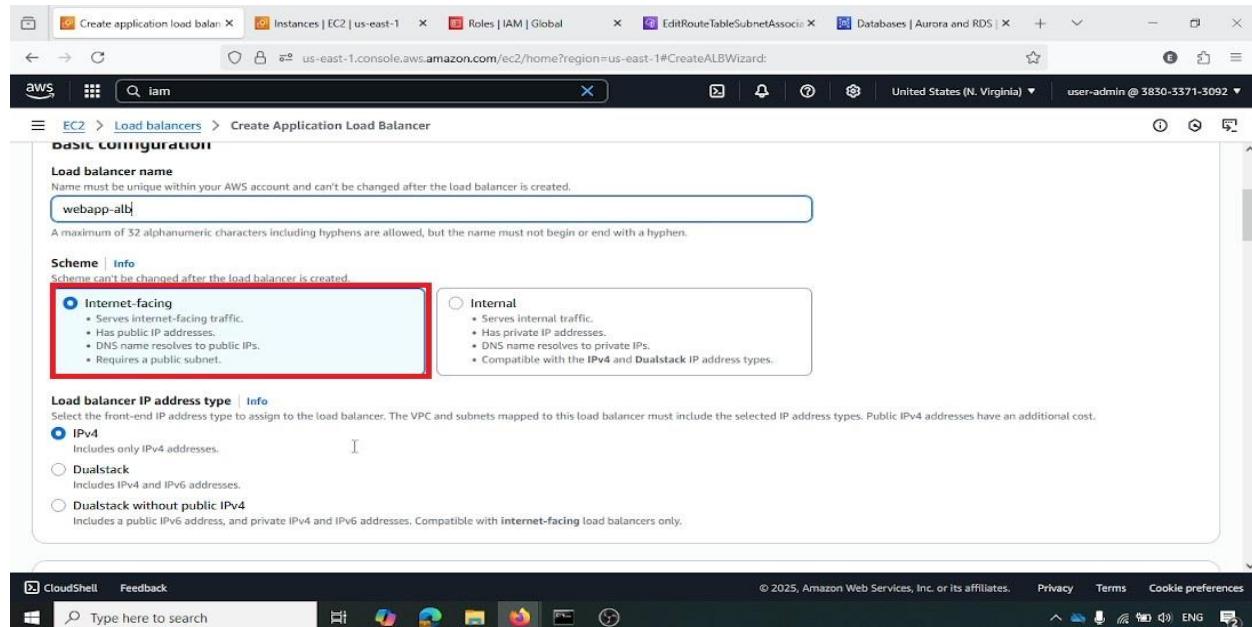
"Application Load Balancer" is highlighted and selected. This type is chosen for its ability to handle HTTP/HTTPS traffic and provide advanced routing features



The screenshot shows the 'Compare and select load balancer type' wizard. It displays three options: 'Application Load Balancer', 'Network Load Balancer', and 'Gateway Load Balancer'. The 'Application Load Balancer' section is highlighted with a red box. It contains a diagram showing traffic from a client through an ALB (Application Load Balancer) to three Lambda functions. Below the diagram, text reads: '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...'. The other two sections show similar diagrams and descriptions for Network and Gateway Load Balancers.

## 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



**Basic configuration**

**Load balancer name**  
Name must be unique within your AWS account and can't be changed after the load balancer is created.  
**webapp-alb**

A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen.

**Scheme | Info**  
Scheme can't be changed after the load balancer is created.

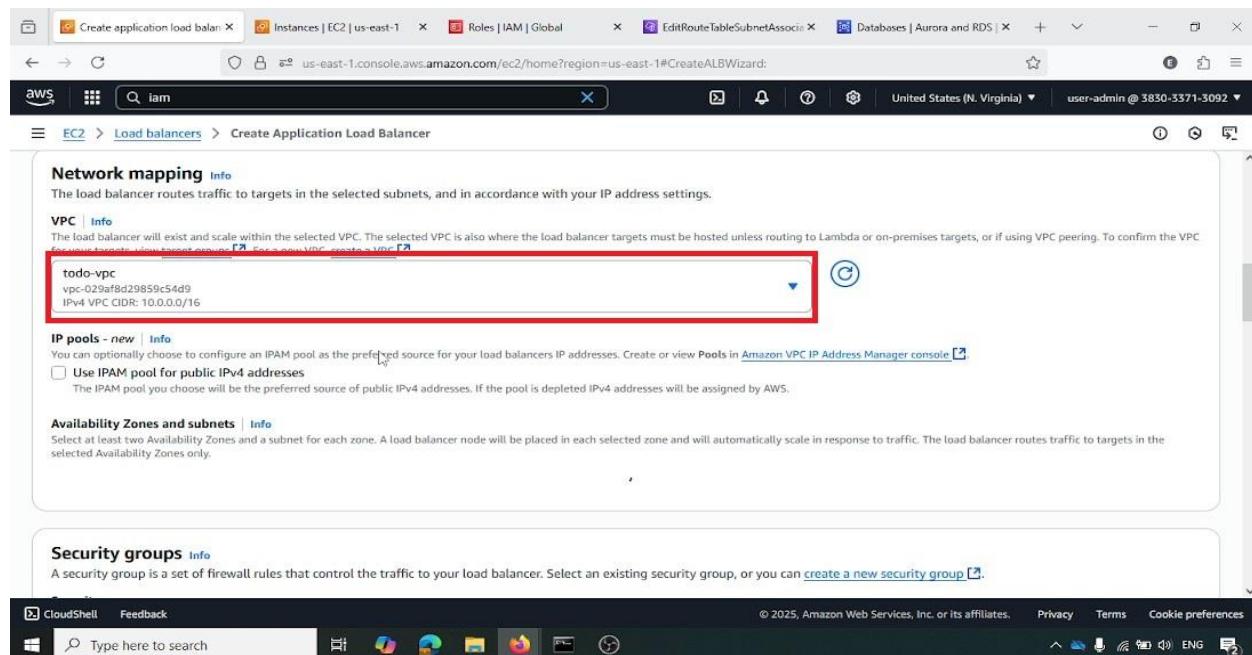
- Internet-Facing**
  - Serves internet-facing traffic.
  - Has public IP addresses.
  - DNS name resolves to public IPs.
  - Requires a public subnet.
- Internal**
  - Serves internal traffic.
  - Has private IP addresses.
  - DNS name resolves to private IPs.
  - Compatible with the IPv4 and Dualstack IP address types.

**Load balancer IP address type | Info**  
Select the front-end IP address type to assign to the load balancer. The VPC and subnets mapped to this load balancer must include the selected IP address types. Public IPv4 addresses have an additional cost.

- IPv4**  
Includes only IPv4 addresses.
- Dualstack**  
Includes IPv4 and IPv6 addresses.
- Dualstack without public IPv4**  
Includes a public IPv6 address, and private IPv4 and IPv6 addresses. Compatible with internet-facing load balancers only.

## 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



**Network mapping | Info**  
The load balancer routes traffic to targets in the selected subnets, and in accordance with your IP address settings.

**VPC | Info**  
The load balancer will exist and scale within the selected VPC. The selected VPC is also where the load balancer targets must be hosted unless routing to Lambda or on-premises targets, or if using VPC peering. To confirm the VPC selection, click [View details](#).

**todo-vpc**  
vpc-029af8d29859c54d9  
IPv4 VPC CIDR: 10.0.0.0/16

**IP pools - new | Info**  
You can optionally choose to configure an IPAM pool as the preferred source for your load balancer's IP addresses. Create or view Pools in Amazon VPC IP Address Manager console.

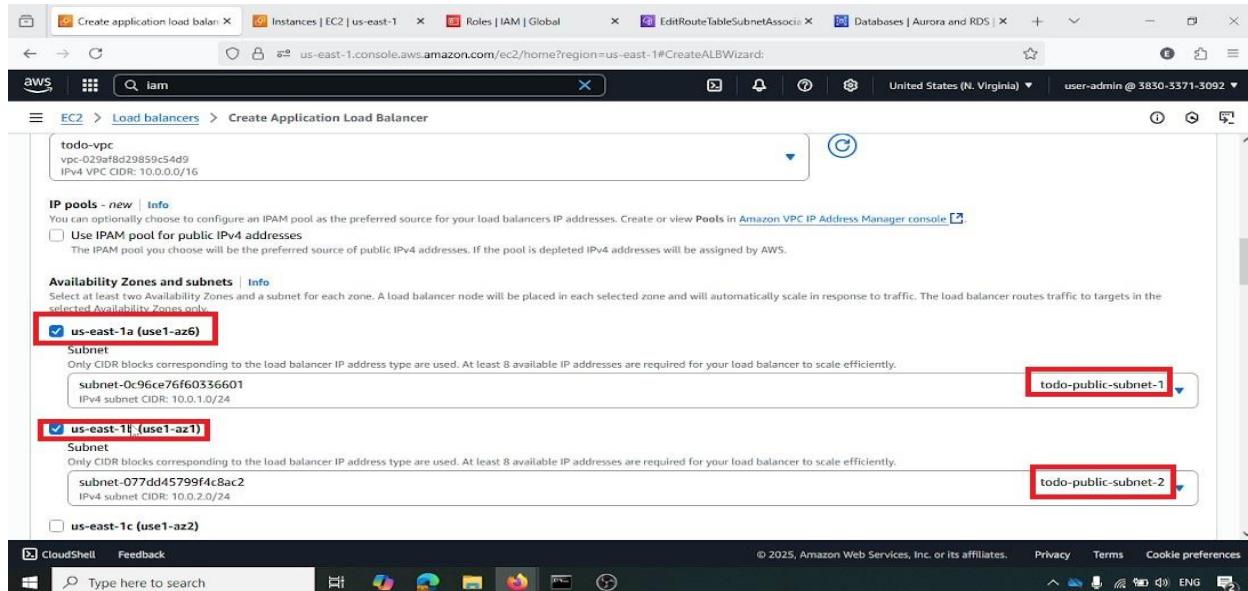
**Use IPAM pool for public IPv4 addresses**  
The IPAM pool you choose will be the preferred source of public IPv4 addresses. If the pool is depleted IPv4 addresses will be assigned by AWS.

