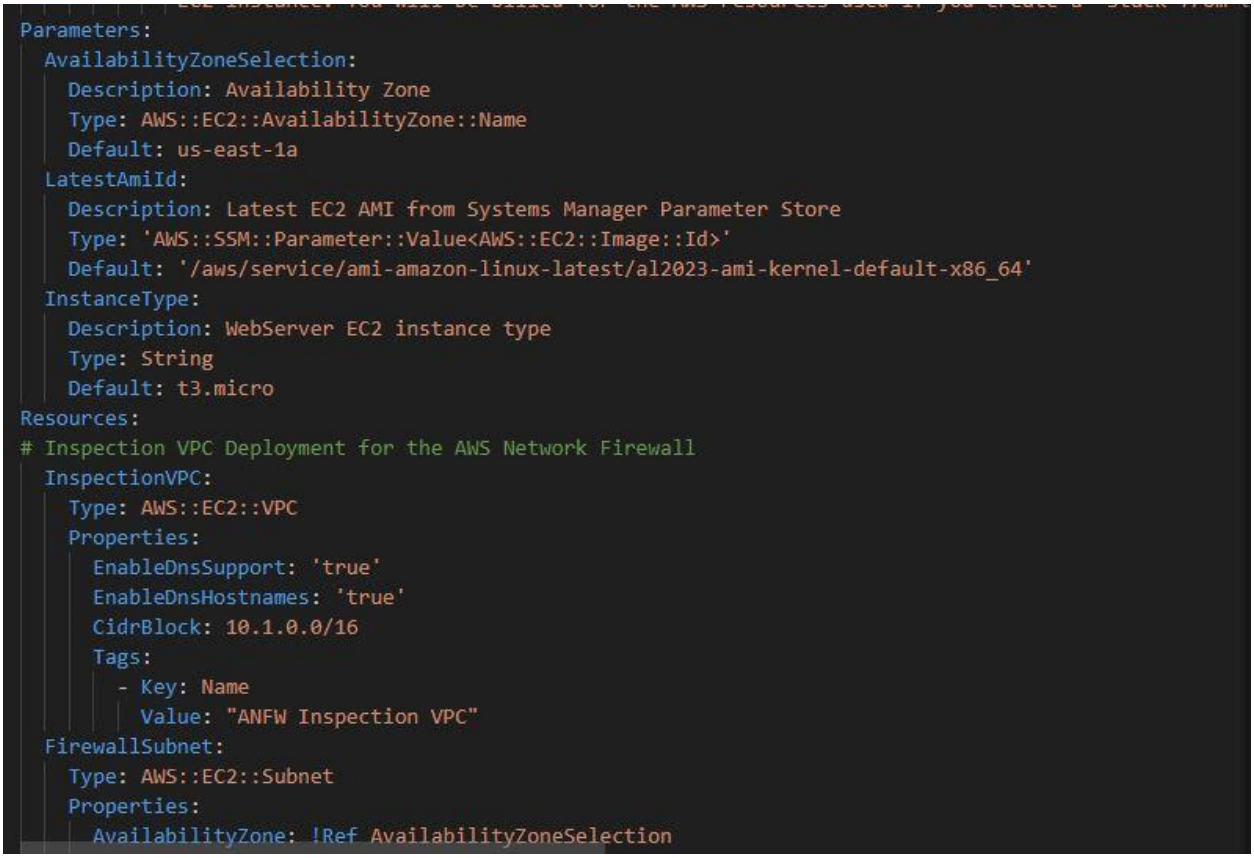
Cloud security continues to be a critical focus for organizations, since most organizations are moving their infrastructure to the cloud. With the growing importance of secure network architectures in the cloud, AWS Network Firewall (ANFW) offers robust protection to control traffic within an AWS environment. Recently, I completed Labs 1 and 2 of the AWS Hands-on Network Firewall Workshop. This documentation documents my journey through setting up and configuring a protected VPC with public and private workloads, providing insights into AWS Network Firewall's powerful capabilities.

Setting Up the Lab Environment for Lab 1.

