

PROJECT 2

Pod 15 DevOps Project Documentation – High Availability Hosting for MediCare

Team Lead: Chinazor Nwode

Assistant Lead: Ifunanya Benedicta

Team: Pod 15 – DevOps Track

Project Objective: To build and document a **fault-tolerant AWS VPC infrastructure** for the MediCare healthcare website, with **redundancy across 2 Availability Zones, automatic DNS failover via Route 53, and web hosting on EC2 instances.**

Overview

This project showcases our ability to:

- Design a **2-tier network** using AWS VPC
- Deploy **EC2 instances across two AZs**
- Secure the network with NACLs and security groups
- Deploy a public website with **SSH-based configuration**
- Register a custom domain and configure **DNS failover** using Route 53 health checks

We also documented our progress visually with **screenshots at every stage** to ensure traceability and clarity during review and grading.

Architecture Components

Component	Description
VPC	<code>10.0.0.0/16</code> custom VPC for network isolation
Public Subnet A	<code>10.0.1.0/24</code> in Availability Zone A
Public Subnet B	<code>10.0.3.0/24</code> in Availability Zone B
Private Subnet A	<code>10.0.2.0/24</code> in Availability Zone A
Private Subnet B	<code>10.0.4.0/24</code> in Availability Zone B
Internet Gateway	Enables public access for web traffic
Route Tables	Define traffic flow (<code>0.0.0.0/0</code> routed to IGW)
EC2 Instances (2)	Amazon Linux-based Nginx web servers in AZ-A and AZ-B
Security Groups	Restrict and allow traffic based on port and IP rules
NACLs	Stateless firewalls securing subnet-level traffic flows
Route 53	Manages DNS failover between the EC2 instances

Step 1: VPC and Subnet Setup

We built the foundation of our cloud network by setting up a **Virtual Private Cloud (VPC)** and **subnets**. A VPC is like your private office space in the cloud — it gives you full control over networking. Subnets are the rooms in that office, where specific services live.

We designed the network to span across **two Availability Zones (AZ-A and AZ-B)** so that if one zone goes down, the other can keep the service running.

1.1 Create a New VPC

The screenshot shows the AWS VPC dashboard. On the left, there's a sidebar with options like 'Virtual private cloud' (Your VPCs, Subnets, Route tables, Internet gateways, Egress-only Internet gateways, Carrier gateways, DHCP option sets, Elastic IPs, Managed prefix lists, NAT gateways, Peering connections, Route servers), 'Security' (Network ACLs), and links to CloudShell and Feedback. The main area is titled 'Your VPCs (1) Info' and shows a table with one row. The table columns are Name, VPC ID, State, Block Public..., IPv4 CIDR, and IPv6 CIDR. The single entry is 'vpc-006426dbd9945a85a' with state 'Available', 'Block Public...' set to 'Off', 'IPv4 CIDR' as '172.31.0.0/16', and 'IPv6 CIDR' as '-'. Below the table, it says 'Select a VPC above'. At the top right, there are 'Actions' and a 'Create VPC' button, which is highlighted with a red box.

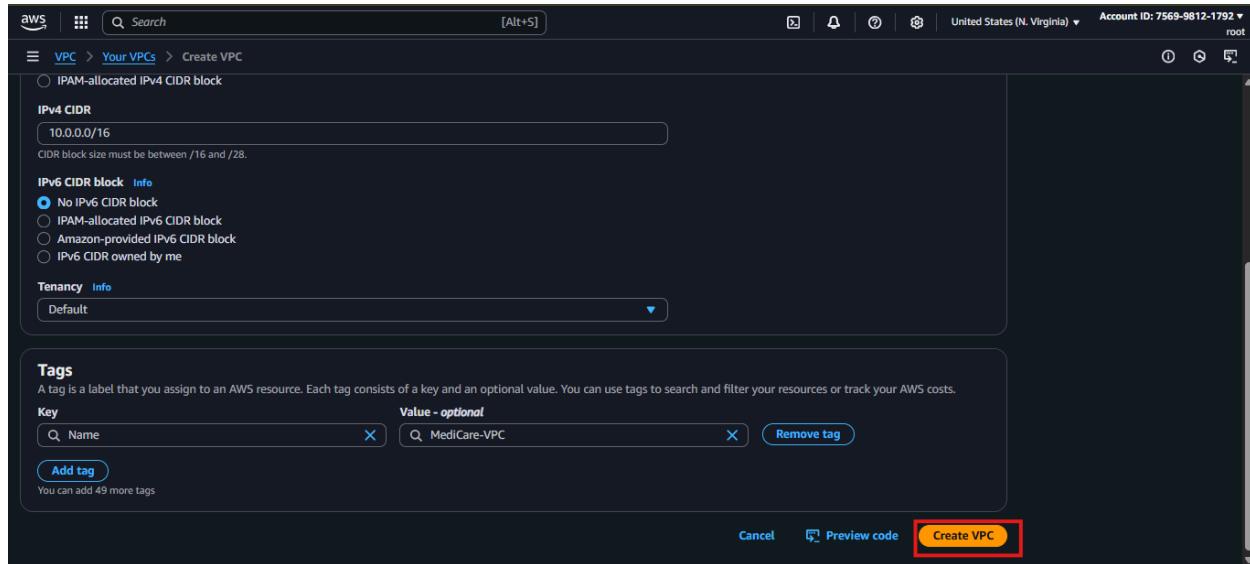
We started by creating a custom VPC named **MediCare-VPC**.

- Went to the VPC dashboard in AWS.
- Clicked "Create VPC" and chose VPC only.
- Gave it the CIDR block: **10.0.0.0/16**

The screenshot shows the 'Create VPC' form. It has several sections:

- VPC settings**:
 - Resources to create**: A radio button for 'VPC only' (selected) and 'VPC and more'.
 - Name tag - optional**: A text input field containing 'MediCare-VPC'.
 - IPv4 CIDR block**: A dropdown menu with 'Info' and three options: 'IPV4 CIDR manual input' (selected), 'IPAM-allocated IPv4 CIDR block', and 'Amazon-provided IPv4 CIDR block'.
 - IPv4 CIDR**: An input field containing '10.0.0.0/16'.
 - IPv6 CIDR block**: A dropdown menu with 'Info' and four options: 'No IPv6 CIDR block' (selected), 'IPAM-allocated IPv6 CIDR block', 'Amazon-provided IPv6 CIDR block', and 'IPv6 CIDR owned by me'.
 - Tenancy**: A dropdown menu with 'Info' and 'Default' selected.

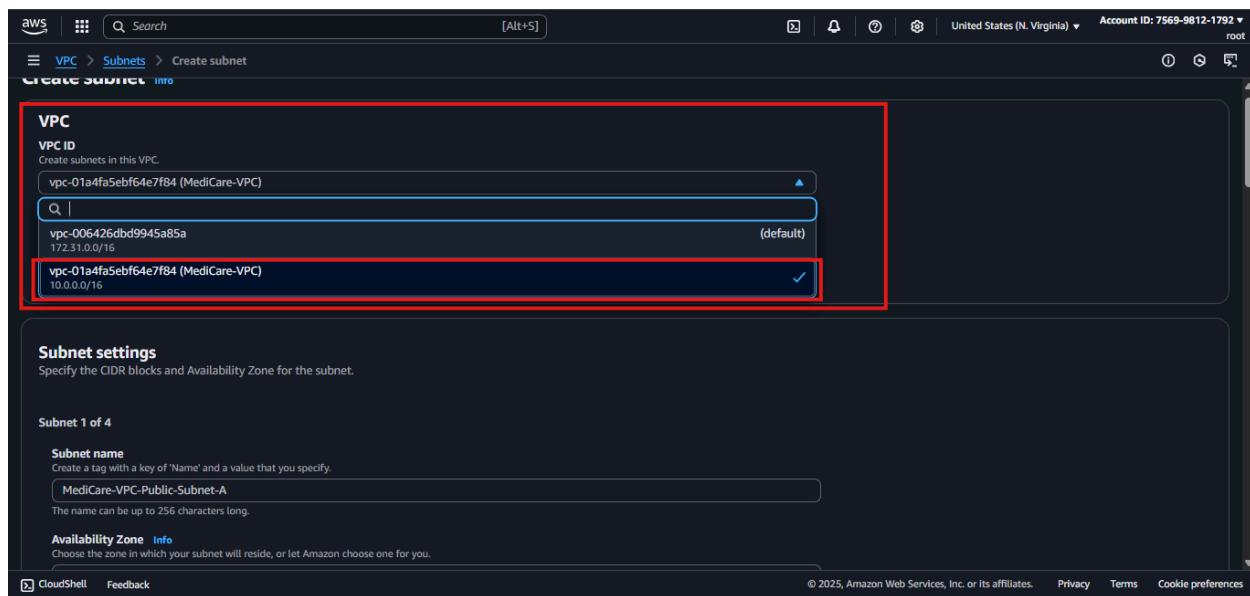
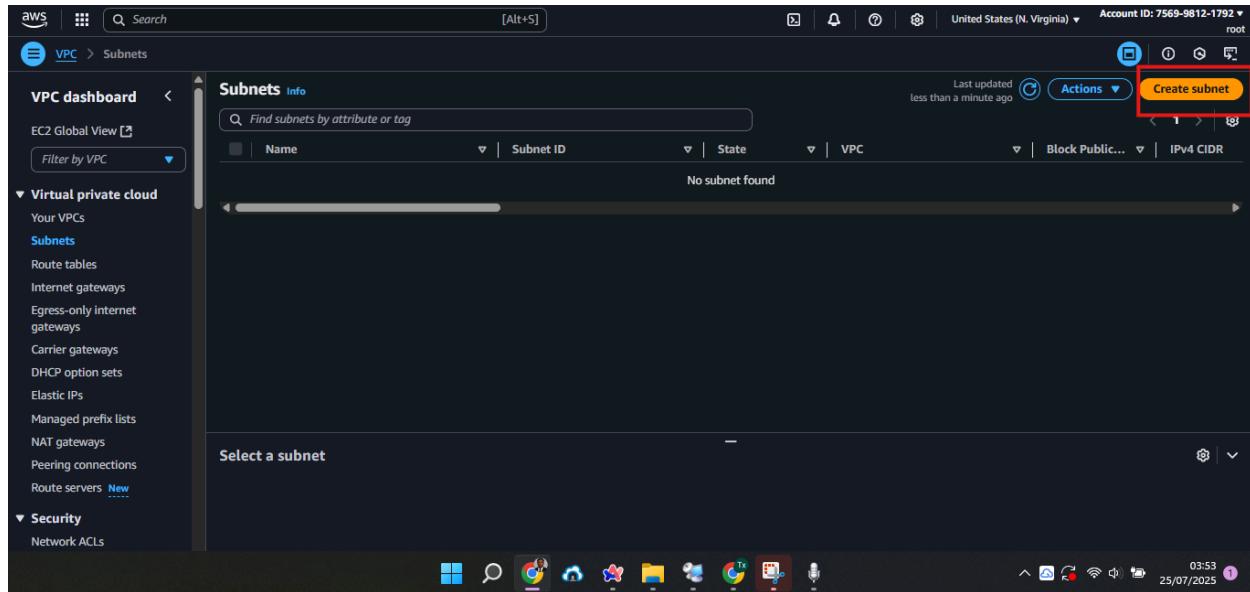
- VPC creation form with CIDR block set



- Confirmation screen showing the VPC listed in dashboard

1.2 Create Four Subnets (Public and Private in 2 AZs)

We created four subnets to split traffic across two availability zones:



- Public Subnet A → **10.0.1.0/24** in AZ-A

Subnet settings
Specify the CIDR blocks and Availability Zone for the subnet.

Subnet 1 of 4

Subnet name
Create a tag with a key of 'Name' and a value that you specify.
 The name can be up to 256 characters long.

Availability Zone Info
Choose the zone in which your subnet will reside, or let Amazon choose one for you.

IPv4 VPC CIDR block Info
Choose the VPC's IPv4 CIDR block for the subnet. The subnet's IPv4 CIDR must lie within this block.

IPv4 subnet CIDR block
 256 IPs

Tags - optional

Key	Value - optional
<input type="text" value="Name"/>	<input type="text" value="MediCare-VPC-Public-Subnet-A"/>

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

- Private Subnet A → **10.0.2.0/24** in AZ-A

Subnet 2 of 4

Subnet name
Create a tag with a key of 'Name' and a value that you specify.
 The name can be up to 256 characters long.

Availability Zone Info
Choose the zone in which your subnet will reside, or let Amazon choose one for you.

IPv4 VPC CIDR block Info
Choose the VPC's IPv4 CIDR block for the subnet. The subnet's IPv4 CIDR must lie within this block.

IPv4 subnet CIDR block
 256 IPs

Tags - optional

Key	Value - optional
<input type="text" value="Name"/>	<input type="text" value="MediCare-VPC-Private-Subnet-A"/>

Add new tag
You can add 49 more tags.

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

- Public Subnet B → **10.0.3.0/24** in AZ-B

Subnet 3 of 4

Subnet name
Create a tag with a key of 'Name' and a value that you specify.
 The name can be up to 256 characters long.

Availability Zone Info
Choose the zone in which your subnet will reside, or let Amazon choose one for you.

IPv4 VPC CIDR block Info
Choose the VPC's IPv4 CIDR block for the subnet. The subnet's IPv4 CIDR must lie within this block.

IPv4 subnet CIDR block
 256 IPs

Tags - optional

Key	Value - optional
<input type="text" value="Name"/>	<input type="text" value="MediCare-VPC-Public-Subnet-B"/> <small>(Remove)</small>

Add new tag You can add 49 more tags. Remove

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

- Private Subnet B → **10.0.4.0/24** in AZ-B

Subnet 4 of 4

Subnet name
Create a tag with a key of 'Name' and a value that you specify.
 The name can be up to 256 characters long.

Availability Zone Info
Choose the zone in which your subnet will reside, or let Amazon choose one for you.

IPv4 VPC CIDR block Info
Choose the VPC's IPv4 CIDR block for the subnet. The subnet's IPv4 CIDR must lie within this block.