**Availability Zones and subnets | Info**  
Select at least two Availability Zones and a subnet for each zone. A load balancer node will be placed in each selected zone and will automatically scale in response to traffic. The load balancer routes traffic to targets in the selected Availability Zones only.

**Security groups | Info**  
A security group is a set of firewall rules that control the traffic to your load balancer. Select an existing security group, or you can [create a new security group](#).

## Selecting Availability Zones and Subnets for Load Balancer

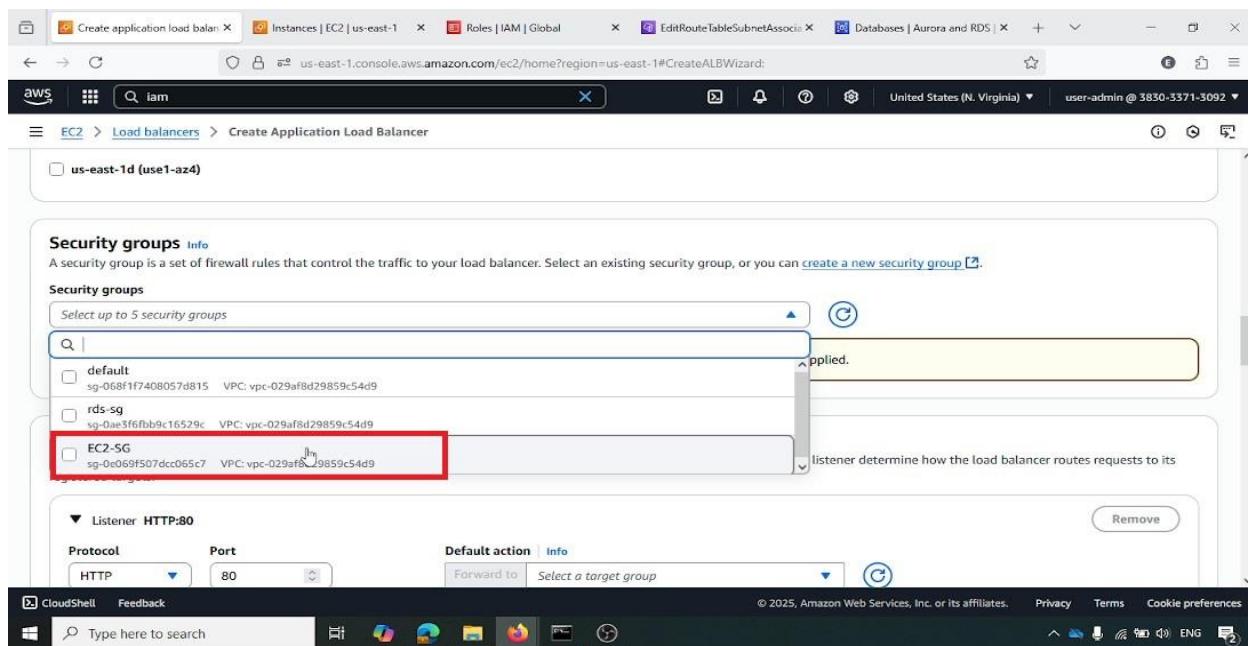
Two public subnets, us-east-1a (todo-public-subnet-1) and us-east-1b (todo-public-subnet-2), are selected. This deploys the load balancer across multiple Availability Zones for improved fault tolerance



The screenshot shows the 'Create Application Load Balancer' wizard step. In the 'Availability Zones and subnets' section, 'us-east-1a (use1-az6)' is selected and highlighted with a red box. Two subnets are listed: 'todo-public-subnet-1' (selected) and 'todo-public-subnet-2'. 'us-east-1b (use1-az1)' and 'us-east-1c (use1-az2)' are also shown but not selected.

## Selecting Security Groups for Load Balancer

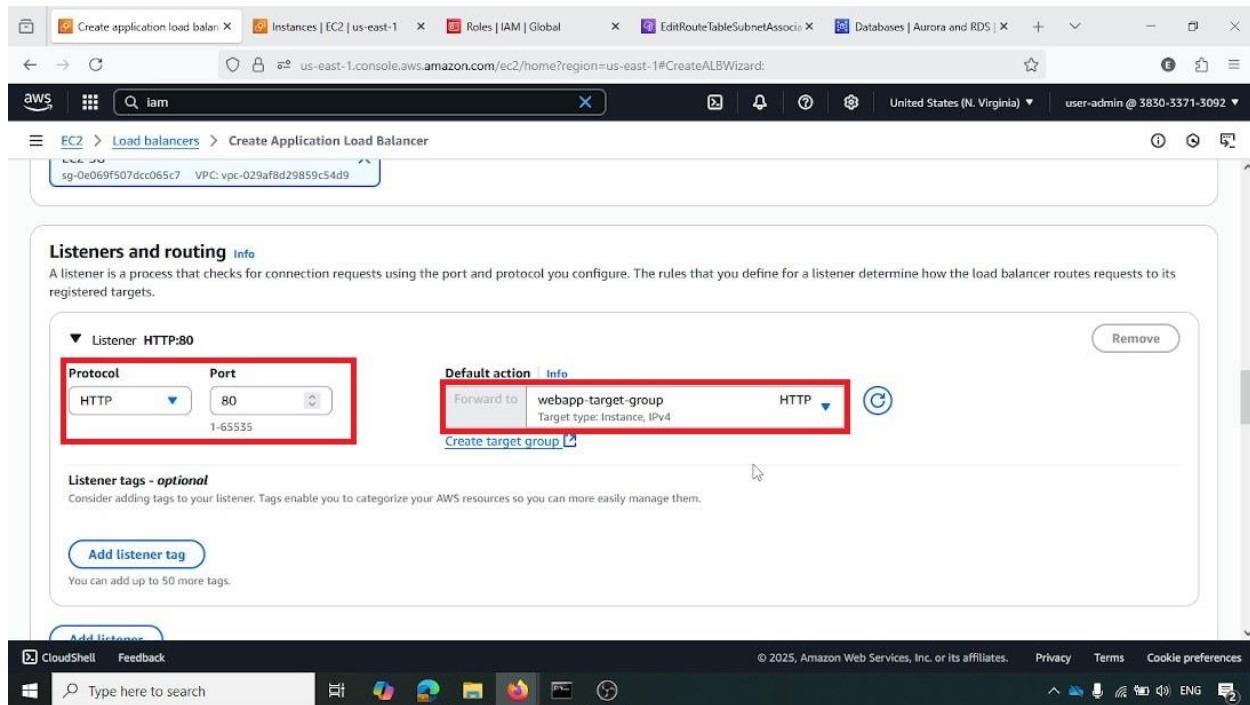
The EC2-SG security group (sg-0e069f507c0c065c7) is highlighted and selected for the load balancer. This security group governs the network access to the ALB



The screenshot shows the 'Create Application Load Balancer' wizard step. In the 'Security groups' section, 'EC2-SG' is selected and highlighted with a red box. Other security groups listed are 'default' and 'rds-sg'.