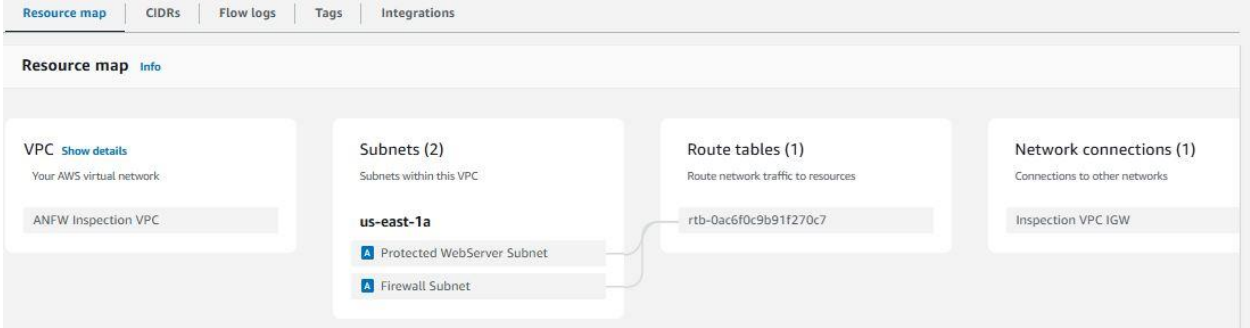
I used CloudFormation to create infrastructure needed for this lab. This template created vpc, firewall subnet, protected webserver subnet, internet gateway, webserver and an EC2 instance. The image below shows code sample from template.



```
Parameters:
  AvailabilityZoneSelection:
    Description: Availability Zone
    Type: AWS::EC2::AvailabilityZone::Name
    Default: us-east-1a
  LatestAmiId:
    Description: Latest EC2 AMI from Systems Manager Parameter Store
    Type: 'AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>'
    Default: '/aws/service/ami-amazon-linux-latest/al2023-ami-kernel-default-x86_64'
  InstanceType:
    Description: WebServer EC2 instance type
    Type: String
    Default: t3.micro
Resources:
# Inspection VPC Deployment for the AWS Network Firewall
  InspectionVPC:
    Type: AWS::EC2::VPC
    Properties:
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
      CidrBlock: 10.1.0.0/16
      Tags:
        - Key: Name
          Value: "ANFW Inspection VPC"
  FirewallSubnet:
    Type: AWS::EC2::Subnet
    Properties:
      AvailabilityZone: !Ref AvailabilityZoneSelection
```

Lab 1: Protected VPC with Public Workload.

The resource map after the the template was loaded is as shown below:

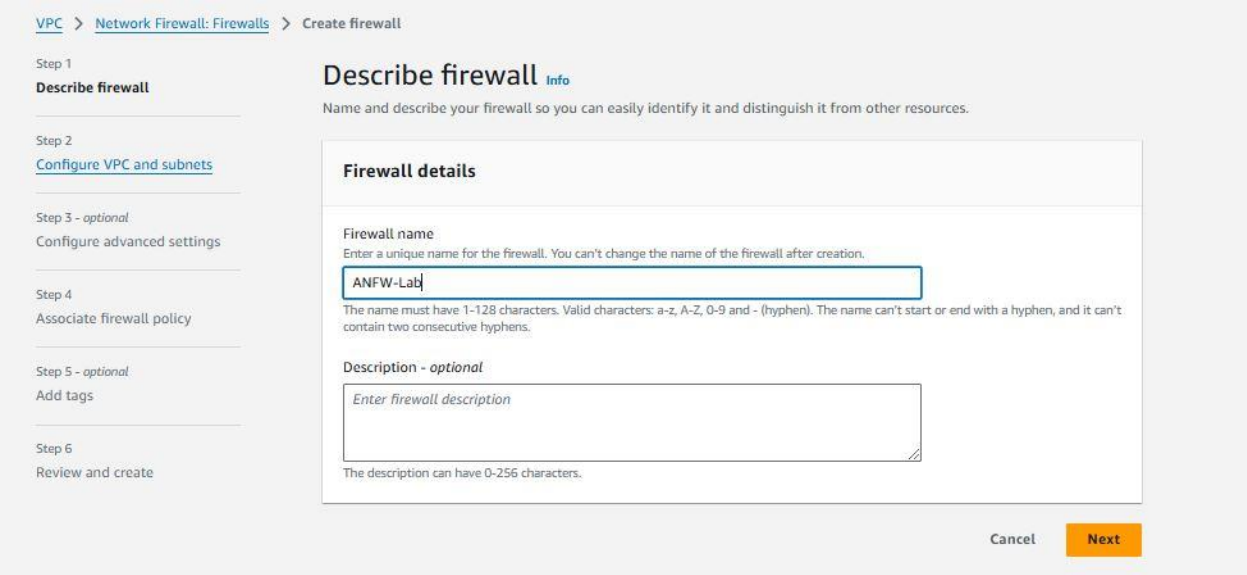


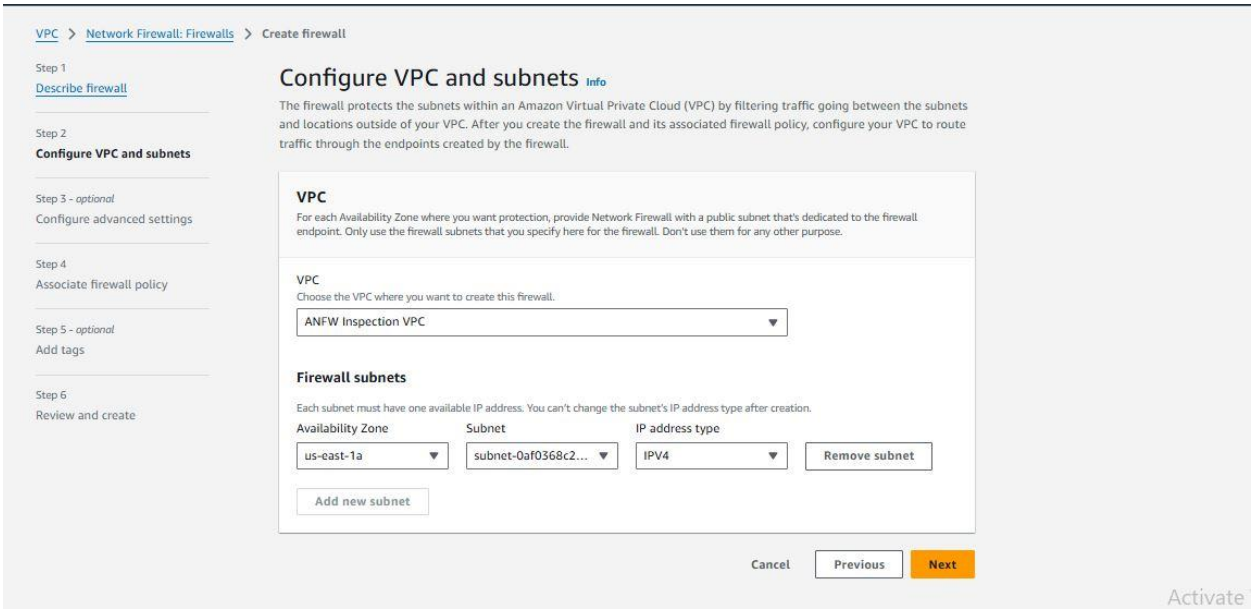
I applied firewall rules to control traffic for the configured vpc. The goal was to set up a basic firewall for a publicly accessible web server, log network activity, and configure initial route tables.

The steps taken were:

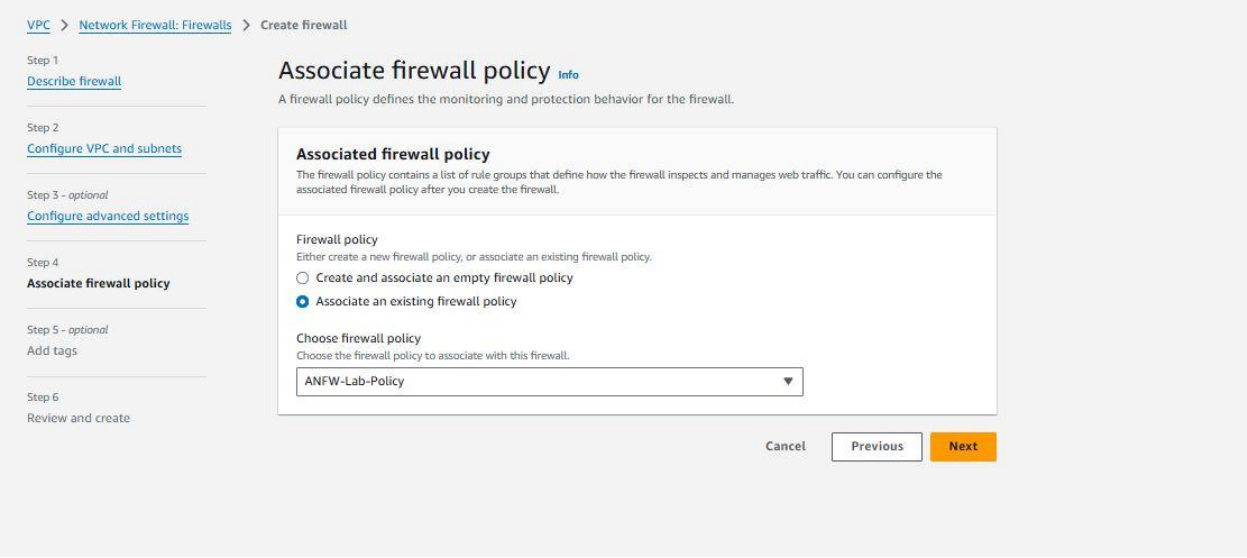
1.Creating a firewall.