IPv4 subnet CIDR block
 256 IPs

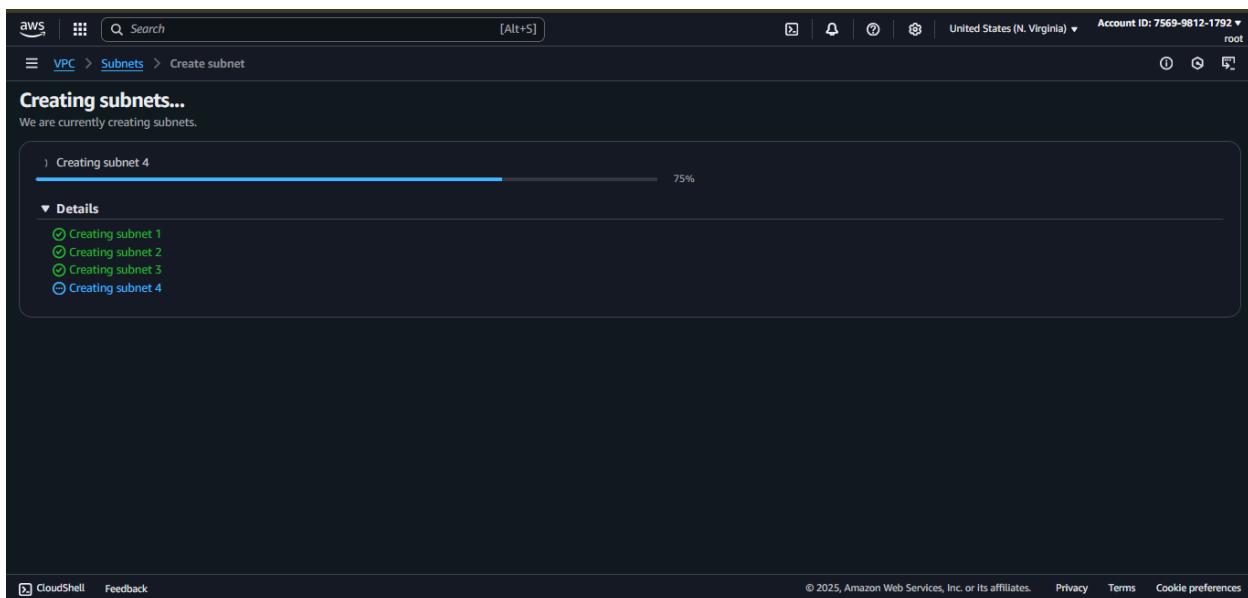
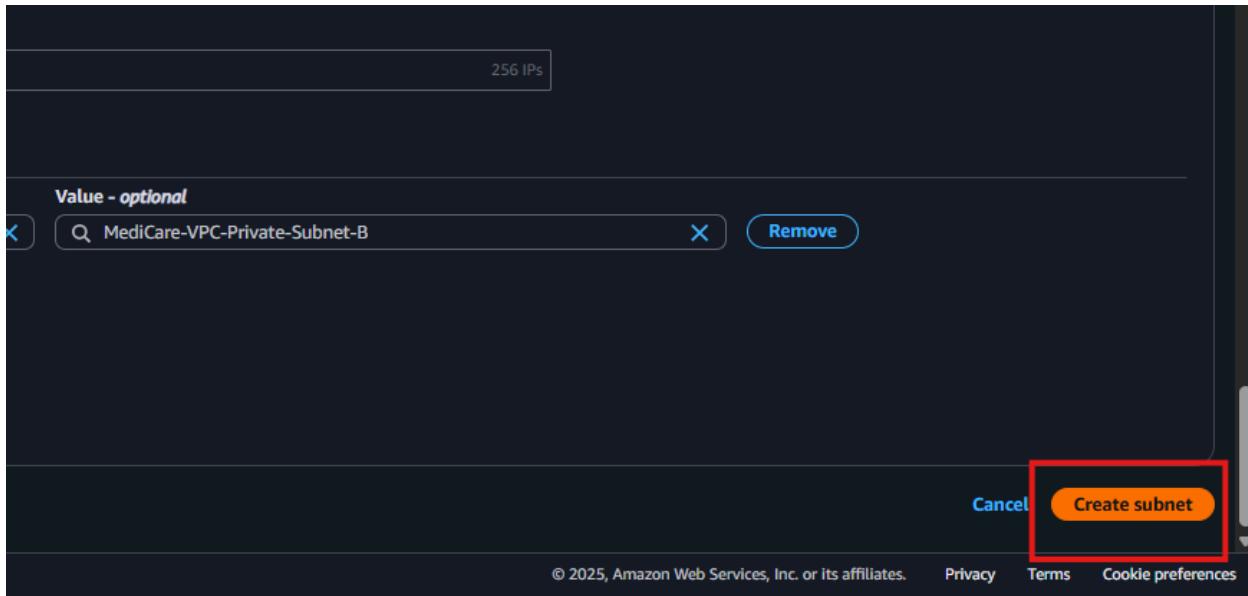
Tags - optional

Key	Value - optional
<input type="text" value="Name"/>	<input type="text" value="MediCare-VPC-Private-Subnet-B"/> <small>(Remove)</small>

Add new tag You can add 49 more tags. Remove

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

- Subnet creation form showing AZ and CIDR



We confirmed that all subnets were created successfully and attached to the VPC. This was done by reviewing the subnet list filtered by VPC ID.

The screenshot shows the AWS VPC dashboard with the 'Subnets' section selected. A green success message at the top states: "You have successfully created 4 subnets: subnet-0bf23edefc03b1513, subnet-03a49167d87bddb2a, subnet-009bc8684fa635633, subnet-049bde196b4db7c78". Below this, a table lists four subnets:

Name	Subnet ID	State	VPC	Block Public...	IPv4 CIDR
MediCare-VPC-Public-Subnet-B	subnet-009bc8684fa635633	Available	vpc-01a4fa5ebf64e7f84 Medi...	Off	10.0.3.0/24
MediCare-VPC-Private-Subnet-A	subnet-03a49167d87bddb2a	Available	vpc-01a4fa5ebf64e7f84 Medi...	Off	10.0.2.0/24
MediCare-VPC-Public-Subnet-A	subnet-0bf23edefc03b1513	Available	vpc-01a4fa5ebf64e7f84 Medi...	Off	10.0.1.0/24
MediCare-VPC-Private-Subnet-B	subnet-049bde196b4db7c78	Available	vpc-01a4fa5ebf64e7f84 Medi...	Off	10.0.4.0/24

A red box highlights the success message and the table. A yellow box highlights the 'Actions' button at the top right of the subnet list.

Step 2: Internet Gateway and Routing

To make our public subnets internet-accessible, we configured an **Internet Gateway (IGW)** and updated the **route table**.

The screenshot shows the AWS VPC dashboard with the 'Internet gateways' section selected. A yellow box highlights the 'Create internet gateway' button at the top right. The table below shows no results: "No internet gateways found in this Region".

2.1 Create and Attach Internet Gateway

- Created a new IGW named **Medicare-IGW**

The screenshot shows the 'Create internet gateway' page in the AWS VPC console. The 'Internet gateway settings' section includes a 'Name tag' field where 'Medicare-IGW' is entered. Below it, a 'Tags - optional' section shows a single tag 'Name: Medicare-IGW'. At the bottom right, there is a 'Create internet gateway' button, which is highlighted with a red box.

The screenshot shows the 'Internet gateways' page in the AWS VPC console. It displays a success message: 'The following internet gateway was created: igw-09c86081d43682281 - Medicare-IGW. You can now attach to a VPC to enable the VPC to communicate with the internet.' Below this, the 'igw-09c86081d43682281 / Medicare-IGW' card shows details like Internet gateway ID, State (Detached), VPC ID (-), and Owner (756998121792). The 'Attach to a VPC' button is highlighted with a red box.

- Attached it to the **MediCare-VPC**

Internet gateways (1/1) [Info](#)

Name	Internet gateway ID	State	VPC ID
Medicare-IGW	igw-09c86081d43682281	Detached	-

Actions

- View details
- Attach to VPC**
- Detach from VPC
- Manage tags
- Delete internet gateway

Attach to VPC (igw-09c86081d43682281) [Info](#)

VPC

Attach an internet gateway to a VPC to enable the VPC to communicate with the internet. Specify the VPC to attach below.

Available VPCs

Attach the internet gateway to this VPC.

Select a VPC

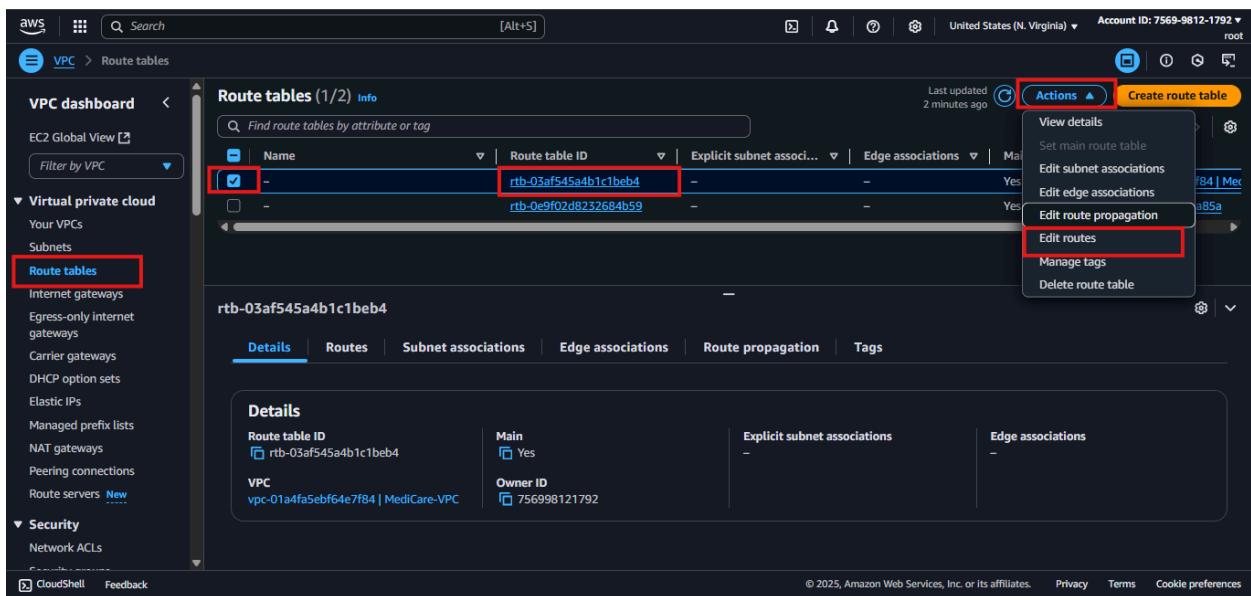
vpc-006426dbd9945a85a
vpc-01a4fa5ebf64e7f84 - MediCare-VPC

Cancel **Attach internet gateway**

- IGW attached to VPC confirmation



2.2 Create Route Table and Add Routes



- Created a custom route table
- Added a route `0.0.0.0/0` pointing to the IGW

aws | Search [Alt+S] | United States (N. Virginia) | Account ID: 7569-9812-1792 | root

VPC > Route tables > rtb-03af545a4b1c1beb4 > Edit routes

Edit routes

Destination	Target	Status	Propagated
10.0.0.0/16	local	Active	No
0.0.0.0/0	Internet Gateway	-	No
	igw-	-	
	igw-09c86081d43682281 (Medicare-IGW)	-	

Add route | Remove | Cancel | Preview | Save changes

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

aws | Search [Alt+S] | United States (N. Virginia) | Account ID: 7569-9812-1792 | root

VPC dashboard < | VPC > Route tables > rtb-03af545a4b1c1beb4

rtb-03af545a4b1c1beb4

Details Info

Route table ID rtb-03af545a4b1c1beb4	Main <input checked="" type="checkbox"/> Yes	Explicit subnet associations -	Edge associations -
VPC vpc-01a4fa5ebf64e7f84 MediCare-VPC	Owner ID 756998121792		

Actions ▾

Routes | Subnet associations | Edge associations | Route propagation | Tags

Routes (2)

Destination	Target	Status	Propagated
0.0.0.0/0	igw-09c86081d43682281	Active	No
10.0.0.0/16	local	Active	No

Both ▾ | Ed. routes | < 1 > | ⚙ | ▾

- Associated the route table with Public Subnet A and B

VPC dashboard < Search [Alt+S]

VPC > Route tables > rtb-03af545a4b1c1beb4 | Medicare-VPC | 756998121792

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

Explicit subnet associations (0)

No subnet associations
You do not have any subnet associations.

Edit subnet associations

Subnets without explicit associations (4)

The following subnets have not been explicitly associated with any route tables and are therefore associated with the main route table:

Name	Subnet ID	IPv4 CIDR	IPv6 CIDR
MediCare-VPC-Public-Subnet-B	subnet-009bc8684fa635633	10.0.3.0/24	-
MediCare-VPC-Private-Subnet-A	subnet-03a49167d87bdd2a	10.0.2.0/24	-
MediCare-VPC-Public-Subnet-A	subnet-0bf23edefc03b1513	10.0.1.0/24	-
MediCare-VPC-Private-Subnet-B	subnet-049bde196b4d87c78	10.0.4.0/24	-

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

aws < Search [Alt+S]

VPC > Route tables > rtb-03af545a4b1c1beb4 > Edit subnet associations

Edit subnet associations

Change which subnets are associated with this route table.