## 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



The screenshot shows the AWS CloudFormation Create Application Load Balancer wizard. The 'Listeners and routing' section is displayed, with two specific configurations highlighted by red boxes:

- Protocol:** HTTP
- Port:** 80
- Default action:** Forward to webapp-target-group (Target type: Instance, IPv4)

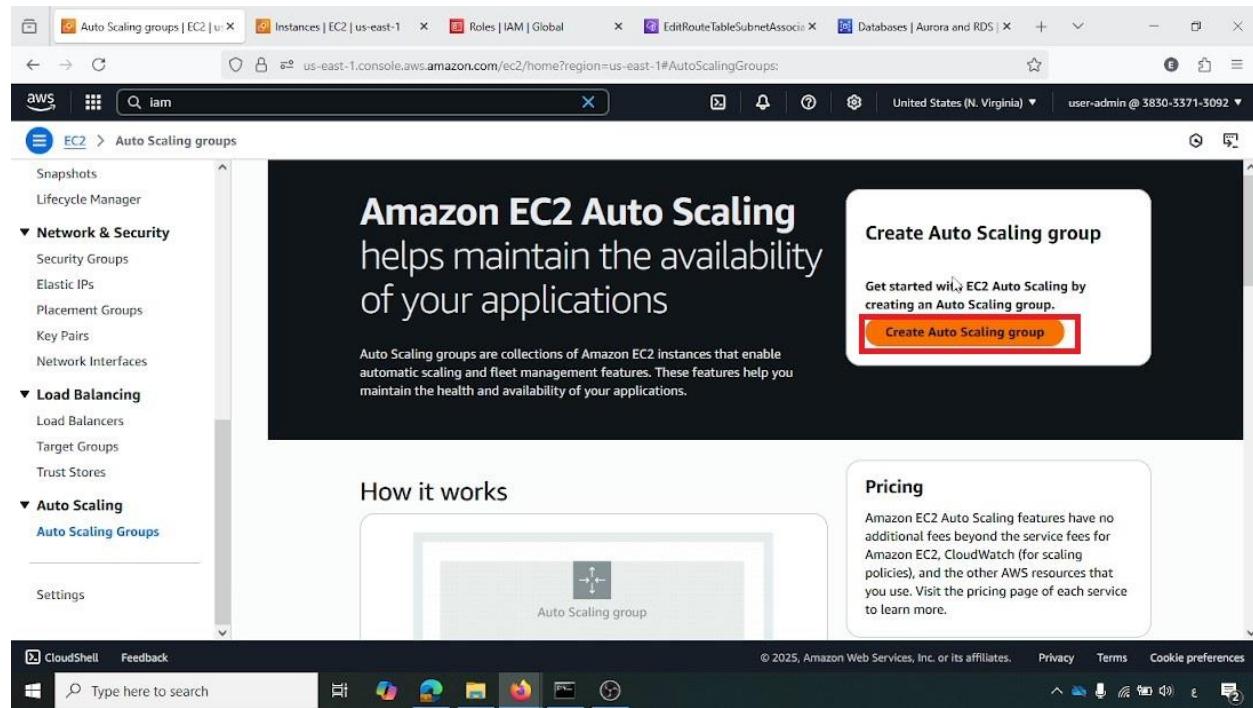
Below these configurations, there is an optional section for adding listener tags.

## 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.

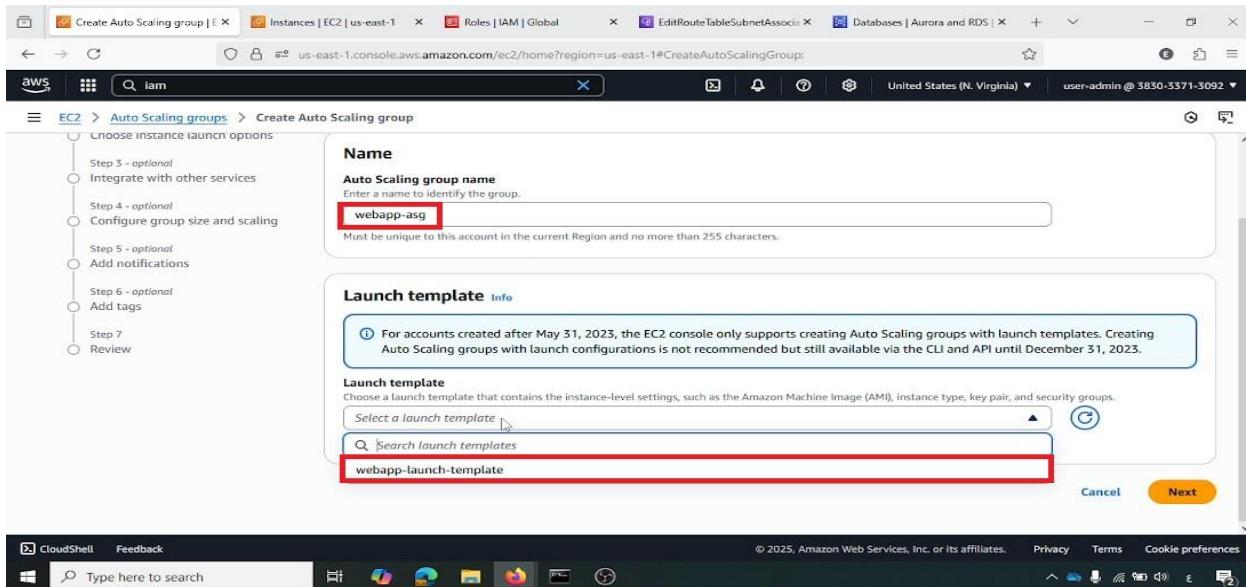
### Initiating Auto Scaling Group Creation

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

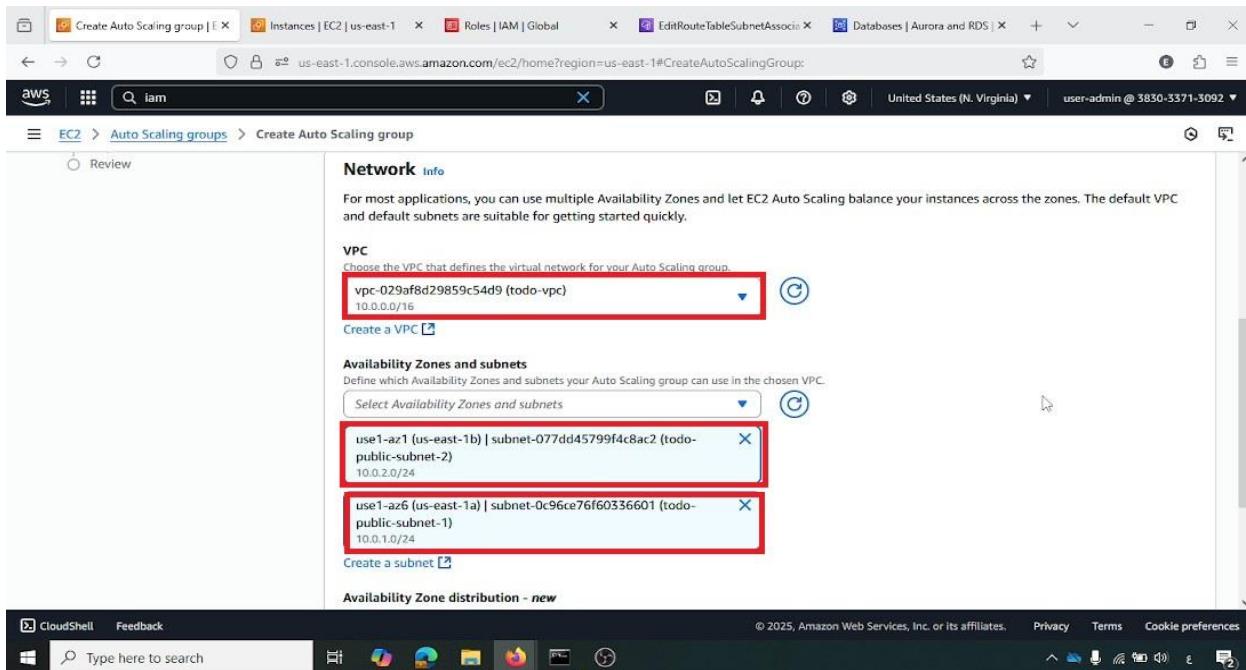
The Auto Scaling group is named webapp-asg, and the webapp-launch-template is selected. This links the Auto Scaling group to the predefined instance configuration



The screenshot shows the 'Create Auto Scaling group' wizard in progress. The current step is 'Choose instance launch options'. In the 'Name' section, the 'Auto Scaling group name' is set to 'webapp-asg'. In the 'Launch template' section, a search bar shows 'Search launch templates' with the result 'webapp-launch-template' highlighted.