I created an AWS Network Firewall and attached it to the firewall subnet, as shown below:





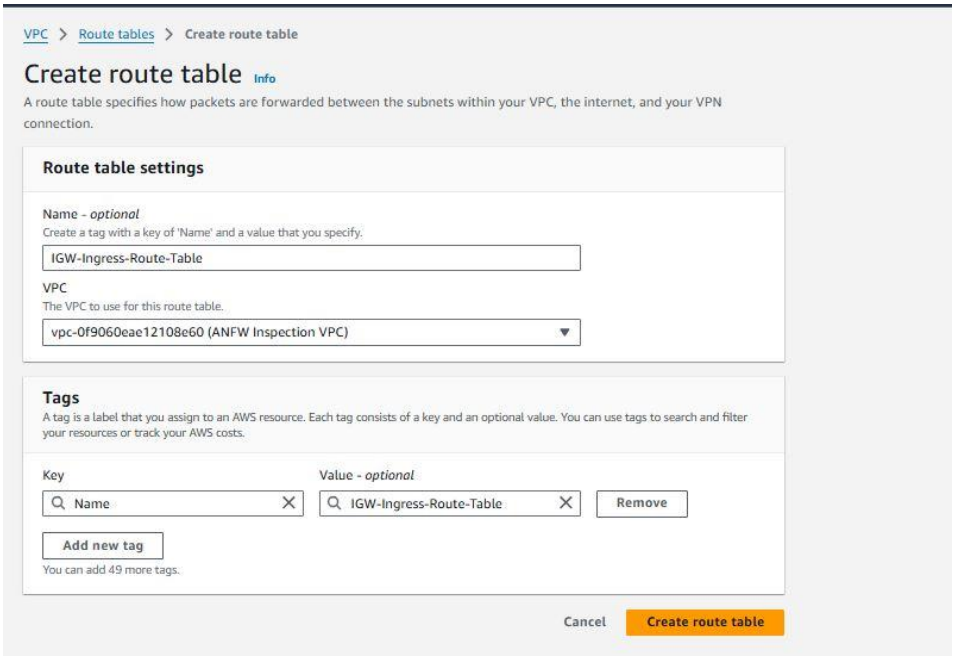
Then I associated an existing firewall policy that I had previously created, to the firewall. All the other setting I left it as default, then I reviewed and created the firewall.



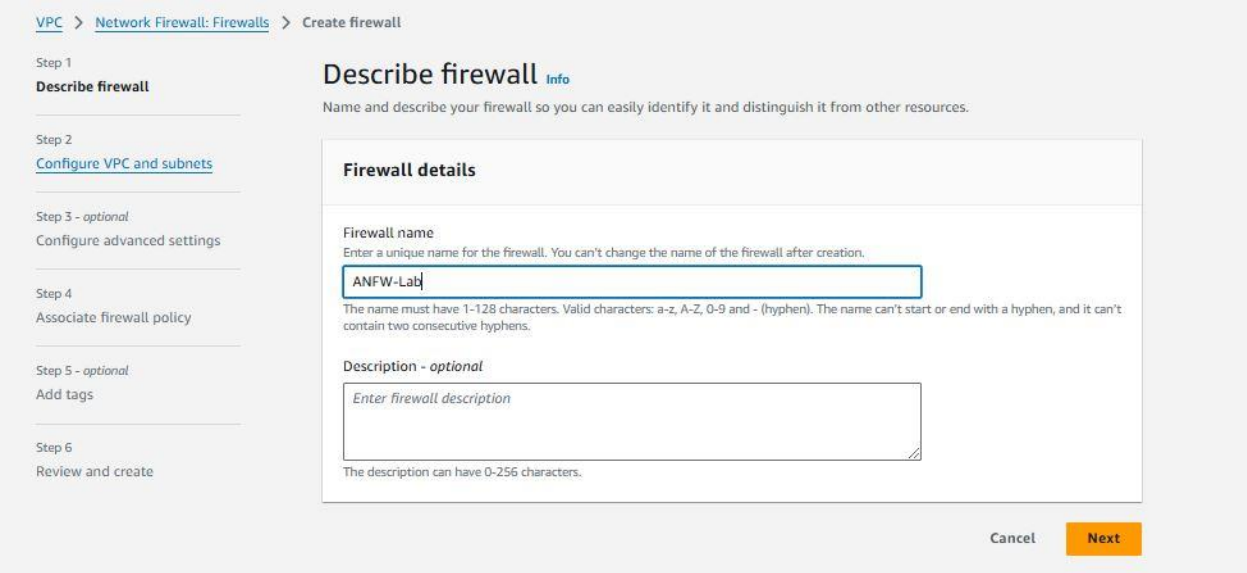
2.Creating Route tables.