Available subnets (2/4)

Name	Subnet ID	IPv4 CIDR	IPv6 CIDR	Route table ID
<input checked="" type="checkbox"/> MediCare-VPC-Public-Subnet-B	subnet-009bc8684fa635633	10.0.3.0/24	-	Main (rtb-03af545a4b1c1beb4)
<input type="checkbox"/> MediCare-VPC-Private-Subnet-A	subnet-03a49167d87bdd2a	10.0.2.0/24	-	Main (rtb-03af545a4b1c1beb4)
<input checked="" type="checkbox"/> MediCare-VPC-Public-Subnet-A	subnet-0bf23edefc03b1513	10.0.1.0/24	-	Main (rtb-03af545a4b1c1beb4)
<input type="checkbox"/> MediCare-VPC-Private-Subnet-B	subnet-049bde196b4d87c78	10.0.4.0/24	-	Main (rtb-03af545a4b1c1beb4)

Selected subnets

subnet-0bf23edefc03b1513 / MediCare-VPC-Public-Subnet-A X | subnet-009bc8684fa635633 / MediCare-VPC-Public-Subnet-B X

Save associations

Cancel

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

- Route table configuration

Routes	Subnet associations	Edge associations	Route propagation	Tags
Routes (2)				
Filter routes				
Destination	Target	Status	Propagated	
0.0.0.0/0	igw-09c86081d43682281	Active	No	
10.0.0.0/16	local	Active	No	

- Add route screen showing IGW target
- Subnet association page showing Public Subnet A & B linked

Routes	Subnet associations	Edge associations	Route propagation	Tags
Explicit subnet associations (2)				
Find subnet association				
Name	Subnet ID	IPv4 CIDR	IPv6 CIDR	
MediCare-VPC-Public-Subnet-B	subnet-009bc8684fa635633	10.0.3.0/24	–	
MediCare-VPC-Public-Subnet-A	subnet-0bf23edefc03b1513	10.0.1.0/24	–	
Subnets without explicit associations (2)				
The following subnets have not been explicitly associated with any route tables and are therefore associated with the main route table:				

💻 Step 3: EC2 Instance Deployment

We launched two EC2 instances (Amazon Linux 2) in separate availability zones to host the MediCare website.

3.1 Launch EC2 Instances in Both AZs

- Launched Medicare-EC2-A in Public Subnet A (AZ-A)

Name and tags

Name: Medicare-Webserver-A

Application and OS Images (Amazon Machine Image)

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

Recent AMIs: Amazon Linux (highlighted with a red box), macOS, Ubuntu, Windows, Red Hat, SUSE Linux, Debian

Virtual server type (instance type): t2.micro

Summary

Number of instances: 1

Software Image (AMI): Amazon Linux 2023 AMI 2023.8.2... (read more)

Virtual server type (instance type): t2.micro

Firewall (security group): New security group

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

Free tier: In your first year of opening an AWS account, you get 750 hours per month of t2.micro instance usage (or t3.micro where applicable).

Launch instance | Preview code

Key pair (login)

Key pair name - required: Websrvr-A-Ssh

Network settings

VPC - required: MediCare-VPC

Subnet: MediCare-VPC-Public-Subnet-A

Auto-assign public IP: Enable

Firewall (security groups): Create security group

Summary

Number of instances: 1

Software Image (AMI): Amazon Linux 2023 AMI 2023.8.2... (read more)

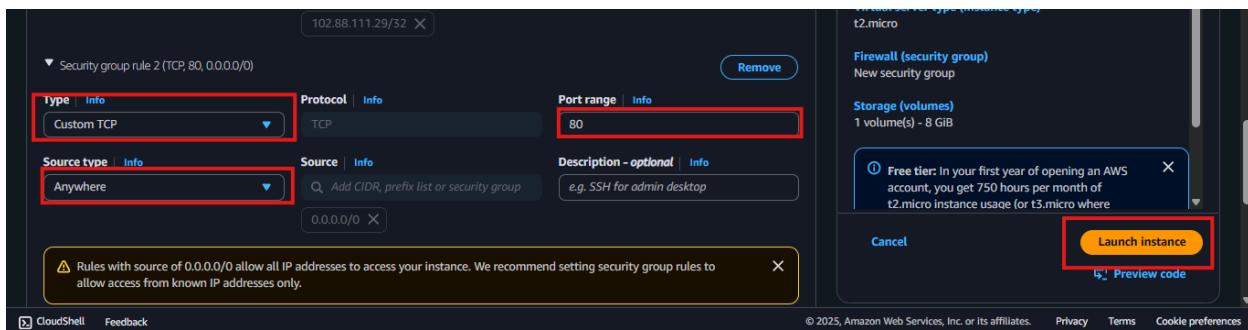
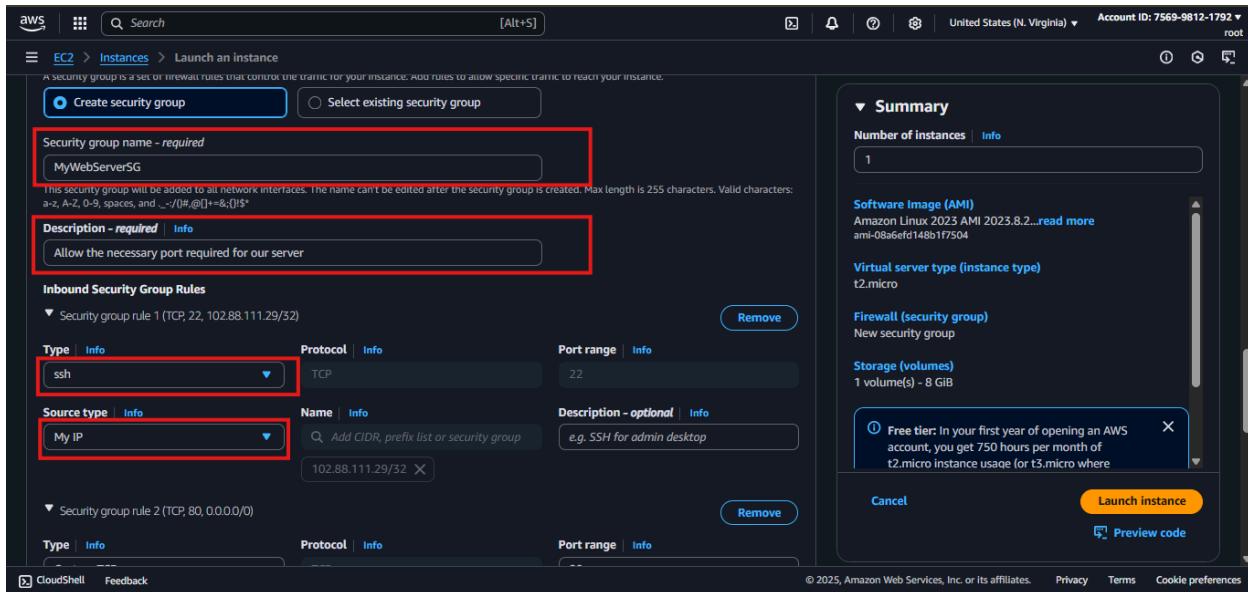
Virtual server type (instance type): t2.micro

Firewall (security group): New security group

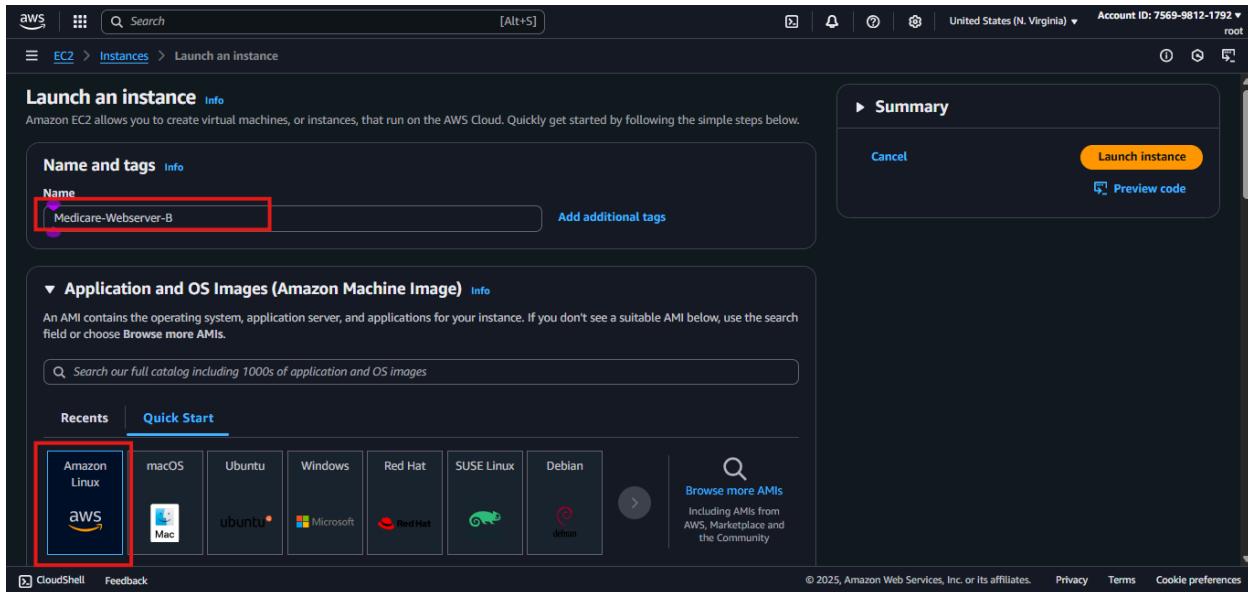
Storage (volumes): 1 volume(s) - 8 GiB

Free tier: In your first year of opening an AWS account, you get 750 hours per month of t2.micro instance usage (or t3.micro where applicable).

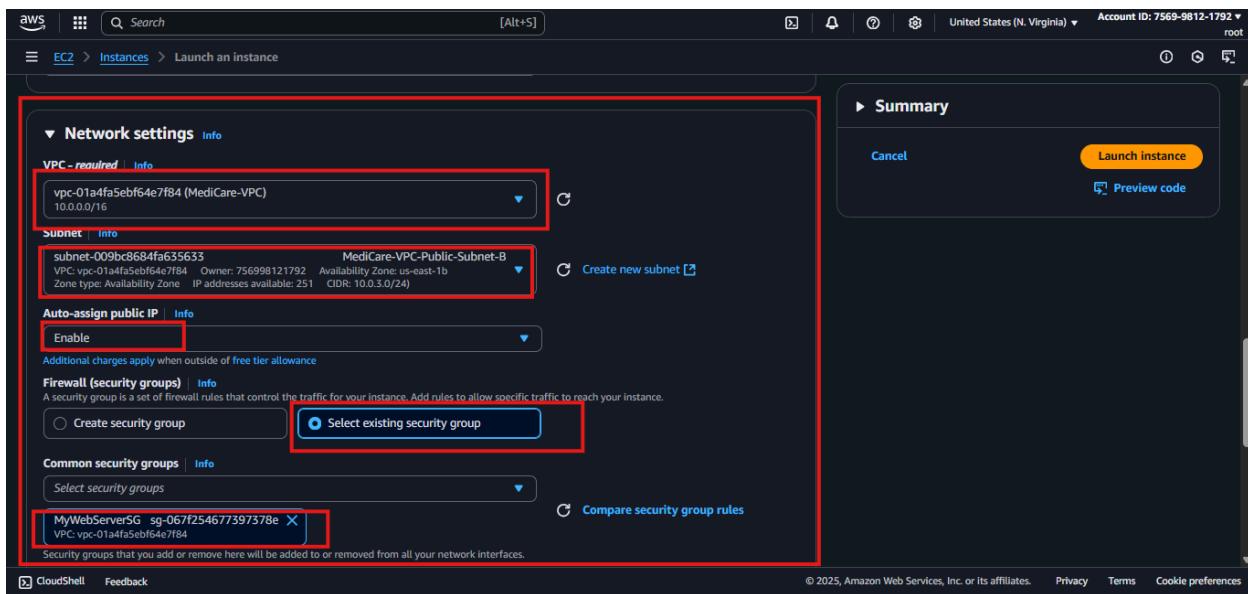
Launch instance | Preview code



- Launched Medicare-EC2-B in Public Subnet B (AZ-B)



- Enabled auto-assign public IP for both



3.2 Verify EC2 Status and Public IPs

- EC2 instance dashboard showing running state and public IP

Instances (2) Info						
Public IPv4 ...		Elastic IP	IPv6 IPs	Monitoring	Security group name	Key name
35.174.138.90	-	-	-	disabled	MyWebServerSG	Webserver-B...
44.193.213.71	-	-	-	disabled	MyWebServerSG	Webserver-A...

3.3 Launch EC2-A in Private Subnet A

- Navigate to EC2 → Launch Instance
- AMI: Amazon Linux 2

The screenshot shows the AWS EC2 'Launch an instance' wizard. In Step 1, 'Name and tags', the instance is named 'Medicare-Private-EC2-A'. In Step 2, 'Application and OS Images (Amazon Machine Image)', the user has selected 'Amazon Linux' from the 'Quick Start' section. Step 3, 'Summary', shows the configuration: 1 instance, AMI: Amazon Linux 2023.8.2, instance type t2.micro, New security group, and 1 volume(s) - 8 GiB. A tooltip for the Free tier is displayed, stating: 'Free tier: In your first year of opening an AWS account, you get 750 hours per month of t2.micro instance usage (or t3.micro where applicable)'.

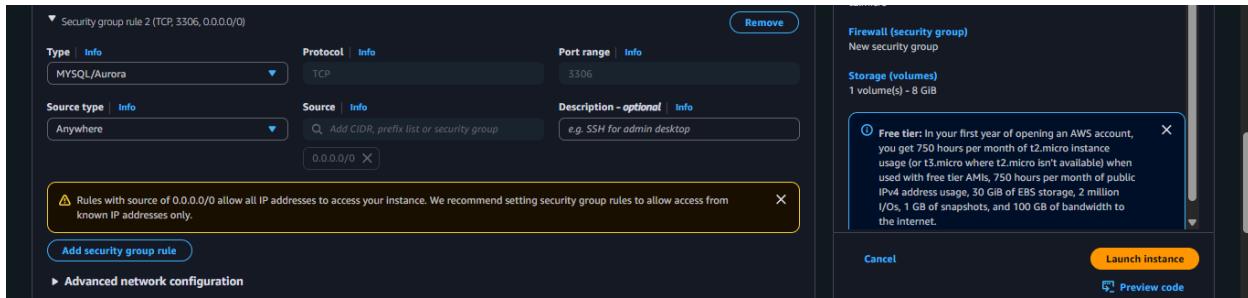
- Instance type: t2.micro (free tier)
- Network: Select our custom VPC
- Subnet: Choose **Private Subnet A (10.0.2.0/24)**
- Auto-assign public IP: **Disabled**
- Key pair: Select existing **.pem** key

The screenshot shows the AWS EC2 'Launch an instance' wizard. In the 'Network settings' section, a VPC is selected (Medicare-VPN) and a subnet (Medicare-VPN-Private-Subnet-A). Auto-assign public IP is enabled. In the 'Firewall (security groups)' section, a new security group is being created named 'Medicare-VPN-Private-Subnet-SG'. The summary on the right indicates 1 instance will be launched using the Amazon Linux 2023 AMI, t2.micro instance type, and a 1-volume(s) - 8 GiB EBS volume.