## Selecting VPC and Subnets for Auto Scaling Group

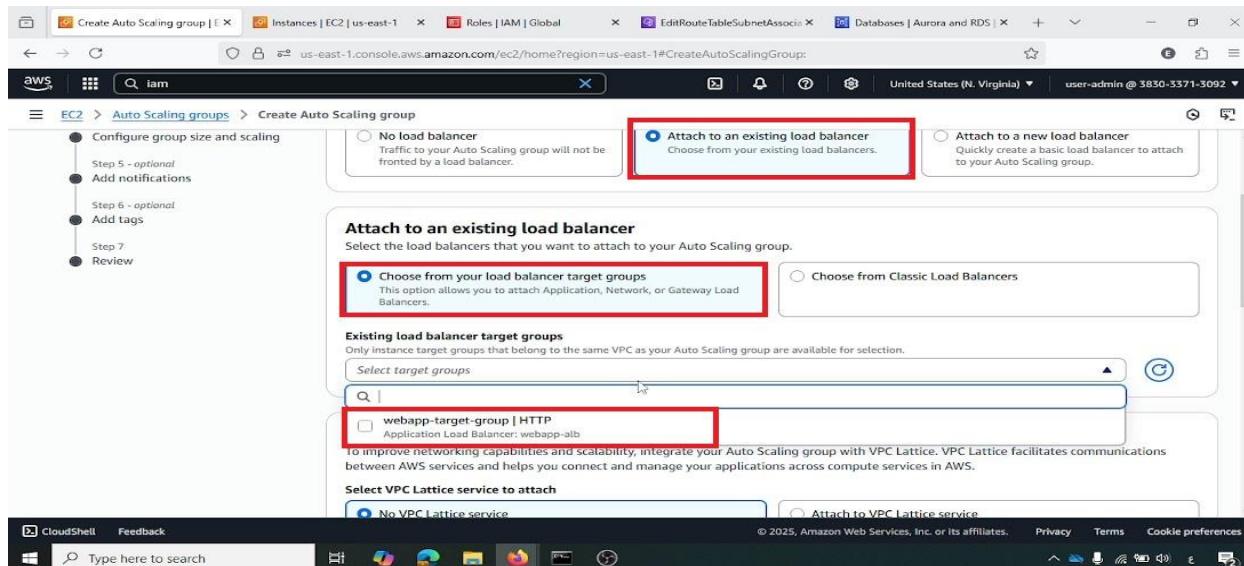
The todo-vpc and the two public subnets (use1-az1 and use1-az6) are selected for the Auto Scaling group. This ensures instances are launched in the correct network and across multiple Availability Zones



The screenshot shows the 'Create Auto Scaling group' wizard in progress. The current step is 'Network'. In the 'VPC' section, the selected VPC is 'vpc-0290fd29859c54d9 (todo-vpc)'. In the 'Availability Zones and subnets' section, two subnets are selected: 'use1-az1 (us-east-1b) | subnet-077dd45799f4c8ac2 (todo-public-subnet-2)' and 'use1-az6 (us-east-1a) | subnet-0c96ce76f60336601 (todo-public-subnet-1)'. Both subnets are highlighted with red boxes.

## Attaching to Existing Load Balancer

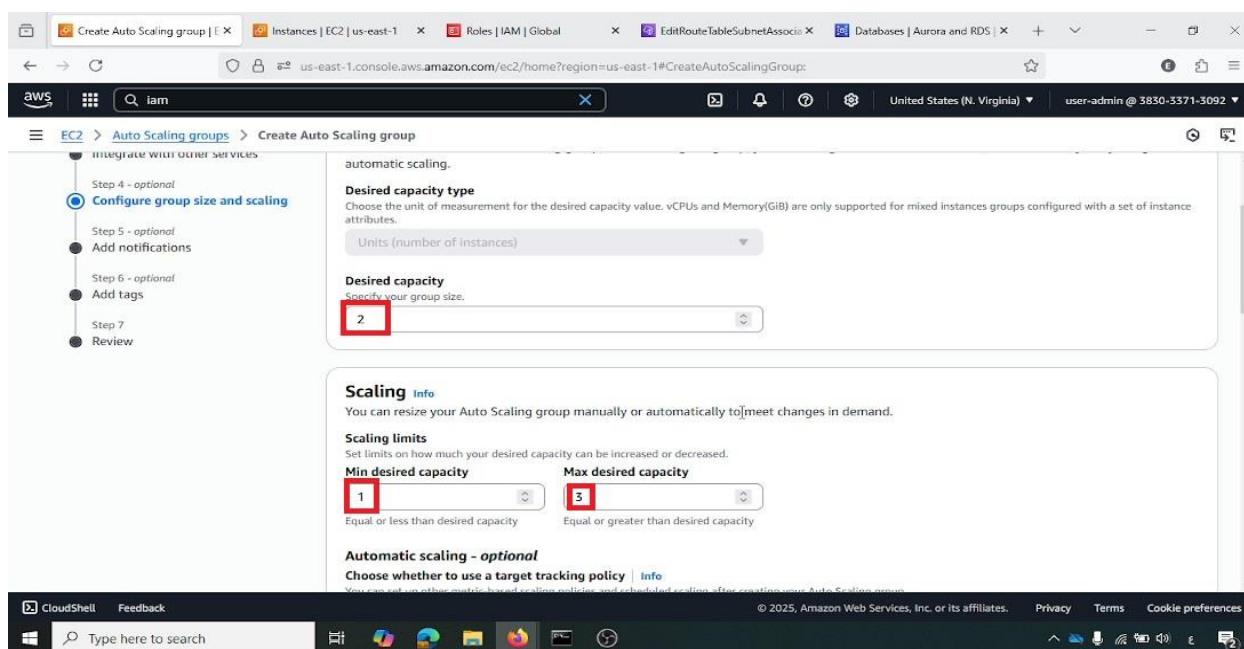
The option "Attach to an existing load balancer" is selected, and the webapp-target-group is chosen. This integrates the Auto Scaling group with the Application Load Balancer, allowing traffic to be distributed to its instances



The screenshot shows the AWS Create Auto Scaling group wizard at Step 4: Configure group size and scaling. The 'Attach to an existing load balancer' option is selected and highlighted with a red box. Below it, the 'Choose from your load balancer target groups' section is also highlighted with a red box, showing the 'webapp-target-group | HTTP' option selected.

## Setting Desired Capacity and Scaling Limits

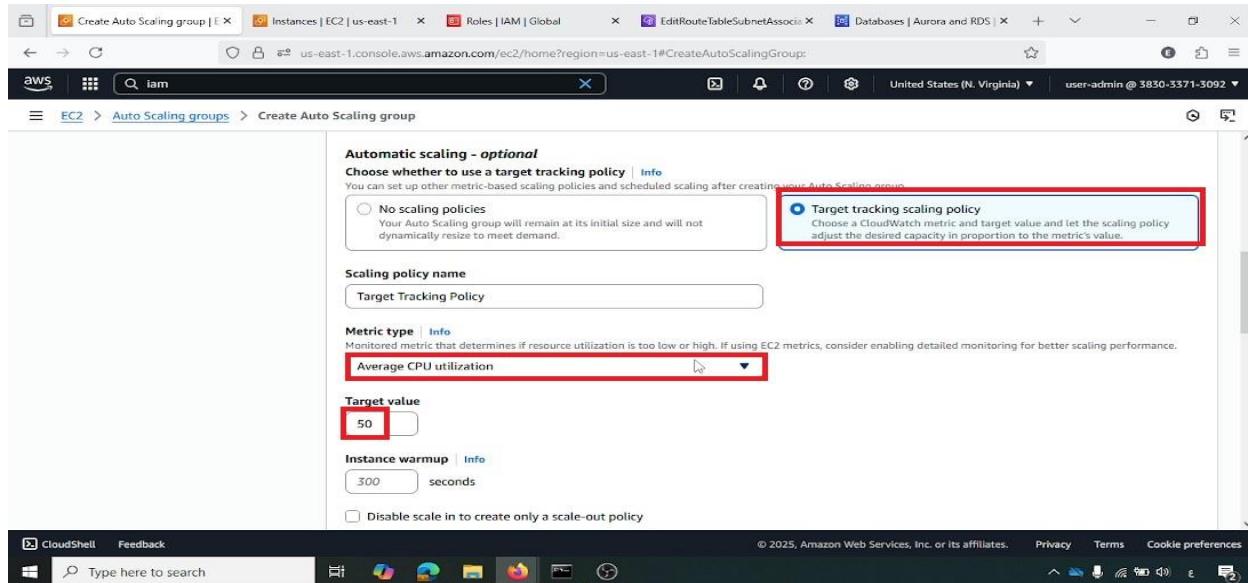
The "Desired capacity" is set to "2" instances, with a "Min desired capacity" of "1" and a "Max desired capacity" of "3." These settings define the initial and boundary scaling behavior of the group



The screenshot shows the AWS Create Auto Scaling group wizard at Step 4: Configure group size and scaling. The 'Desired capacity' field is set to '2' and highlighted with a red box. The 'Min desired capacity' field is set to '1' and highlighted with a red box. The 'Max desired capacity' field is set to '3' and highlighted with a red box.

## 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%



**Automatic scaling - optional**

Choose whether to use a target tracking policy | [Info](#)  
You can set up other metric-based scaling policies and scheduled scaling after creating your Auto Scaling group.

No scaling policies  
Your Auto Scaling group will remain at its initial size and will not dynamically resize to meet demand.

Target tracking scaling policy  
Choose a CloudWatch metric and target value and let the scaling policy adjust the desired capacity in proportion to the metric's value.

**Scaling policy name**

**Metric type** | [Info](#)  
Monitored metric that determines if resource utilization is too low or high. If using EC2 metrics, consider enabling detailed monitoring for better scaling performance.