I created three route tables to direct traffic to and from my NGINX web server through the firewall for inspection. Traffic will flow through the firewall subnet where the firewall policies can be applied before the request reaches the server.

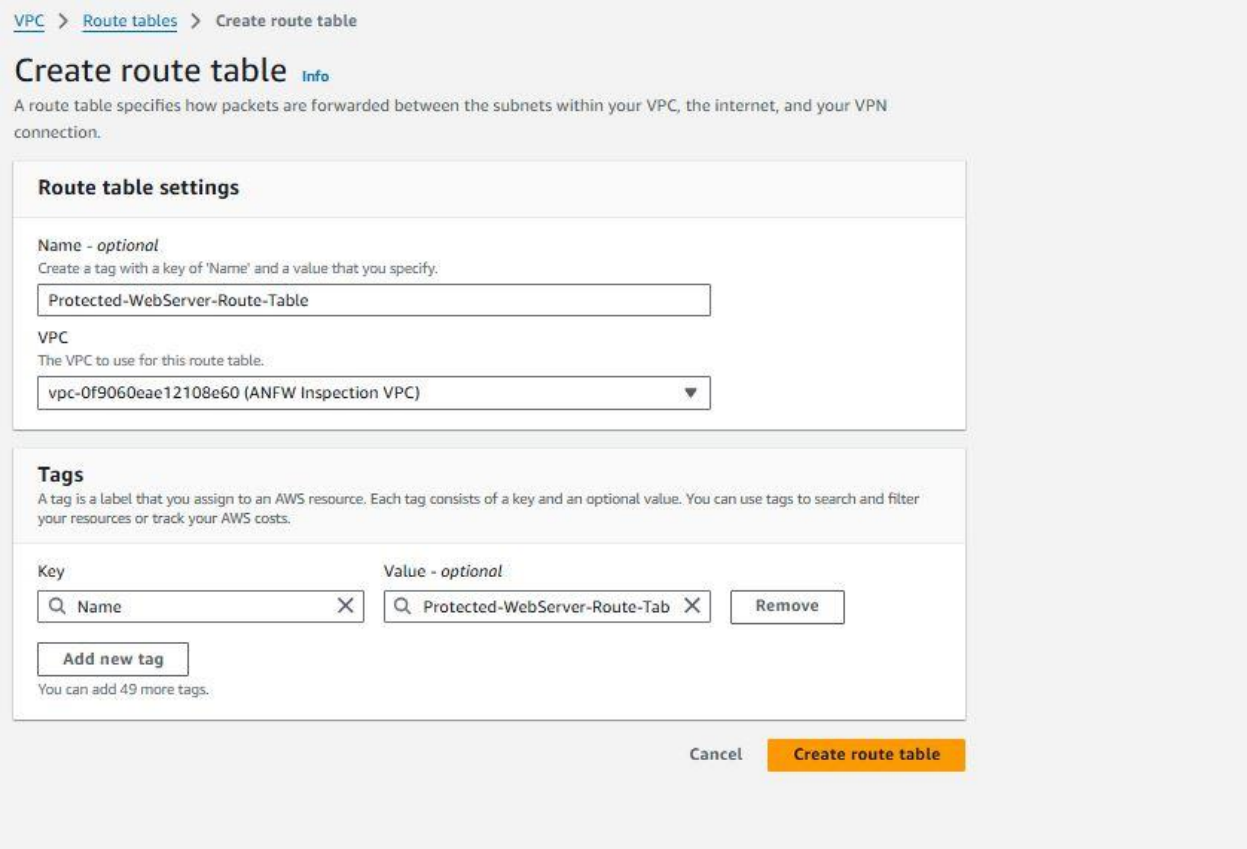
The IGW Ingress route table is used to ensure traffic coming in to the VPC gets routed to the network firewall first and then the firewall can make a determination on what to do with that traffic.