Security Group Configuration

- inbound SSH rule for the Elastic IP, which will be attached to our bastion host later
- Inbound MySQL Rule for Database

The screenshot shows the AWS EC2 'Launch an instance' wizard. In the 'Inbound Security Group Rules' section, two rules are defined: one for SSH (TCP port 22) from a custom source (IP 44.215.73.40) and another for MySQL/Aurora (TCP port 3306) from anywhere. The summary on the right indicates 1 instance will be launched using the Amazon Linux 2023 AMI, t2.micro instance type, and a 1-volume(s) - 8 GiB EBS volume.



3.4 Launch EC2-B in Private Subnet B

Repeat the same steps for AZ-B:

- Subnet: [Private Subnet B \(10.0.4.0/24\)](#)
- Confirm instance is private (no public IP)

Screenshots to Include:

- Instance launched in AZ-B private subnet
- Private IP confirmation

Step 4: NACL Setup (3+ Screenshots)

What We Did

We secured our EC2 instances using **shared security groups** and a **common NACL**, applied to both public subnets.

- Network ACL:
 - Allow ports 22 and 80 for inbound and outbound

The screenshot shows the 'Edit inbound rules' section of the AWS VPC Network ACL configuration. It lists three existing rules and a fourth rule being added:

- Rule number Info**: 100, Type Info: SSH (22), Protocol Info: TCP (6), Port range Info: 22, Source Info: 0.0.0.0/0, Allow/Deny Info: Allow.
- Rule number Info**: 200, Type Info: HTTPS (443), Protocol Info: TCP (6), Port range Info: 443, Source Info: 0.0.0.0/0, Allow/Deny Info: Allow.
- Rule number Info**: 300, Type Info: HTTP (80), Protocol Info: TCP (6), Port range Info: 80, Source Info: 0.0.0.0/0, Allow/Deny Info: Allow.
- Rule number Info**: +, Type Info: All traffic, Protocol Info: All, Port range Info: All, Source Info: 0.0.0.0/0, Allow/Deny Info: Deny.

Buttons at the bottom include 'Add new rule', 'Sort by rule number', 'Cancel', 'Preview changes', and 'Save changes'.

The screenshot shows the 'Edit outbound rules' section of the AWS VPC Network ACL configuration. It lists three existing rules and a fourth rule being added:

- Rule number Info**: 100, Type Info: SSH (22), Protocol Info: TCP (6), Port range Info: 22, Destination Info: 0.0.0.0/0, Allow/Deny Info: Allow.
- Rule number Info**: 200, Type Info: HTTPS (443), Protocol Info: TCP (6), Port range Info: 443, Destination Info: 0.0.0.0/0, Allow/Deny Info: Allow.
- Rule number Info**: 300, Type Info: HTTP (80), Protocol Info: TCP (6), Port range Info: 80, Destination Info: 0.0.0.0/0, Allow/Deny Info: Allow.
- Rule number Info**: +, Type Info: All traffic, Protocol Info: All, Port range Info: All, Destination Info: 0.0.0.0/0, Allow/Deny Info: Deny.

Buttons at the bottom include 'Add new rule', 'Sort by rule number', 'Cancel', 'Preview changes', and 'Save changes'.

- Associated with both public subnets

The screenshot shows the AWS VPC Network ACLs interface. The URL is `ad-00e81b42940d27282 / MyNACL > Edit subnet associations`. The title is "Edit subnet associations". A sub-header says "Change which subnets are associated with this network ACL." Below this is a table titled "Available subnets (2/4)". It lists four subnets: "MediCare-VPC-Public-Subnet-B" (selected), "MediCare-VPC-Private-Subnet-A", "MediCare-VPC-Public-Subnet-A" (selected), and "MediCare-VPC-Private-Subnet-B". The "Selected subnets" section contains the same two subnets. At the bottom right are "Cancel" and "Save changes" buttons.

Name	Subnet ID	Associated with	Availability Zone	IPv4 CIDR	IPv6 CIDR
<input checked="" type="checkbox"/> MediCare-VPC-Public-Subnet-B	subnet-009bc8684fa635633	acl-0ba45e47b9512b08f	us-east-1b	10.0.3.0/24	-
<input type="checkbox"/> MediCare-VPC-Private-Subnet-A	subnet-03a49167d87bdb2a	acl-0ba45e47b9512b08f	us-east-1a	10.0.2.0/24	-
<input checked="" type="checkbox"/> MediCare-VPC-Public-Subnet-A	subnet-0bf23edec03b1513	acl-0ba45e47b9512b08f	us-east-1a	10.0.1.0/24	-
<input type="checkbox"/> MediCare-VPC-Private-Subnet-B	subnet-049bde196b4d87c78	acl-0ba45e47b9512b08f	us-east-1b	10.0.4.0/24	-

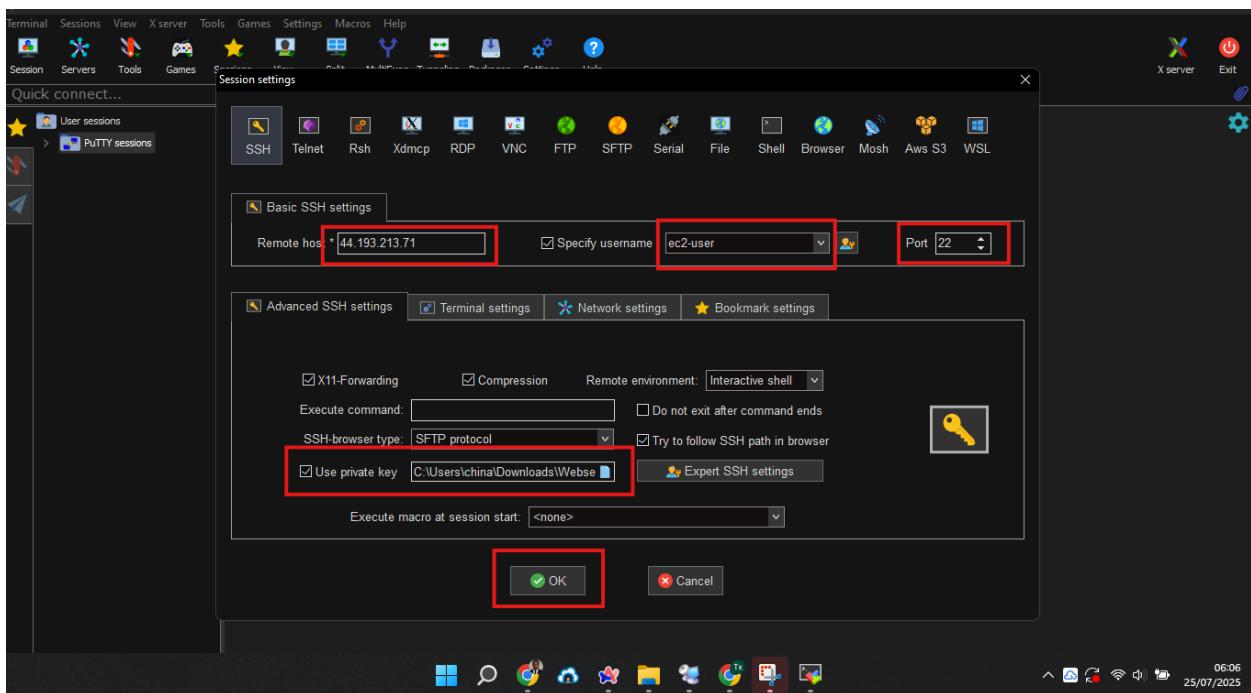
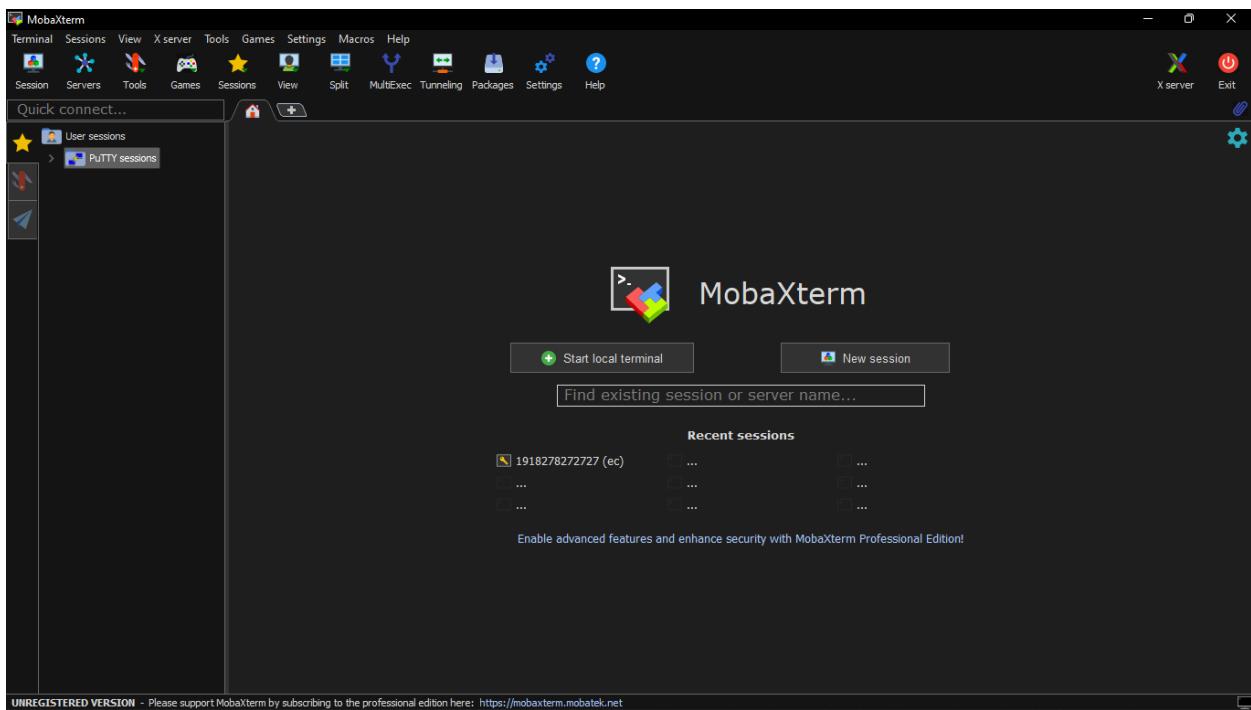


Step 5: SSH Access and Web Server Setup

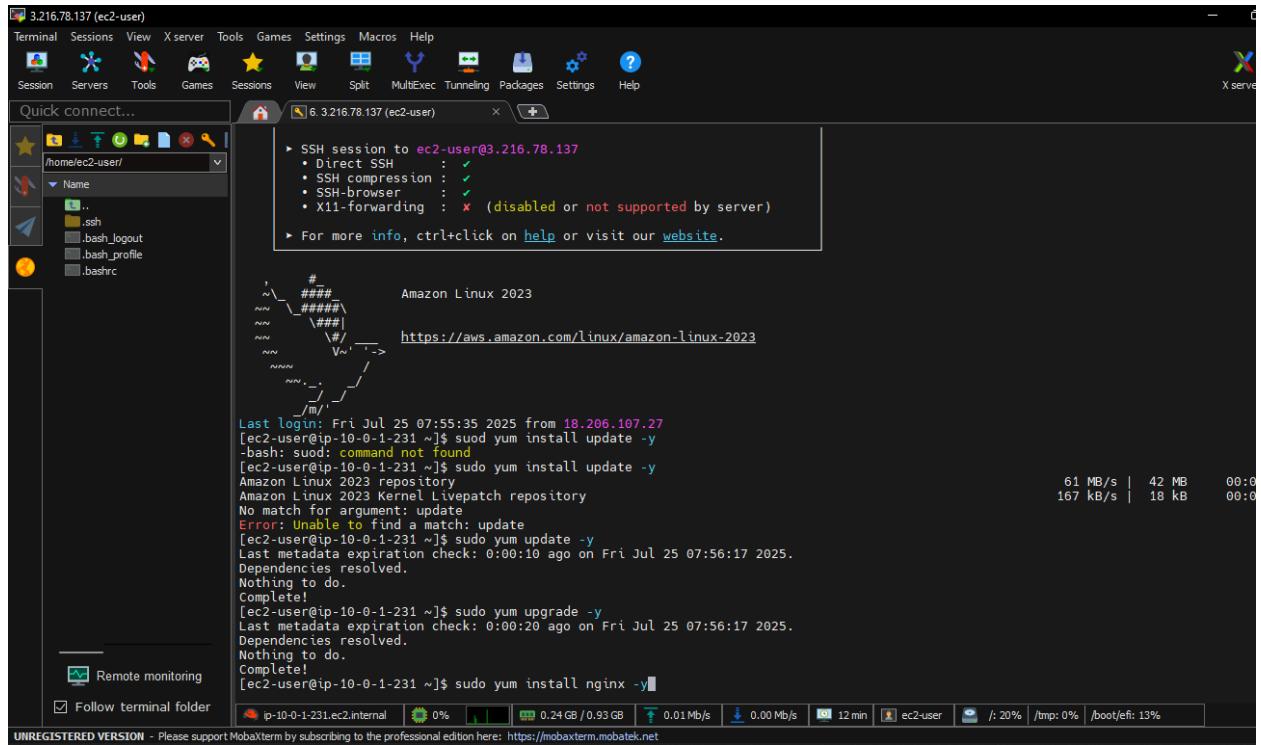
We installed a web server and deployed the MediCare site on each EC2.