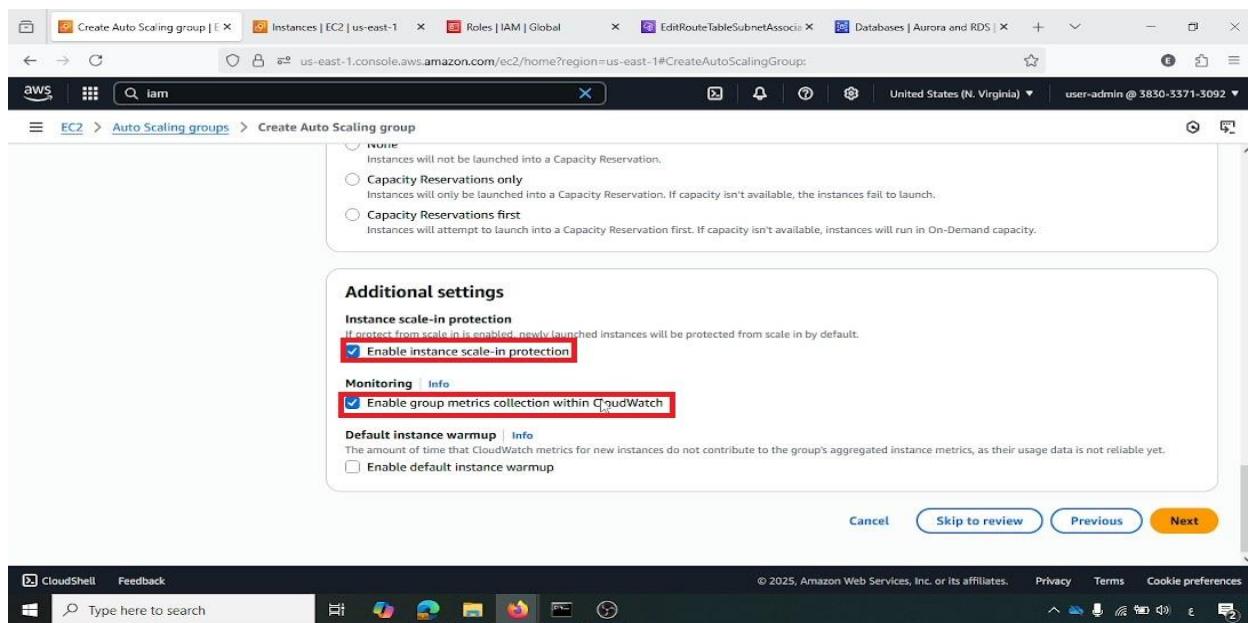
**Target value**

**Instance warmup** | [Info](#)  
 seconds

Disable scale in to create only a scale-out policy

## 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



**Instance scale-in protection**  
If protect from scale in is enabled, newly launched instances will be protected from scale in by default.  
 Enable instance scale-in protection

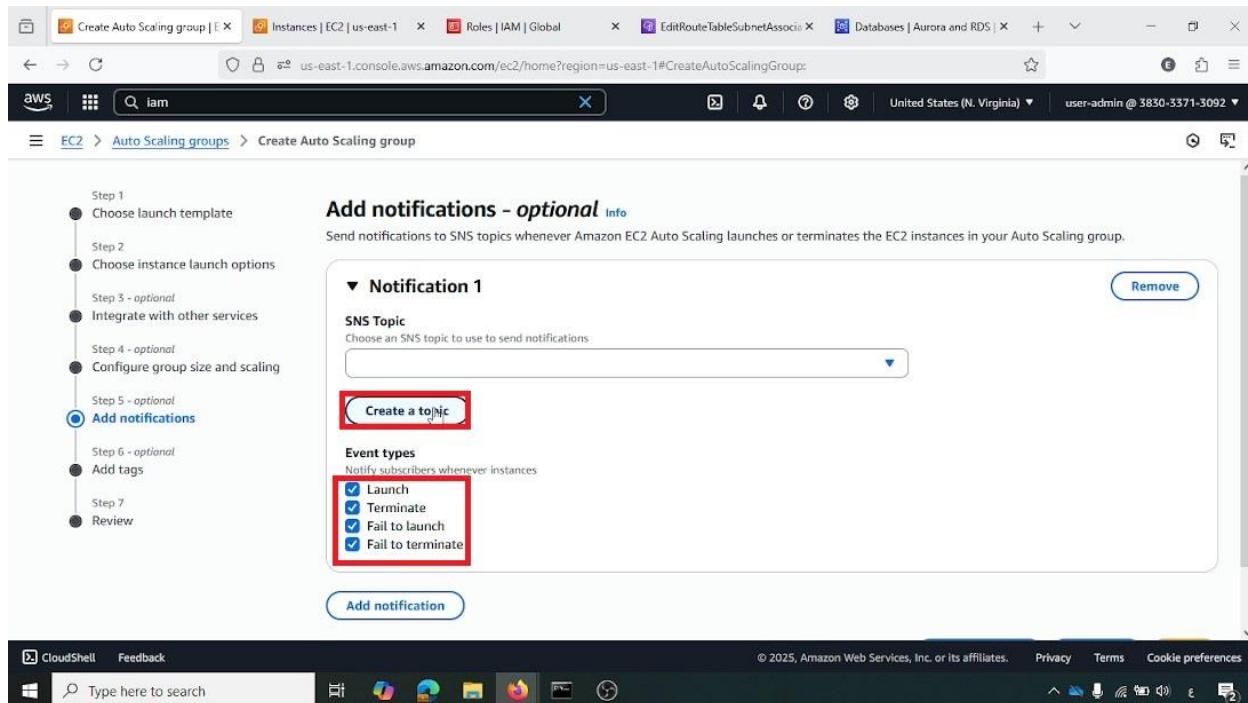
**Monitoring** | [Info](#)  
 Enable group metrics collection within CloudWatch

**Default instance warmup** | [Info](#)  
The amount of time that CloudWatch metrics for new instances do not contribute to the group's aggregated instance metrics, as their usage data is not reliable yet.  
 Enable default instance warmup

[Cancel](#) [Skip to review](#) [Previous](#) [Next](#)

## 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



The screenshot shows the AWS EC2 Create Auto Scaling group wizard at Step 5 - optional: Add notifications. The 'Create a topic' button and the selected event types (Launch, Terminate, Fail to launch, Fail to terminate) are highlighted with red boxes.

**Screenshot Description:**

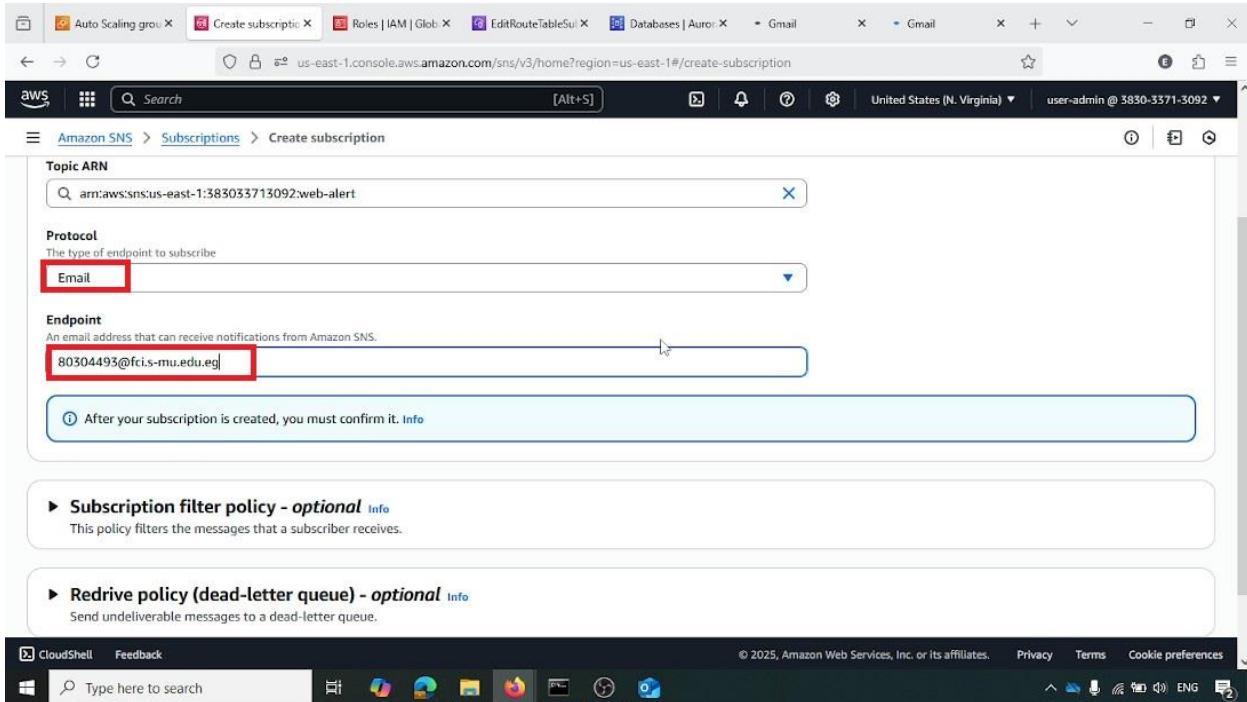
- Left sidebar (Step 5 - optional):**
  - Add notifications (highlighted with a red circle)
  - Step 6 - optional: Add tags
  - Step 7: Review
- Main Content Area:**
  - Add notifications - optional** (Info: Send notifications to SNS topics whenever Amazon EC2 Auto Scaling launches or terminates the EC2 instances in your Auto Scaling group.)
  - Notification 1** (Remove button)
  - SNS Topic:** Choose an SNS topic to use to send notifications. (Dropdown menu)
  - Create a topic** (Button)
  - Event types:** Notify subscribers whenever instances
    - Launch
    - Terminate
    - Fail to launch
    - Fail to terminate
  - Add notification** (Button)