The **Firewall Route Table**. This route table will control flow from the firewall subnet to the rest of the VPC as well as all other traffic (the default route) going to the Internet Gateway (IGW).

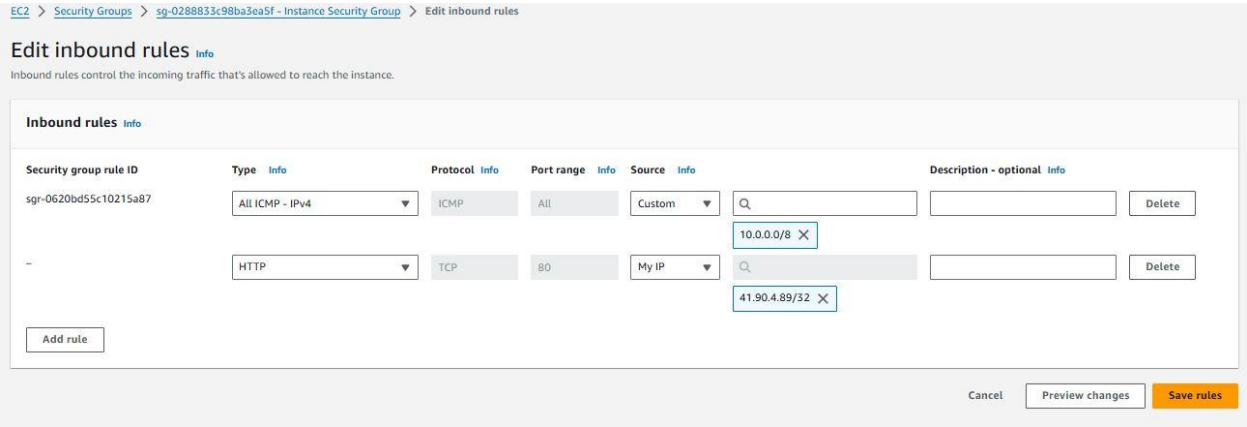


I created the **Protected WebServer Route Table**. This route table directed all traffic from the protected workflow subnet bound for anywhere outside of the VPC to the firewall endpoint so that outbound requests can also be processed by the firewall.