5.1 SSH into EC2 Instances

- Used MobaXterm and .pem key to SSH into Medicare-EC2-A and Medicare-Webserver-B



- MobaXterm SSH session for Medicare-EC2-A



- Terminal connected confirmation

5.2 Install Nginx and Git

```
sudo yum update -y
sudo yum install nginx git -y
```

```

[ec2-user@ip-10-0-3-103 ~]$ sudo yum install git nginx -y
Amazon Linux 2023 repository
Amazon Linux 2023 Kernel Livepatch repository
Dependencies resolved.
=====
Package           Architecture     Version      Repository    Size
=====
Installing:
git              x86_64          2.50.1-1.amzn2023.0.1   amazonlinux  53 k
nginx            x86_64          1:1.28.0-1.amzn2023.0.1   amazonlinux  33 k
Installing dependencies:
generic-logos-httd noarch        18.0.0-12.amzn2023.0.3   amazonlinux  19 k
git-core          x86_64          2.50.1-1.amzn2023.0.1   amazonlinux  4.9 M
git-core-doc      noarch        2.50.1-1.amzn2023.0.1   amazonlinux  2.8 M
gperftools-libs   x86_64          2.9.1-1.amzn2023.0.3   amazonlinux  308 k
libunwind          x86_64          1.4.0-5.amzn2023.0.2   amazonlinux  66 k
nginx-core        x86_64          1:1.28.0-1.amzn2023.0.1   amazonlinux  669 k
nginx-filesystem  noarch        1:1.28.0-1.amzn2023.0.1   amazonlinux  9.5 k
nginx-mime types  noarch        2.1.49-3.amzn2023.0.3   amazonlinux  21 k
perl-Error         noarch        1:0.17029-5.amzn2023.0.2   amazonlinux  41 k
perl-File-Find    noarch        1.37-477.amzn2023.0.7   amazonlinux  25 k
perl-Git           noarch        2.50.1-1.amzn2023.0.1   amazonlinux  41 k
perl-TermReadKey  x86_64          2.38-9.amzn2023.0.2   amazonlinux  36 k
perl-lib            x86_64          0.65-477.amzn2023.0.7   amazonlinux  15 k
=====
Transaction Summary
=====
Install  15 Packages

Total download size: 9.0 M
Installed size: 45 M
Downloading Packages:
(1/15): git-2.50.1-1.amzn2023.0.1.x86_64.rpm          1.5 MB/s |  53 kB  00:00
(2/15): generic-logos-httd-18.0.0-12.amzn2023.0.3.noarch.rpm 513 kB/s |  19 kB  00:00
(3/15): gperftools-libs-2.9.1-1.amzn2023.0.3.x86_64.rpm  8.7 MB/s | 308 kB  00:00
(4/15): git-core-2.50.1-1.amzn2023.0.1.x86_64.rpm       47 MB/s |  4.9 MB  00:00
(5/15): libunwind-1.4.0-5.amzn2023.0.2.x86_64.rpm      1.9 MB/s |  66 kB  00:00
(6/15): git-core-doc-2.50.1-1.amzn2023.0.1.noarch.rpm   28 MB/s |  2.8 MB  00:00
(7/15): nginx-1.28.0-1.amzn2023.0.1.x86_64.rpm        985 kB/s |  33 kB  00:00

```

5.3 Deploy Website

```

git clone https://github.com/digitalwitchdemo/mediplus.git
cd mediplus
sudo mv * /usr/share/nginx/html
sudo systemctl restart nginx

```

```

Verifying : generic-logos-nginx-1.28.0-1.amzn2023.0.1.x86_64 1/15
Verifying : git-2.50.1-1.amzn2023.0.1.x86_64 2/15
Verifying : git-core-2.50.1-1.amzn2023.0.1.x86_64 3/15
Verifying : git-core-doc-2.50.1-1.amzn2023.0.1.noarch 4/15
Verifying : gperftools-libs-2.9.1-1.amzn2023.0.3.x86_64 5/15
Verifying : libunwind-1.4.0-5.amzn2023.0.2.x86_64 6/15
Verifying : nginx-1:1.28.0-1.amzn2023.0.1.x86_64 7/15
Verifying : nginx-core-1:1.28.0-1.amzn2023.0.1.x86_64 8/15
Verifying : nginx-fs-1:1.28.0-1.amzn2023.0.1.noarch 9/15
Verifying : nginx-mimetypes-2.1.49-3.amzn2023.0.3.noarch 10/15
Verifying : perl-Error-1:0.17029-5.amzn2023.0.2.noarch 11/15
Verifying : perl-File-Find-1.37-477.amzn2023.0.7.noarch 12/15
Verifying : perl-Git-2.50.1-1.amzn2023.0.1.noarch 13/15
Verifying : perl-TermReadKey-2.38-9.amzn2023.0.2.x86_64 14/15
Verifying : perl-lib-0.65-477.amzn2023.0.7.x86_64 15/15

Installed:
generic-logos-nginx-1.28.0-1.amzn2023.0.1.x86_64
git-2.50.1-1.amzn2023.0.1.x86_64
git-core-2.50.1-1.amzn2023.0.1.noarch
gperftools-libs-2.9.1-1.amzn2023.0.3.x86_64
libunwind-1.4.0-5.amzn2023.0.2.x86_64
nginx-1:1.28.0-1.amzn2023.0.1.x86_64
nginx-fs-1:1.28.0-1.amzn2023.0.1.noarch
nginx-mimetypes-2.1.49-3.amzn2023.0.3.noarch
perl-Error-1:0.17029-5.amzn2023.0.2.noarch
perl-Git-2.50.1-1.amzn2023.0.1.noarch
perl-TermReadKey-2.38-9.amzn2023.0.2.x86_64
perl-lib-0.65-477.amzn2023.0.7.x86_64

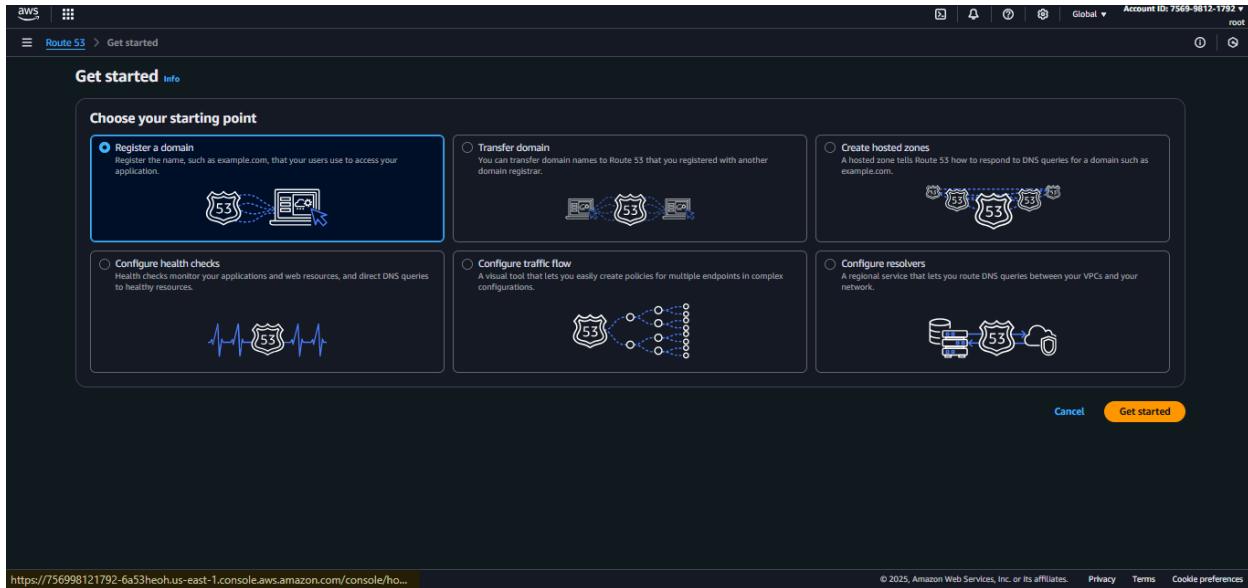
Complete!
[ec2-user@ip-10-0-3-103 ~]$ git clone https://github.com/digitalwitchdemo/mediplus.git
cd mediplus
sudo mv * /usr/share/nginx/html
sudo systemctl restart nginx
Cloning into 'mediplus'...
remote: Enumerating objects: 108, done.
remote: Counting objects: 100% (108/108), done.
remote: Compressing objects: 100% (96/96), done.
remote: Total 108 (delta 10), reused 92 (delta 5), pack-reused 0 (from 0)
Receiving objects: 100% (108/108), 4.89 MiB | 24.20 MiB/s, done.
Resolving deltas: 100% (10/10), done.
[ec2-user@ip-10-0-3-103 mediplus]$ █

```

Step 6: Domain Purchase and Route 53 Setup (3+ Screenshots)

What We Did

We purchased medicare.pod15.cloud from GoDaddy and set up **Amazon Route 53** to manage DNS, enabling automatic failover between Medicare-EC2-A and Medicare-Webserver-B.



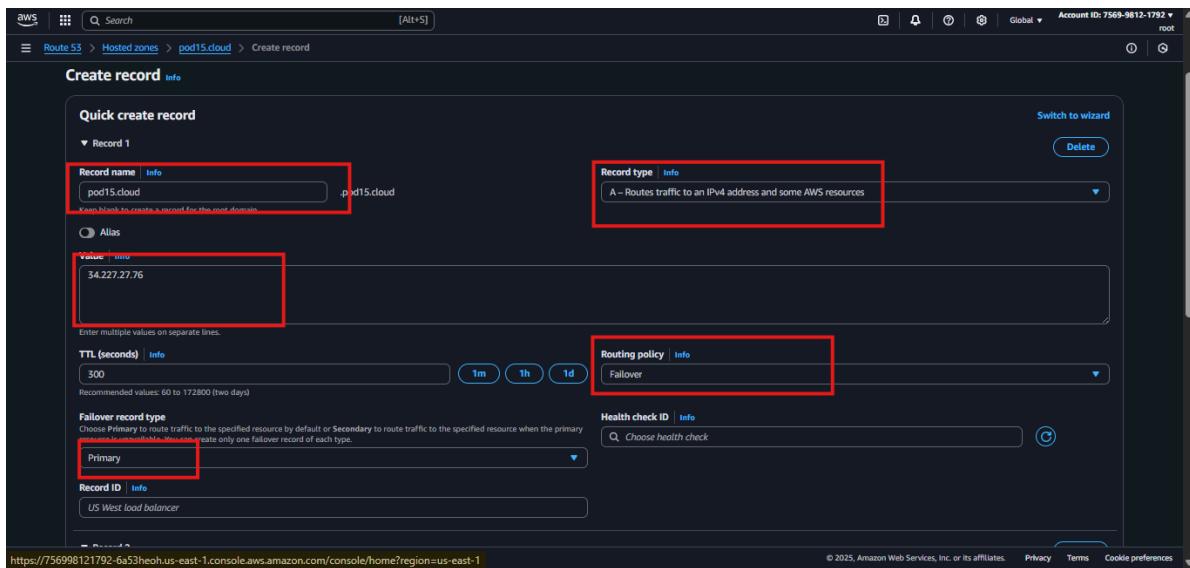
The screenshot shows the AWS Route 53 'Hosted zones' page. The left sidebar includes categories like IP-based routing, Traffic flow, Domains, and Resolver. The main area shows a success message: 'pod15.cloud was successfully created. Now you can create records in the hosted zone to specify how you want Route 53 to route traffic for your domain.' Below this is a table of existing records:

Record name	Type	Routing policy	Alias	Value/Route traffic to	TTL (seconds)
pod15.cloud	NS	Simple	No	ns-877.awsdns-45.net. ns-1965.awsdns-53.co.uk. ns-1170.awsdns-18.org. ns-471.awsdns-58.com.	172800
pod15.cloud	SOA	Simple	No	ns-877.awsdns-45.net. awsd...	900

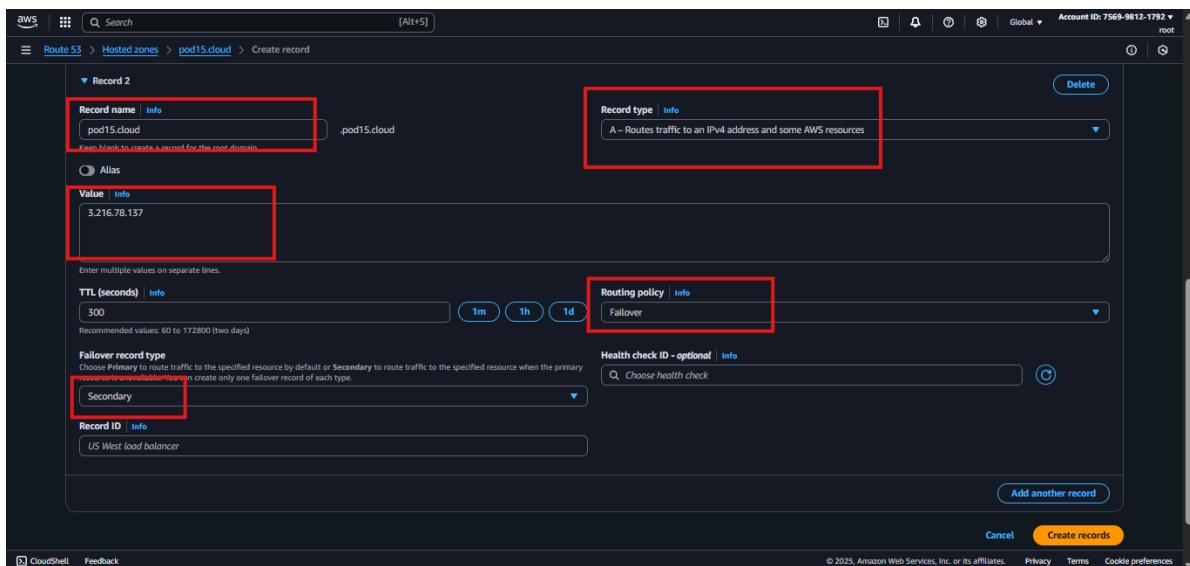
The right panel displays 'Record details' for the NS record, including its value (ns-877.awsdns-45.net), record type (NS), and TTL (172800 seconds). It also shows the 'Alias' field is set to 'No'.