## 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

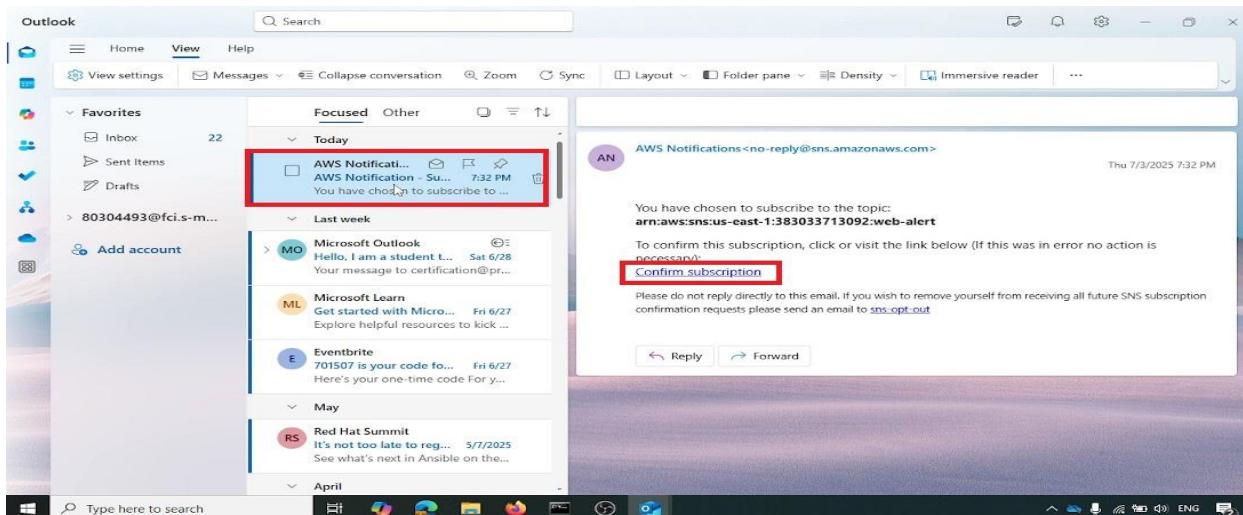
In the AWS SNS console, a new subscription is created with "Protocol" set to "Email" and the "Endpoint" specified as 80304493@fci-s.mu-edu.eg. This configures the email address to receive notifications from the SNS topic



The screenshot shows the 'Create subscription' page in the AWS SNS console. The 'Topic ARN' field contains 'arn:aws:sns:us-east-1:383033713092:web-alert'. The 'Protocol' dropdown is set to 'Email'. The 'Endpoint' field contains '80304493@fci-s.mu-edu.eg'. A note at the bottom says 'After your subscription is created, you must confirm it.' Below the main form, there are sections for 'Subscription filter policy - optional' and 'Redrive policy (dead-letter queue) - optional'. The browser's address bar shows 'us-east-1.console.aws.amazon.com/sns/v3/home?region=us-east-1#/create-subscription'. The status bar at the bottom indicates the user is in United States (N. Virginia) and has a session ID.

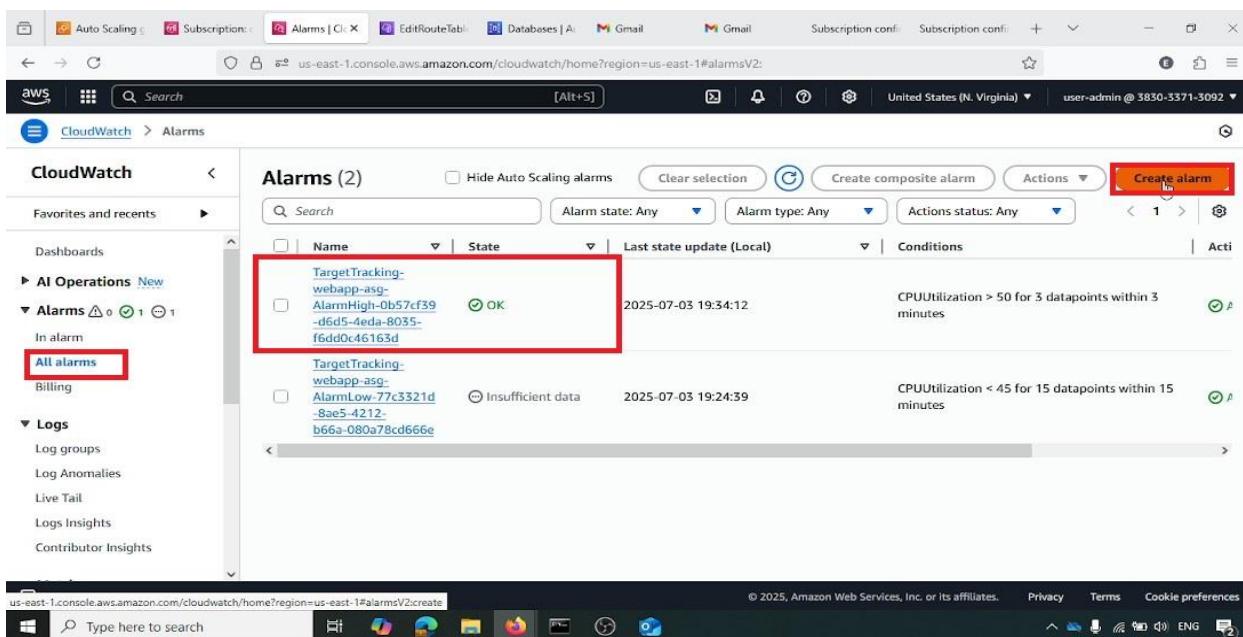
## Confirming SNS Subscription

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

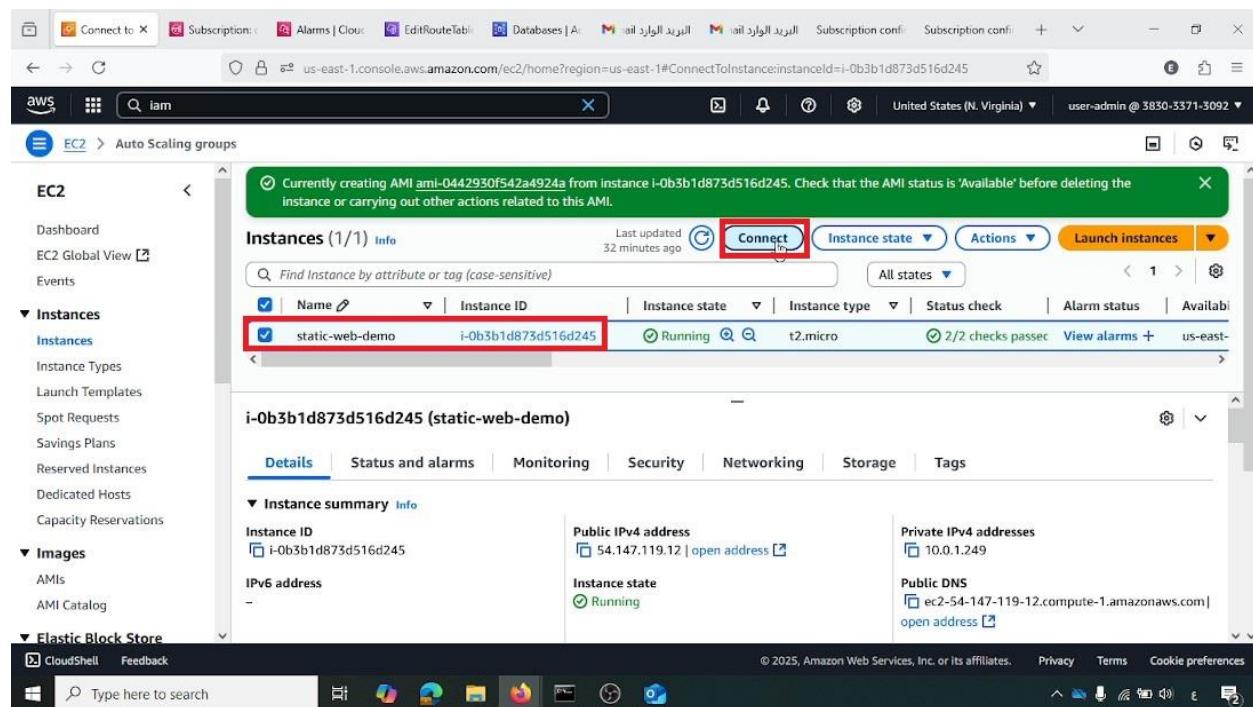


## Step 6: Testing and Validation

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

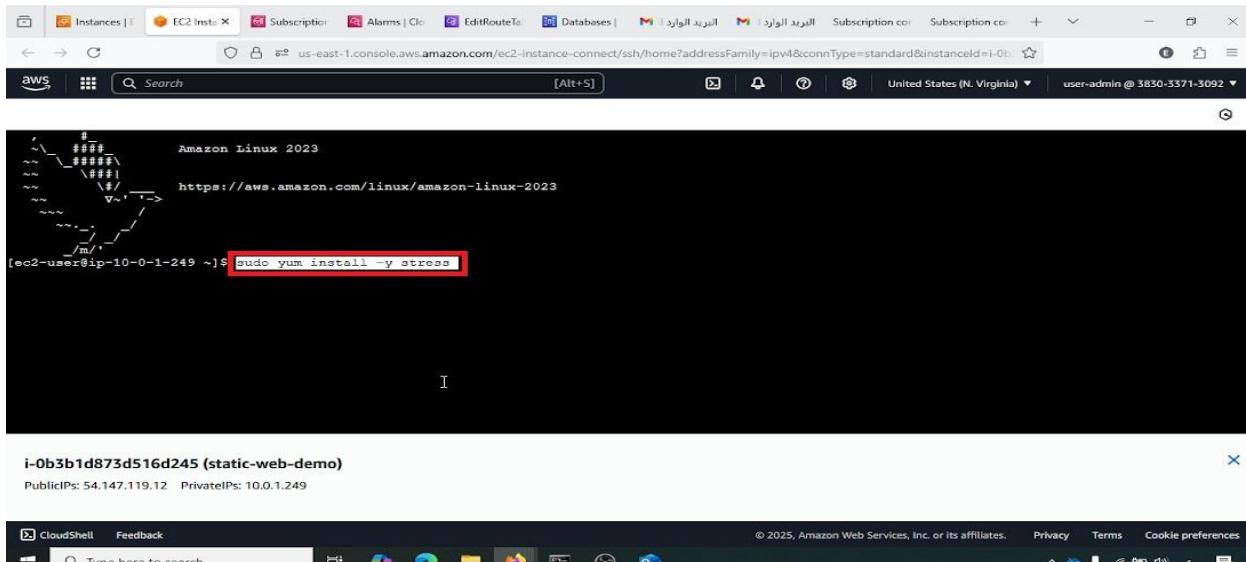
### Connecting to EC2 Instance

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



```

Amazon Linux 2023
https://aws.amazon.com/linux/amazon-linux-2023

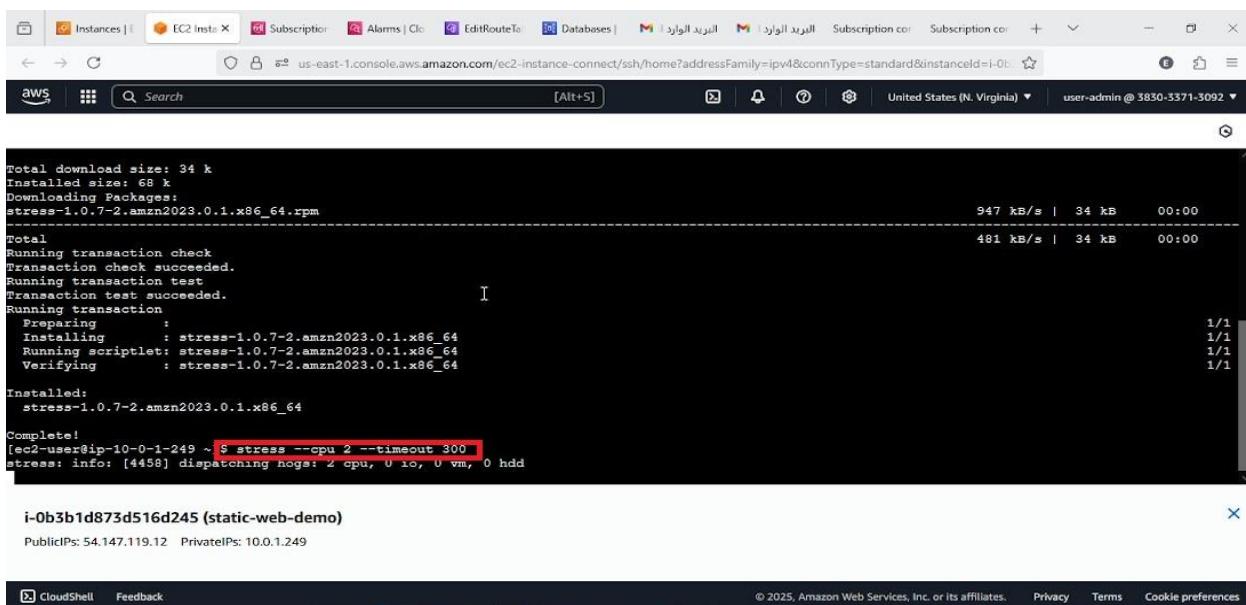
[ec2-user@ip-10-0-1-249 ~] $ sudo yum install -y stress

```

i-0b3b1d873d516d245 (static-web-demo)  
PublicIPs: 54.147.119.12 PrivateIPs: 10.0.1.249

## 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



```

Total download size: 34 k
Installed size: 68 k
Downloading Packages:
stress-1.0.7-2.amzn2023.0.1.x86_64.rpm
Total
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
Preparing : 947 kB/s | 34 kB 00:00
Installing : stress-1.0.7-2.amzn2023.0.1.x86_64 481 kB/s | 34 kB 00:00
Running scriptlet: stress-1.0.7-2.amzn2023.0.1.x86_64
Verifying : stress-1.0.7-2.amzn2023.0.1.x86_64 1/1
1/1
1/1
1/1
1/1

Installed:
stress-1.0.7-2.amzn2023.0.1.x86_64

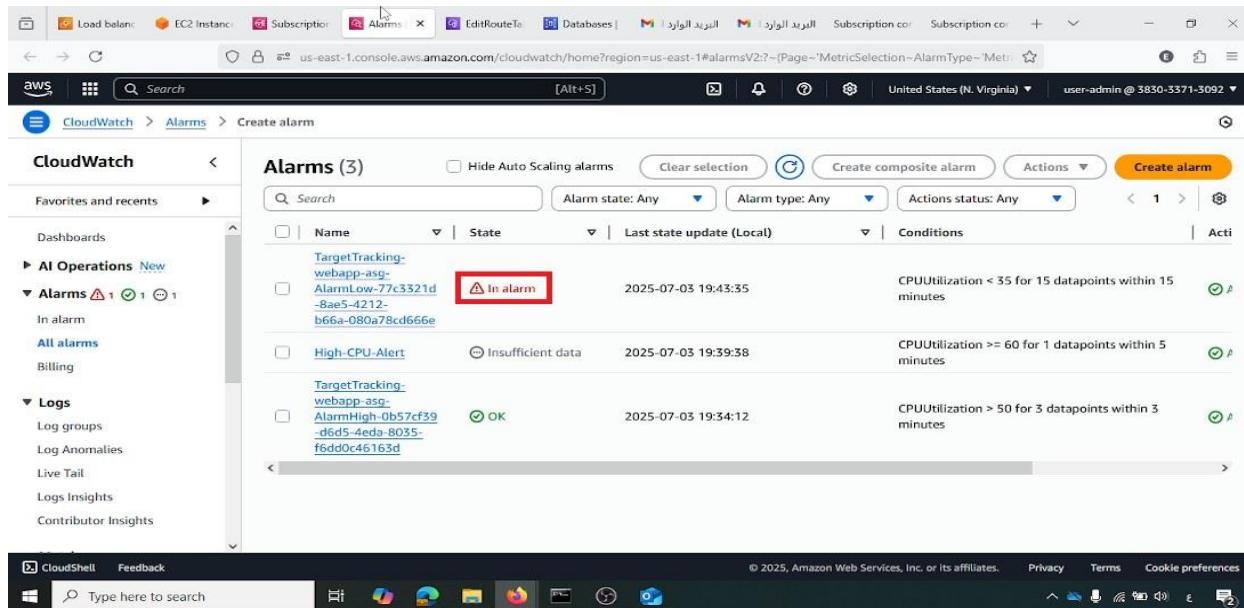
Complete!
[ec2-user@ip-10-0-1-249 ~] $ stress --cpu 2 --timeout 300
stress: info: [4458] dispatching hogs: 2 cpu, 0 io, 0 vm, 0 hdd