✓ Actions Taken:

- Purchased domain
- Created Route 53 Hosted Zone for `medicare.pod15.cloud`
- Added two A records:
 - Primary → Medicare-EC2-A public IP



- Secondary → Medicare-Webserver-B public IP

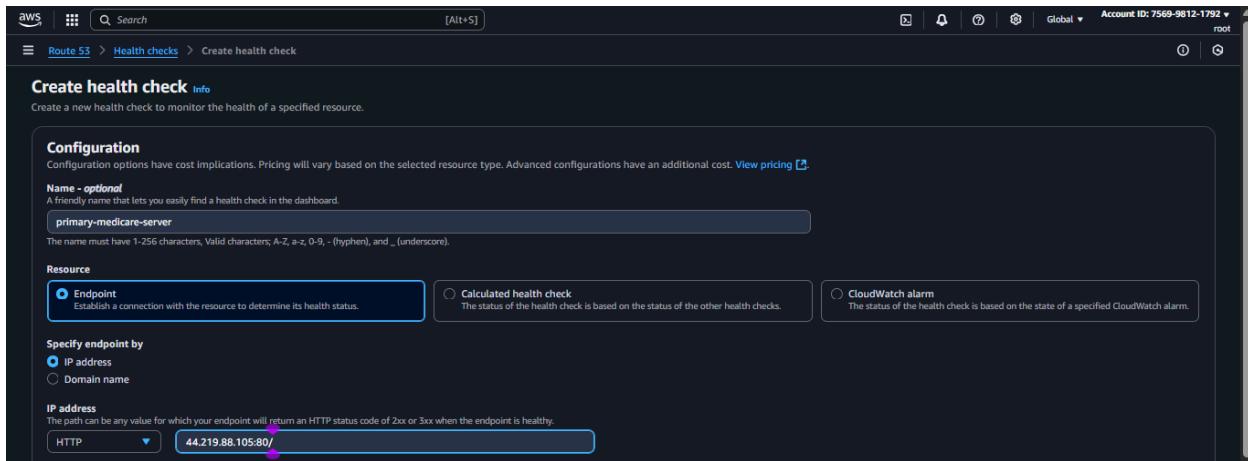


❤️ Step 7: Health Check and DNS Failover Testing (3+ Screenshots)

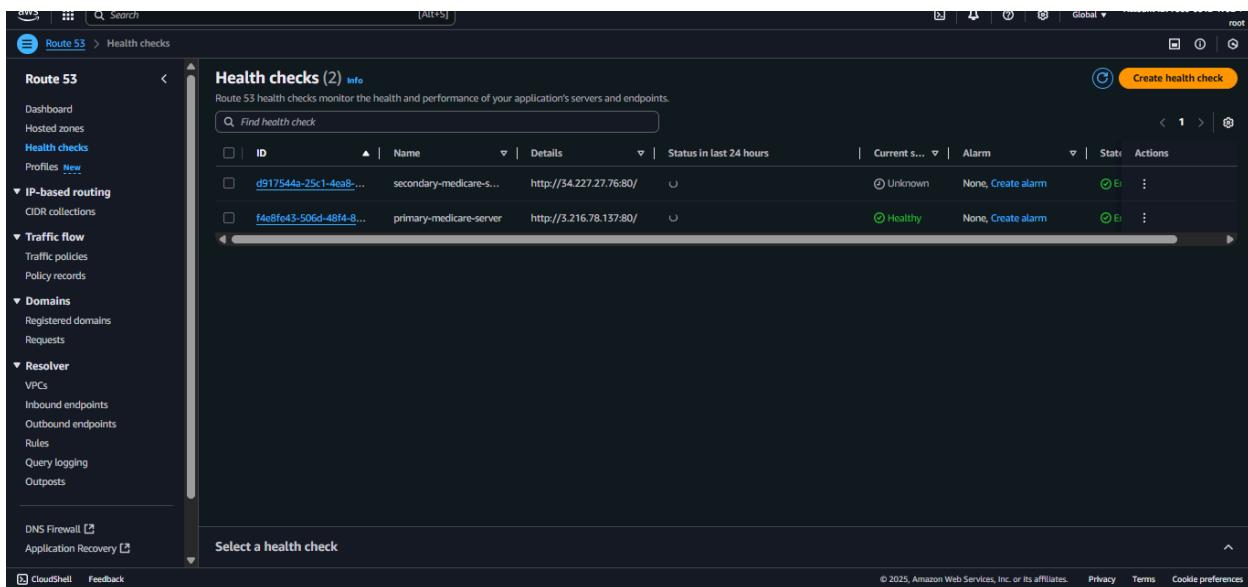
What We Did

We configured a **Route 53 health check** on Medicare-EC2-A so that if it becomes unhealthy, traffic is automatically routed to Medicare-Webserver-B.

- Created health check on Medicare-EC2-A using port 80 and / path



- repeated the same steps and created for secondary-medicare-server



- Linked health check to primary A record

Screenshot of the AWS Route 53 'Create record' interface for a hosted zone named 'medicare.pod15.cloud'.

Record name: medicare.pod15.cloud

Record type: A – Routes traffic to an IPv4 address and some AWS resources

Value: 44.219.88.105

TTL (seconds): 300

Routing policy: Failover

Failover record type: Primary

Record ID: primary

Health check ID: 6b1d3ddc-22e3-4d6e-85fd-589a02612874

Use: "6b1d3ddc-22e3-4d6e-85fd-589a02612874"

primary-medicare-server: 6b1d3ddc-22e3-4d6e-85fd-589a02612874

secondary-medicare-server: f4e8fe43-506d-48f4-856b-d514a87d3241

Create records

Screenshot of a medical website for 'Mediplus'.

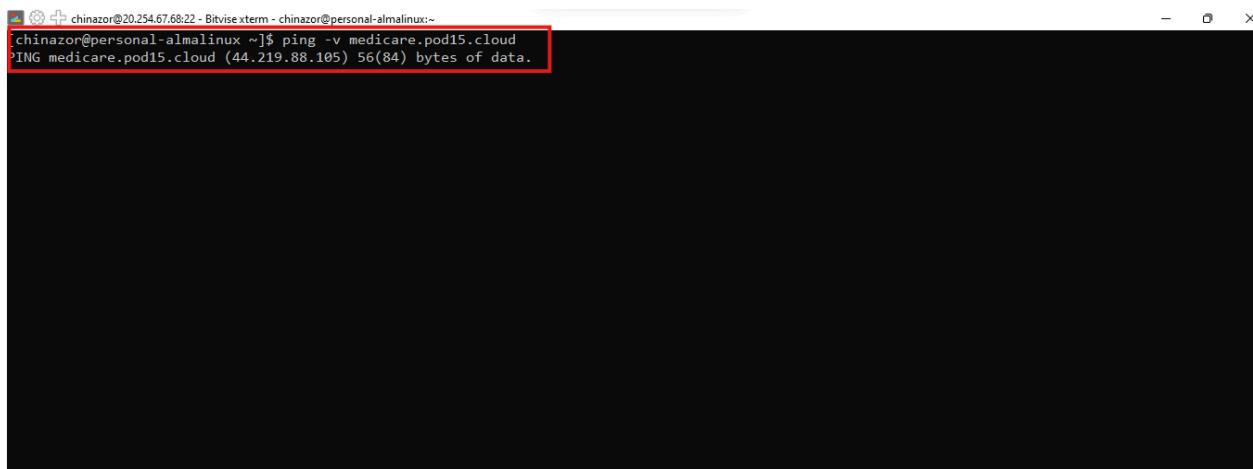
Header: Mediplus, Home, Doctors, Services, Pages, Blogs, Contact Us, Book Appointment, Get Pro.

Hero Section: We Provide Medical Services and Sport services That You Can Trust!

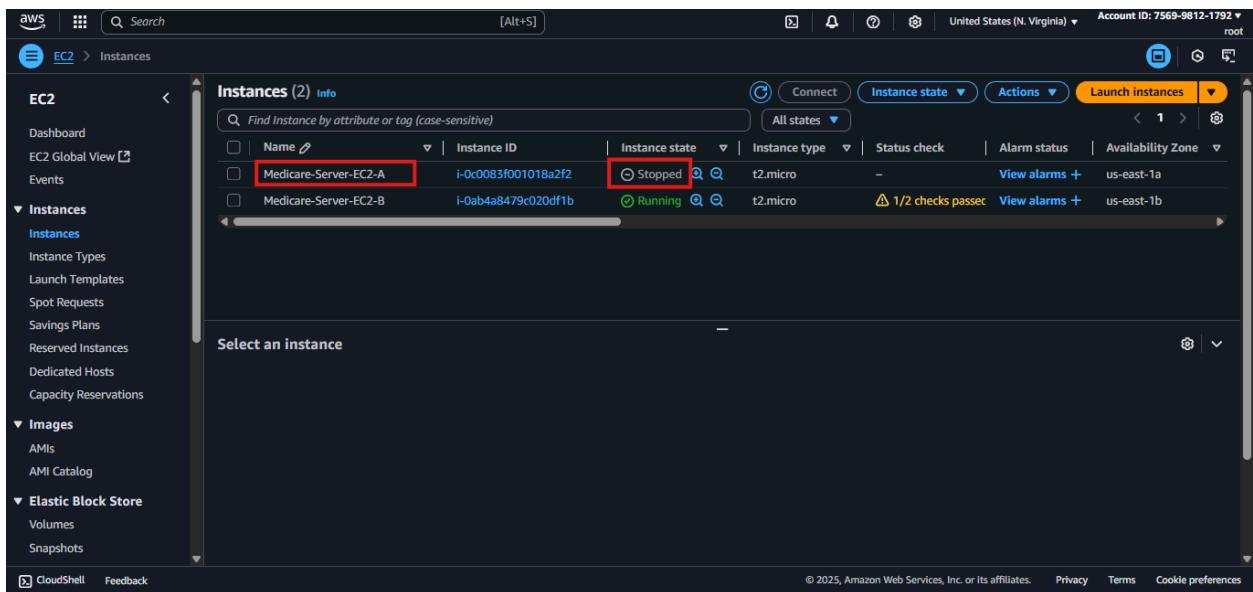
Text: Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris sed nisl pellentesque, faucibus libero eu, gravida quam.

Buttons: Get Appointment, About Us.

Footer: Lorem Amet, Fusce Porttitor, Donec luctus.



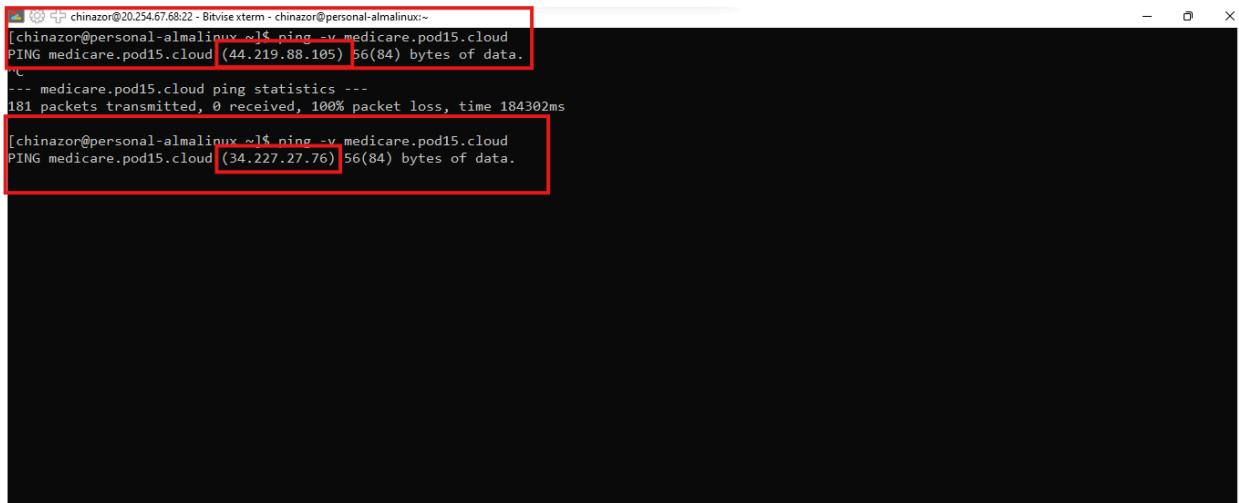
```
chinazor@personal-almalinux ~]$ ping -v medicare.pod15.cloud
PING medicare.pod15.cloud (44.219.88.105) 56(84) bytes of data.
```



The screenshot shows the AWS EC2 Instances page. The left sidebar is collapsed. The main area displays a table titled "Instances (2) Info". The table has columns: Name, Instance ID, Instance state, Instance type, Status check, Alarm status, and Availability Zone. Two instances are listed:

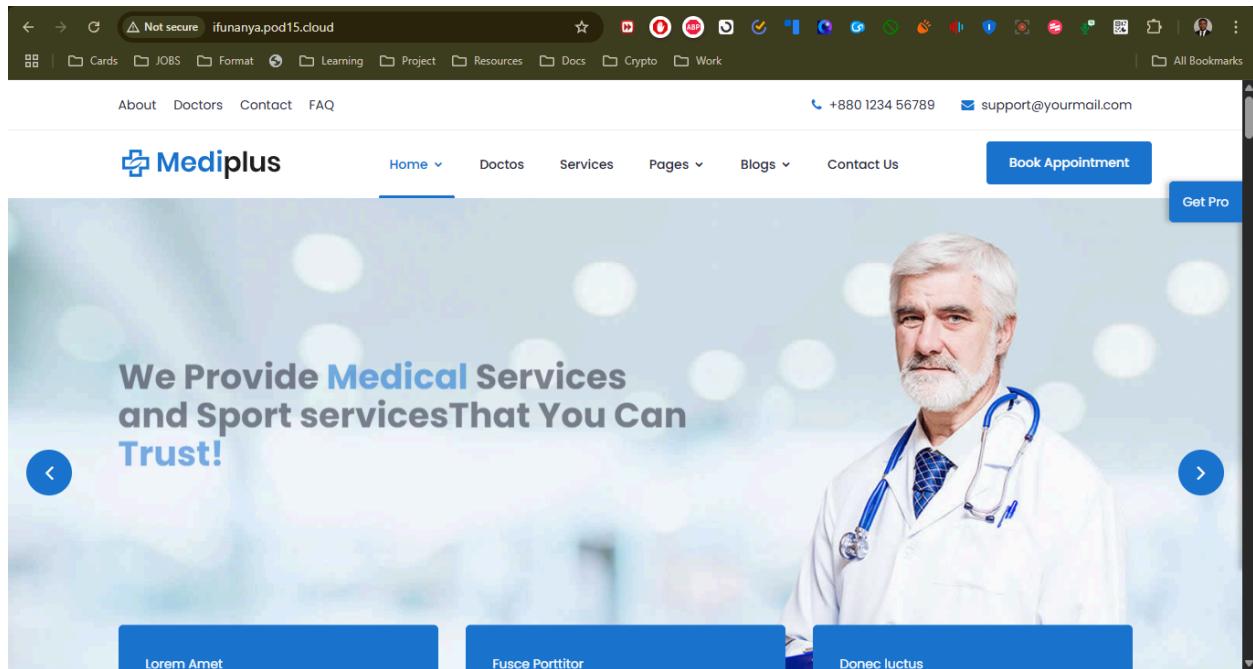
Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
Medicare-Server-EC2-A	i-0c0083f001018a2f2	Stopped	t2.micro	-	View alarms +	us-east-1a
Medicare-Server-EC2-B	i-0ab4a8479c020df1b	Running	t2.micro	⚠️ 1/2 checks passed	View alarms +	us-east-1b