```

i-0b3b1d873d516d245 (static-web-demo)  
PublicIPs: 54.147.119.12 PrivateIPs: 10.0.1.249

## 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.

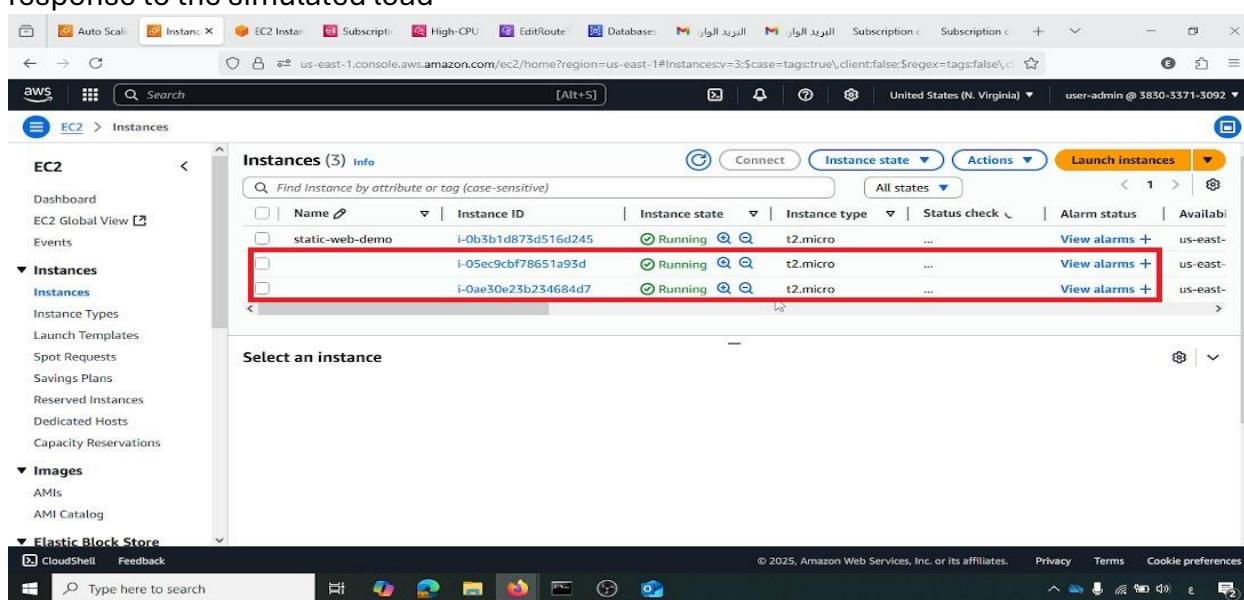


The screenshot shows the AWS CloudWatch Alarms interface. On the left, there's a navigation sidebar with 'CloudWatch' selected. Under 'Alarms', there are three items: 'In alarm' (with 1 item), 'All alarms' (with 1 item), and 'Billing'. The main area displays a table of alarms:

Name	State	Last state update (Local)	Conditions
TargetTracking-webapp-asg-AlarmLow-77c3321d-Bae5-4212-b66a-080a78cd666e	In alarm	2025-07-03 19:43:35	CPUUtilization < 35 for 15 datapoints within 15 minutes
High-CPU-Alert	Insufficient data	2025-07-03 19:39:38	CPUUtilization >= 60 for 1 datapoints within 5 minutes
TargetTracking-webapp-asg-AlarmHigh-0b57cf39-d6d5-4eda-8035-f6dd0c46163d	OK	2025-07-03 19:34:12	CPUUtilization > 50 for 3 datapoints within 3 minutes

## 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.

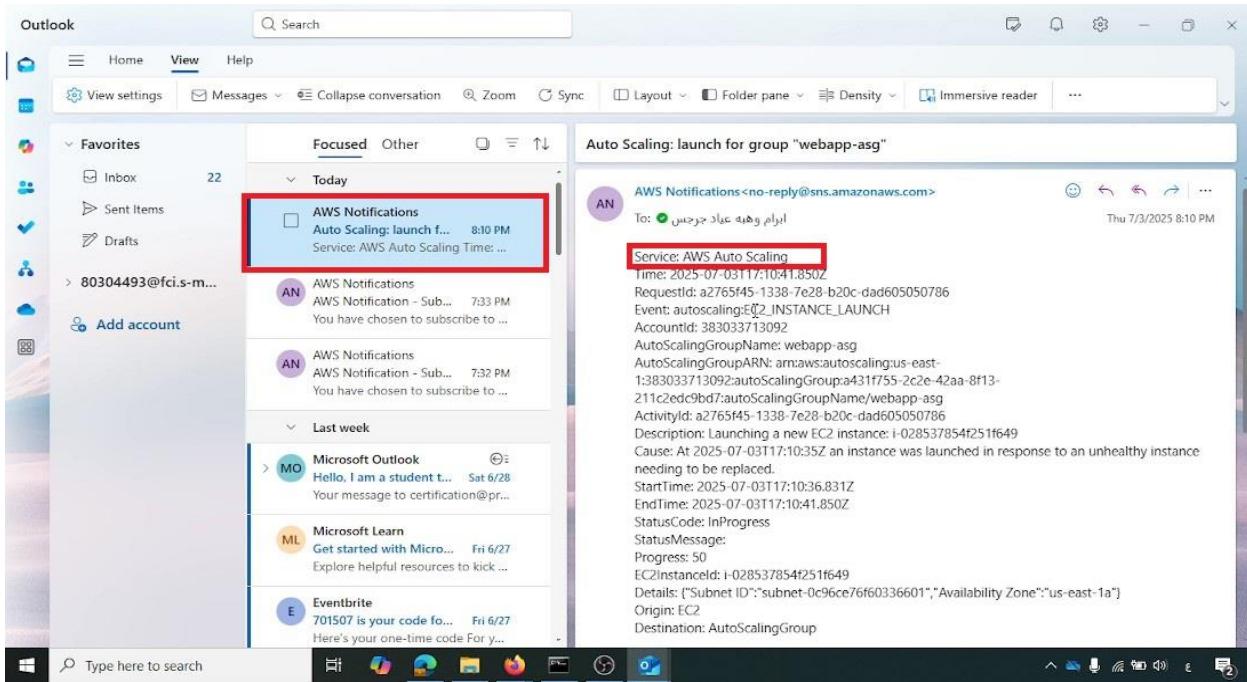


The screenshot shows the AWS EC2 Instances interface. On the left, there's a navigation sidebar with 'EC2' selected. Under 'Instances', there are several items: 'Instances Types', 'Launch Templates', 'Spot Requests', 'Savings Plans', 'Reserved Instances', 'Dedicated Hosts', 'Capacity Reservations', 'Images', and 'Elastic Block Store'. The main area displays a table of instances:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability zone
static-web-demo	i-0b3b1d873d516d245	Running	t2.micro	...	<a href="#">View alarms +</a>	us-east-1
	i-056c3cbf7f8651a93d	Running	t2.micro	...	<a href="#">View alarms +</a>	us-east-1
	i-0ae30e23b234684d7	Running	t2.micro	...	<a href="#">View alarms +</a>	us-east-1

## 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



The screenshot shows an email in Microsoft Outlook. The subject of the email is "Auto Scaling: launch for group 'webapp-asg'". The email is from "AWS Notifications <no-reply@sns.amazonaws.com>" and is dated "Thu 7/3/2025 8:10 PM". The body of the email contains detailed information about the Auto Scaling event:

```

AWS Notifications<no-reply@sns.amazonaws.com>
To: ابرام وساد علاء جرجاني
Service: AWS Auto Scaling
Time: 2025-07-03T17:10:41.850Z
RequestID: a2765f45-1338-7e28-b20c-dad605050786
Event: autoscaling:EC2_INSTANCE_LAUNCH
AccountID: 383033713092
AutoScalingGroupName: webapp-asg
AutoScalingGroupARN: arn:aws:autoscaling:us-east-1:383033713092:autoScalingGroup:a431f755-2c2e-42aa-8f13-211c2edc9bd7:autoScalingGroupName/webapp-asg
ActivityID: a2765f45-1338-7e28-b20c-dad605050786
Description: Launching a new EC2 instance: i-028537854f251f649
Cause: At 2025-07-03T17:10:35Z an instance was launched in response to an unhealthy instance needing to be replaced.
StartTime: 2025-07-03T17:10:36.831Z
EndTime: 2025-07-03T17:10:41.850Z
StatusCode: InProgress
StatusMessage:
Progress: 50
EC2InstanceId: i-028537854f251f649
Details: {"Subnet ID": "subnet-0c96ce76f60336601", "Availability Zone": "us-east-1a"}
Origin: EC2
Destination: AutoScalingGroup

```

## 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