A red box highlights the "Stopped" status of the first instance. Below the table, a message says "Select an instance".



```
[chinazor@personal-almalinux ~]$ ping -v medicare.pod15.cloud
PING medicare.pod15.cloud (44.219.88.105) 56(84) bytes of data.
^C
--- medicare.pod15.cloud ping statistics ---
181 packets transmitted, 0 received, 100% packet loss, time 184302ms

[chinazor@personal-almalinux ~]$ ping -v medicare.pod15.cloud
PING medicare.pod15.cloud (34.227.27.76) 56(84) bytes of data.
```



✓ Actions Taken:

- Shut down Medicare-EC2-A to test failover
- Route 53 failed over to Medicare-Webserver-B

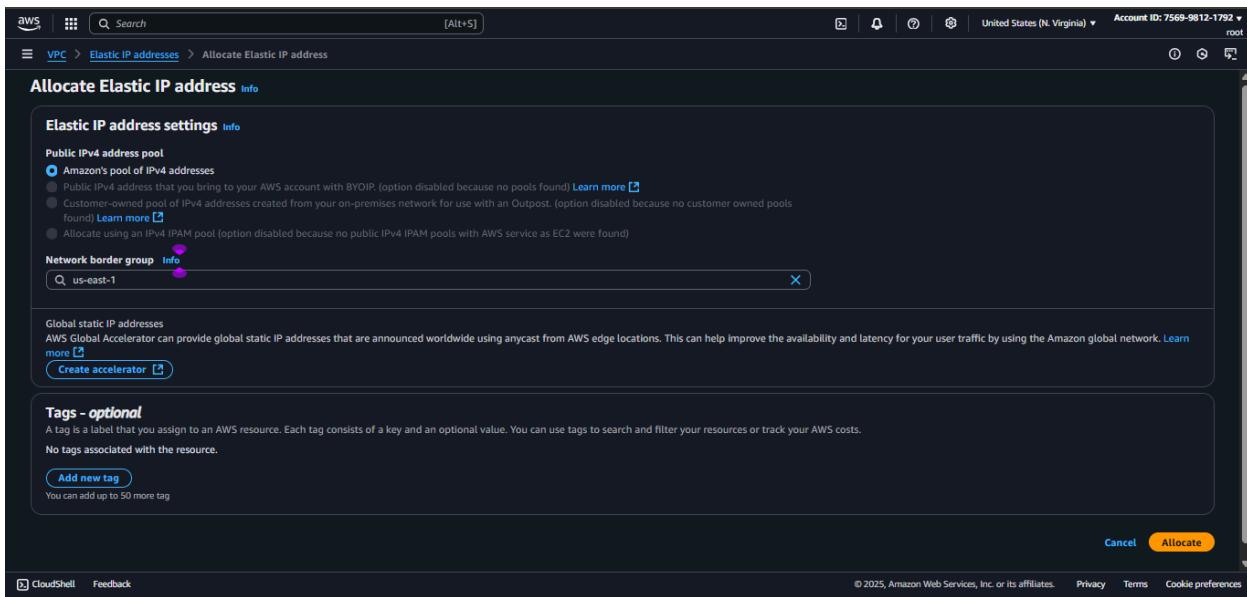
- Restarted Medicare-EC2-A to watch traffic switch back

Step 8: NAT Gateway & Bastion Host Configuration

To allow EC2 instances in **private subnets** to access the internet (for updates, packages, etc.), we configured a **NAT Gateway** in the public subnet. After that, we set up a **Bastion Host** to securely SSH into the private EC2s.

8.1 Create Elastic IP for NAT Gateway

- Navigate to **VPC** → **Elastic IPs** → **Allocate**



- Allocate a new Elastic IP (EIP) for the NAT Gateway
- Elastic IP allocation confirmation
- EIP listed under the allocated addresses

The screenshot shows the AWS VPC dashboard with the 'NAT gateways' section selected. There are two entries in the table:

Name	NAT gateway ID	Connectivity...	State	Primary public IP...	Primary private I...
Public-NAT-Gateway-Private-Subnet-B	nat-0984d9e9fdd03809	Public	Available	18.210.96.218	10.0.4.40
Public-NAT-Gateway-Private-Subnet-A	nat-08cb6031cf0577086	Public	Available	52.72.123.78	10.0.2.76

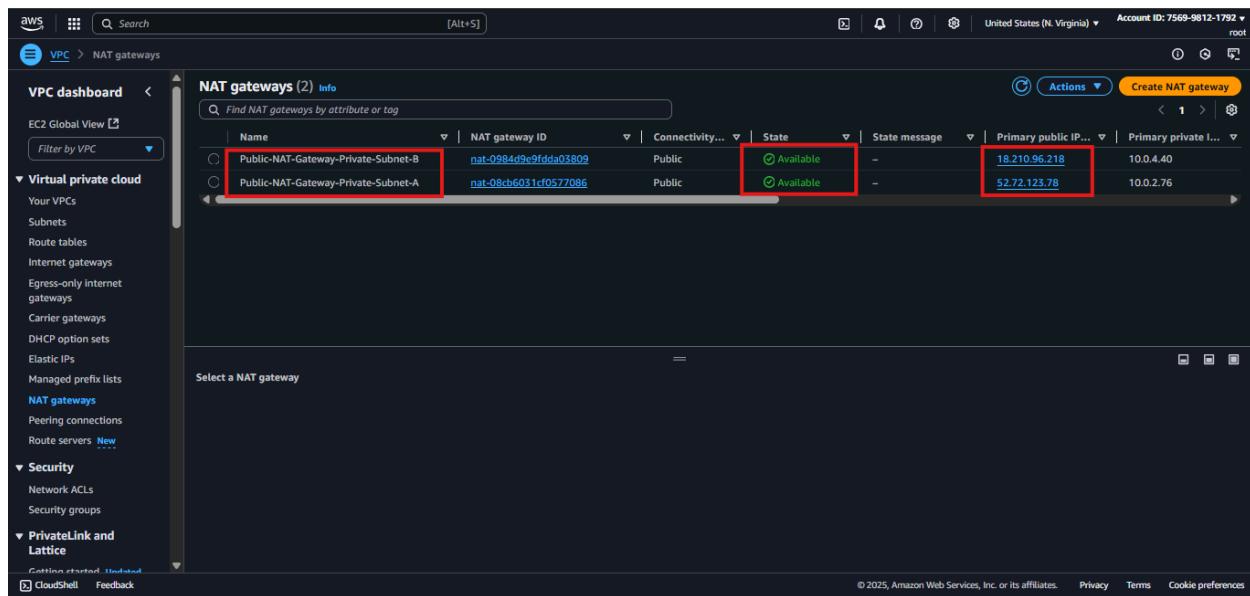
8.2 Create and Attach NAT Gateway

- Go to **VPC → NAT Gateways → Create**
- Select **Public Subnet A (10.0.1.0/24)**
- Use the previously allocated Elastic IP

The screenshot shows the 'Create NAT gateway' wizard. The configuration steps are as follows:

- NAT gateway settings**
 - Name - optional**: Public-NAT-Gateway-Private-Subnet-A
 - Subnet**: subnet-0c3a3c4f1a4ed6082 (Medicare-VPC-Private-Subnet-A)
 - Connectivity type**: Public (selected)
 - Elastic IP allocation ID - Info**: eipalloc-096cbe839efc29371
- Tags**: A tag named 'Name' is added with the value 'Public-NAT-Gateway-Private-Subnet-A'.

- NAT Gateway creation screen with correct subnet and EIP
- NAT Gateway status: Available



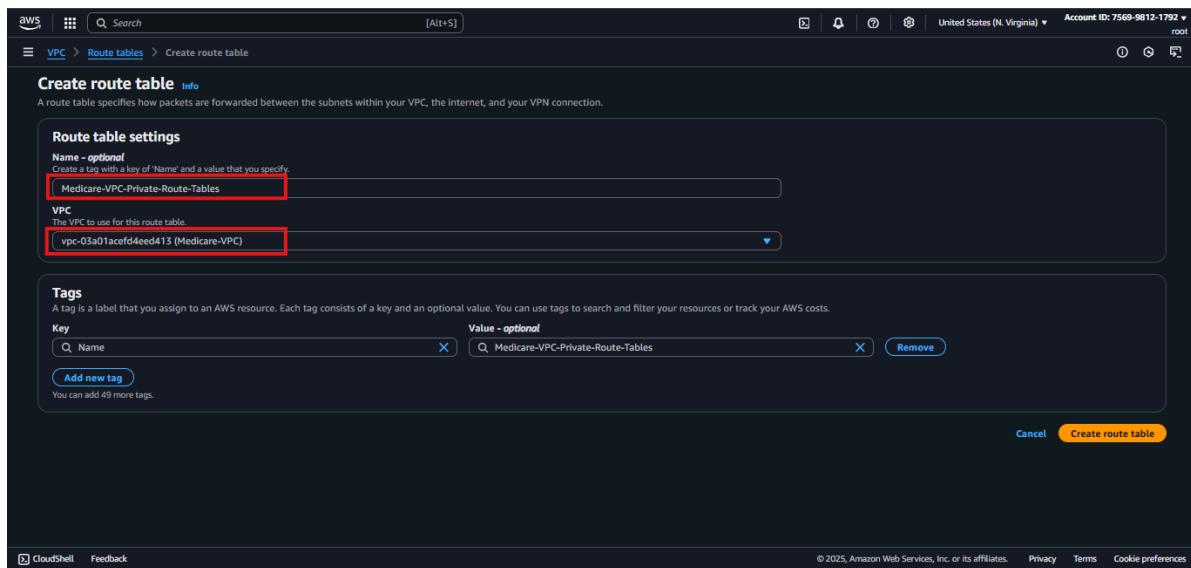
The screenshot shows the AWS VPC dashboard with the 'NAT gateways' section selected. There are two entries in the table:

Name	NAT gateway ID	Connectivity...	State	Primary public IP...	Primary private I...
Public-NAT-Gateway-Private-Subnet-B	nat-0984d09e9fdda03809	Public	Available	18.210.96.219	10.0.4.40
Public-NAT-Gateway-Private-Subnet-A	nat-08cb6031cf0577086	Public	Available	52.72.123.78	10.0.2.76

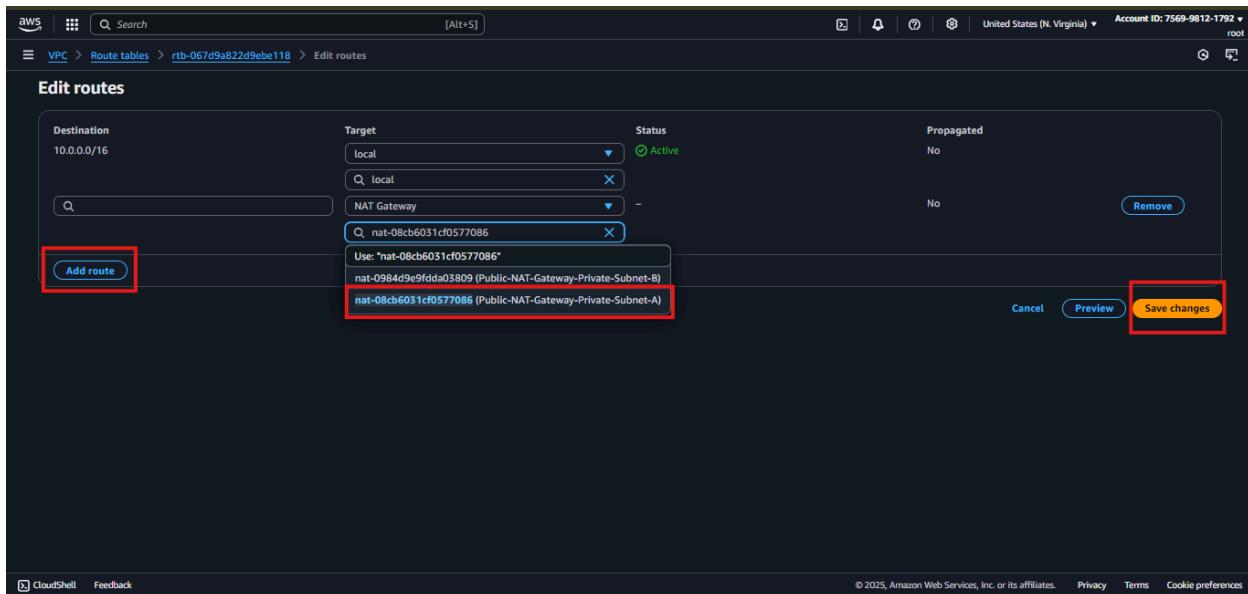
The rows for both NAT gateways are highlighted with red boxes, and the 'Available' status in the 'State' column is also highlighted.

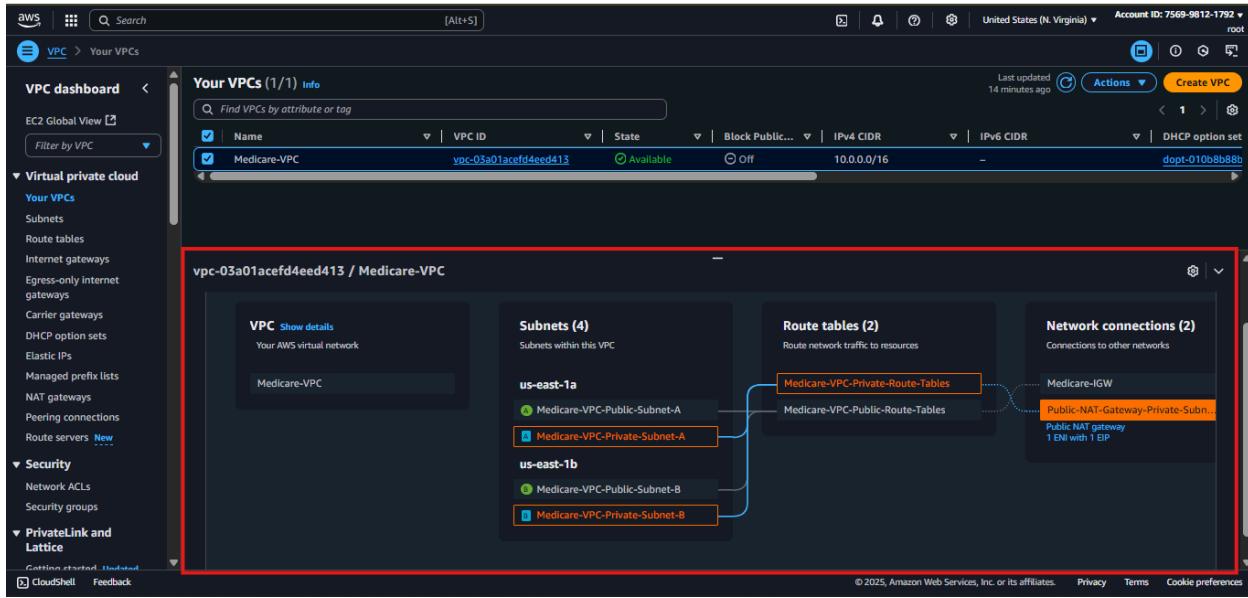
8.3 Create Route Tables for Private Subnets

- Navigate to **Route Tables**
- Select the **private route table**
- Add a route:
 - Destination: 0.0.0.0/0
 - Target: NAT Gateway ID



- Private route table before and after editing
- Route to NAT Gateway confirmed





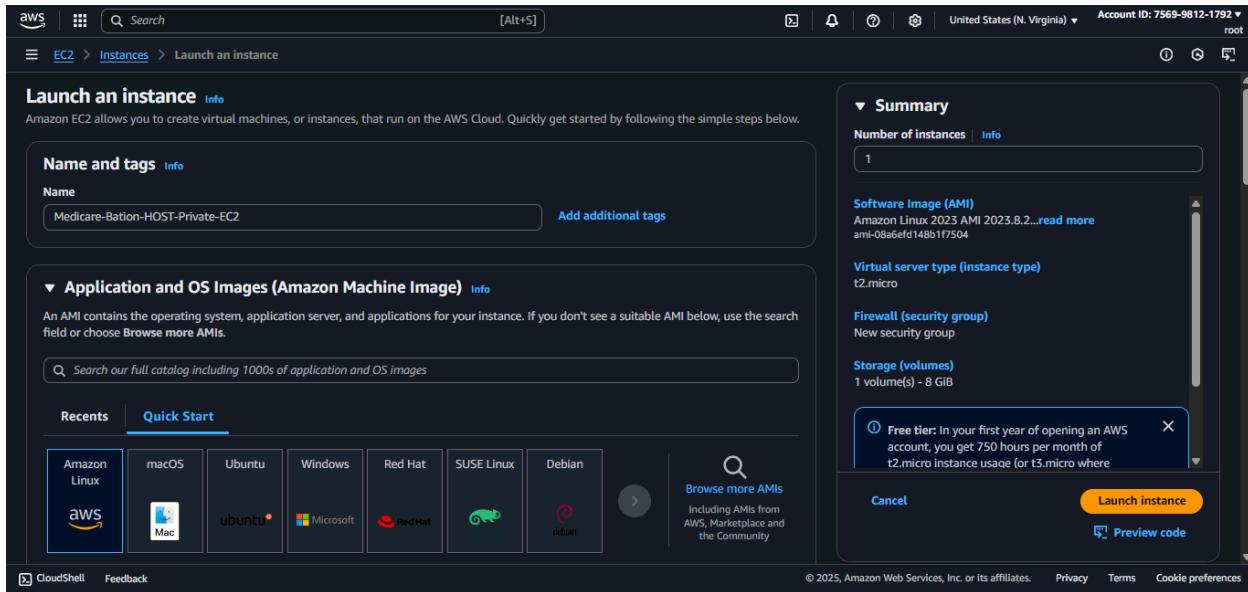
Step 9: Bastion Host Setup for Secure Private Access

To enable secure SSH access to EC2 instances in the **private subnets**, we configured a **Bastion Host** (also called a jump box) in a public subnet.

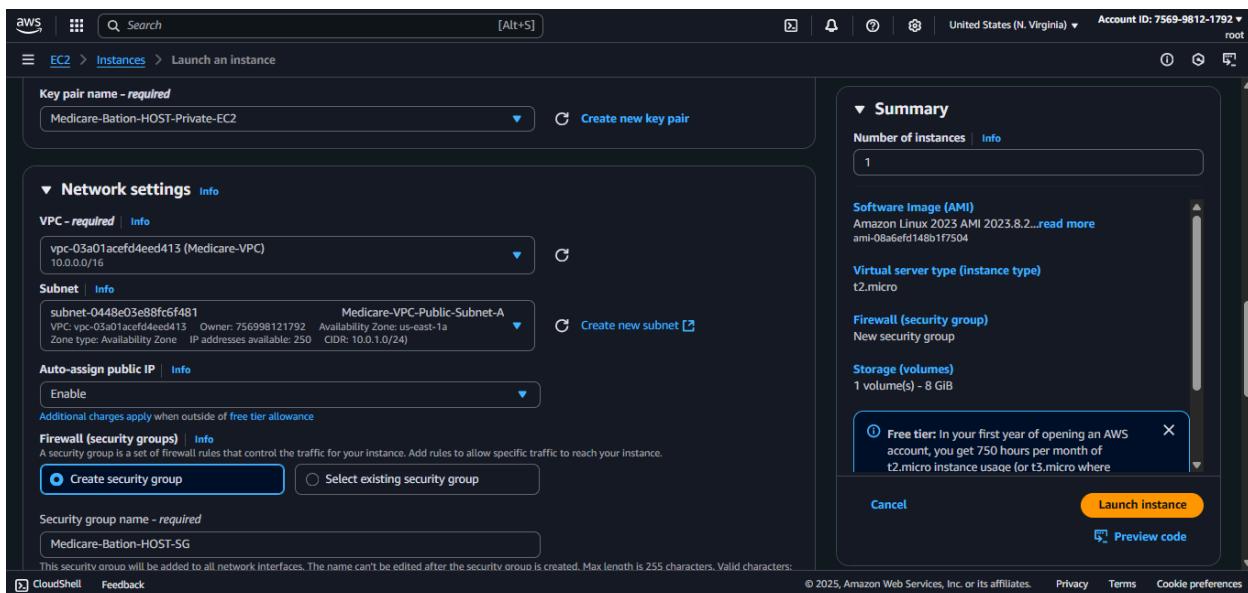
This host acts as an intermediary instead of giving public IPs to private EC2s, we connect to the Bastion and then hop into the private EC2s using internal IP addresses.

9.1 Launch Bastion EC2 in Public Subnet

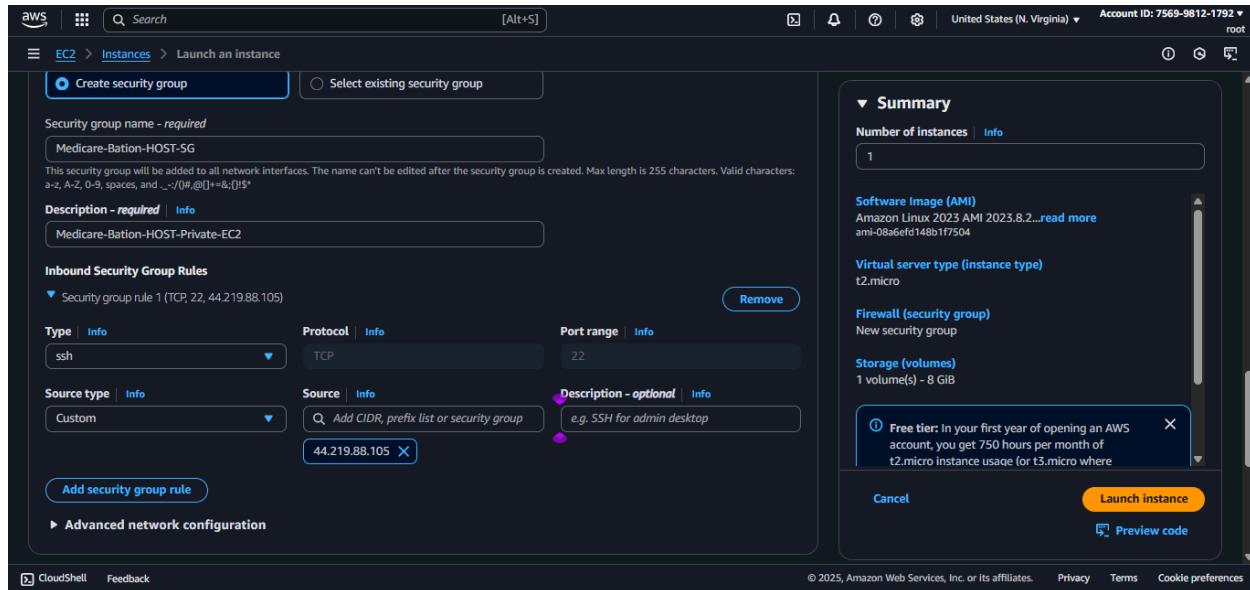
- Go to EC2 → Launch Instance
- Select Amazon Linux 2 (t2.micro for free tier)



- Place in **Public Subnet A**
- Enable **Auto-assign Public IP**
- Attach the same key pair used for your private EC2s



- EC2 launch config (with subnet and public IP enabled)



9.2 SSH into Bastion Host

From a Regular host to Test the security

- We got rejected when we tried to login from a host that was not specified in the security group even with the ssh key

```
chinazor@20.254.67.68:22 - Bitvise xterm - chinazor@personal-almalinux:~  
Last login: Sun Jul 27 01:31:41 2025 from 102.89.47.68  
[chinazor@personal-almalinux ~]$ sudo ssh -i "bastion.pem" ec2-user@44.215.73.40  
  
ssh: connect to host 44.215.73.40 port 22: Connection timed out  
  
You can find some explanations for typical errors at this link:  
https://red.ht/support_rhel_ssh  
[chinazor@personal-almalinux ~]$  
[chinazor@personal-almalinux ~]$  
[chinazor@personal-almalinux ~]$
```

From the Specified HOST in the Security Group

We use MobaXterm or SSH terminal with my `.pem` key:

```
sudo ssh -i "bastion.pem" ec2-user@44.215.73.40
```

Screenshots to Include:

- MobaXterm or terminal connected to Bastion
- Confirmed SSH session

9.3 Allow SSH to Private EC2s from Bastion

The screenshot shows the AWS EC2 Security Groups console. On the left, there's a navigation sidebar with options like Dashboard, Instances, Images, Elastic Block Store, and Network & Security. Under Network & Security, 'Security Groups' is selected. In the main area, a security group named 'sg-0da1dc27382d3e8d - Medicare-VPC-Private-Subnet-SG' is displayed. The 'Inbound rules' tab is selected, showing two entries. The second entry has its source IP address, '44.215.73.40/32', highlighted with a red box. The table columns include Security group rule ID, IP version, Type, Protocol, Port range, Source, and Description.

Security group rule ID	IP version	Type	Protocol	Port range	Source	Description
sgr-028b3813ddd27ea85	IPv4	SSH	TCP	22	44.215.73.40/32	-
sgr-06239bc7c9bfa77bc	IPv4	MySQL/Aurora	TCP	3306	0.0.0.0/0	-

Update the **Security Group** of the private EC2s:

- Allow inbound SSH (port 22) **only from the Bastion's SG or IP** (`44.215.73.40`)
- Private EC2 SG with SSH from Bastion's SG

9.4 SSH Into Private EC2 From Bastion Host

The screenshot shows the AWS EC2 Instances page. On the left, there's a navigation sidebar with sections like Images, Elastic Block Store, Network & Security, Load Balancing, and Auto Scaling. The main area displays a table of instances. One instance, 'Medicare-Private-EC2-A', is selected and highlighted with a red box. The instance details panel on the right shows the following information:

AMI ID	Monitoring	Platform details
ami-08a6ef148b1f7504	disabled	Linux/UNIX
AMI name	Allowed image	Termination protection
al2023-ami-2023.8.20250721.2-kernel-6.1-x86_64	-	Disabled
Stop protection	Launch time	AMI location
Disabled	Sat Jul 26 2025 19:26:34 GMT+0100 (West Africa Standard Time) (about 8 hours)	amazon/al2023-ami-2023.8.20250721.2-kernel-6.1-x86_64

Once inside the Bastion:

```
ssh ec2-user@10.0.2.XX # Private EC2-A  
ssh ec2-user@10.0.4.XX # Private EC2-B
```

📸 Screenshots to Include:

- Terminal inside Bastion connecting to private EC2s
- Success login prompt on private EC2

This setup ensures that no private EC2s are directly exposed to the internet, while still maintaining admin access when needed.

✓ Final Outcome

Goal	Status
Fully functional VPC with 2 AZs	✓ Completed
Redundant web servers in separate AZs	✓ Completed

Goal	Status
Secure networking with SGs and NACLs	<input checked="" type="checkbox"/> Completed
Live web server setup on both EC2s	<input checked="" type="checkbox"/> Completed
Domain mapped via Route 53 with failover	<input checked="" type="checkbox"/> Completed
DNS failover tested with shutdown event	<input checked="" type="checkbox"/> Completed

Submission Checklist

Item	Status
.drawio VPC diagram	<input checked="" type="checkbox"/> Included
At least 3 screenshots/step	<input checked="" type="checkbox"/> Captured
DNS + Route 53 setup visuals	<input checked="" type="checkbox"/> Captured
PDF/Word doc of this file	<input checked="" type="checkbox"/> Ready
.pem key (not uploaded)	<input checked="" type="checkbox"/> Secured