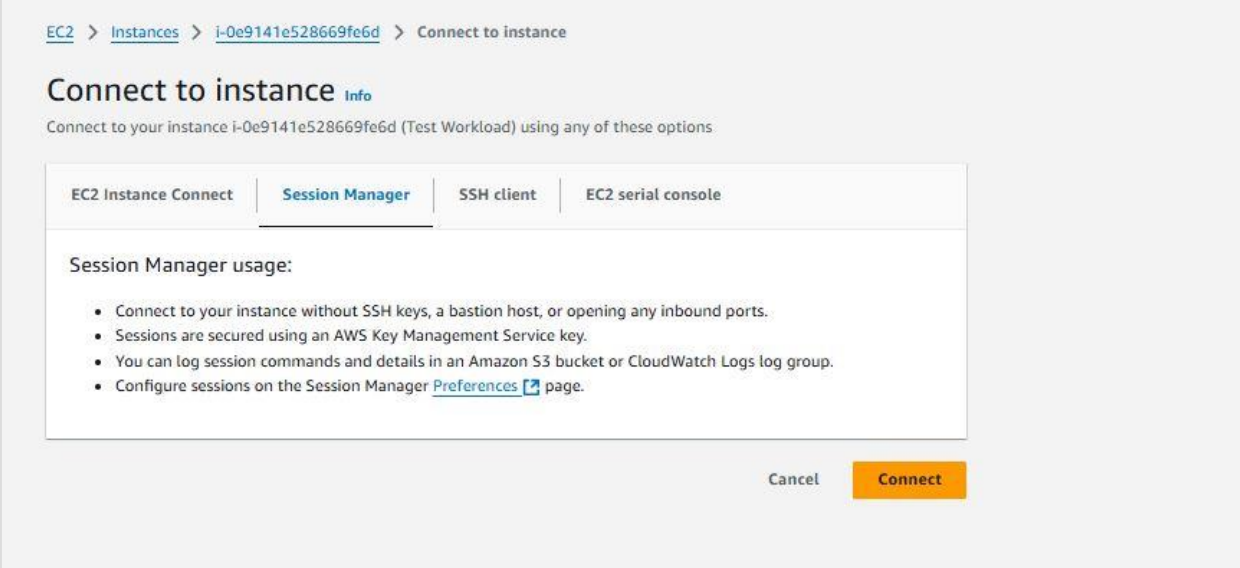


Connect to Ec2 and enable the webserver.

I added a rule in the pre-configured security group to allow http traffic from my IP address.



I then connected to my EC2 instance via the session manager.



Typed the following code after it opened, to install NGINX to accept HTTP requests and test ingress web server traffic to my Test Workload through the ANFW:

sudo dnf install nginx -y

sudo systemctl enable nginx

sudo systemctl start nginx

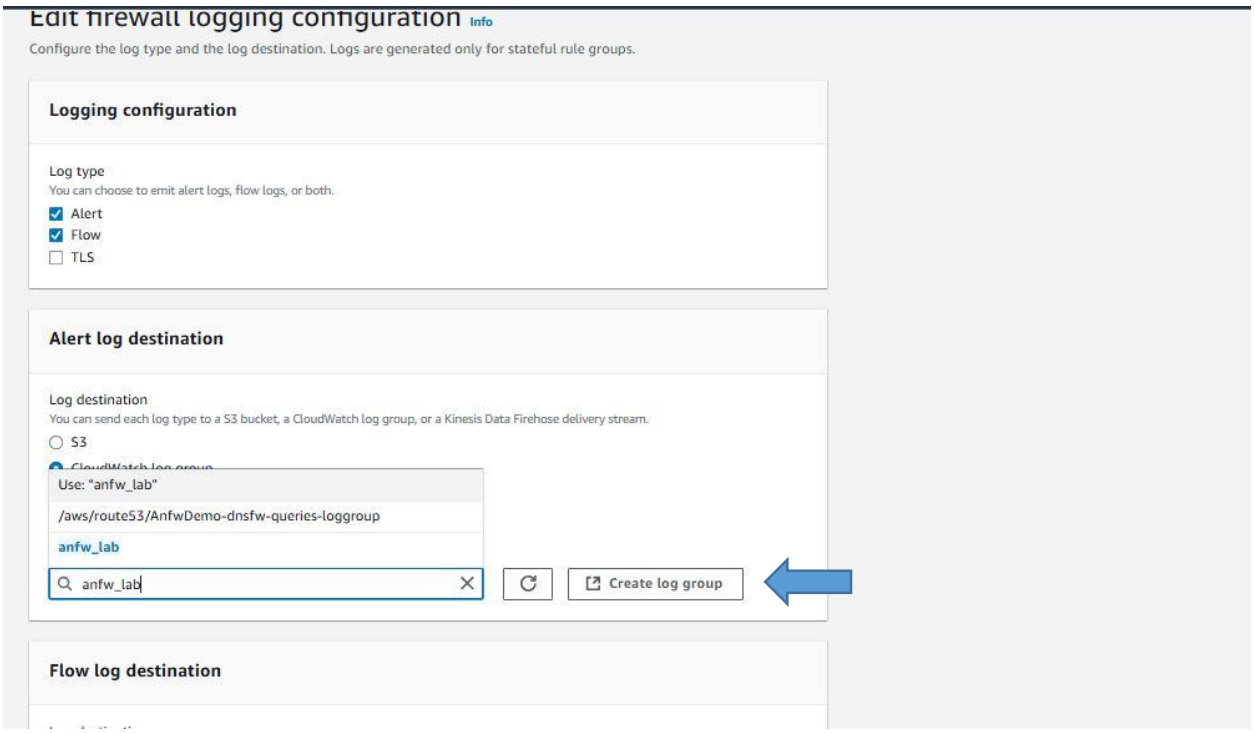
I then accessed my websever in a new tab using it’s public IP address and this was the output:

3. Set up firewall logging.

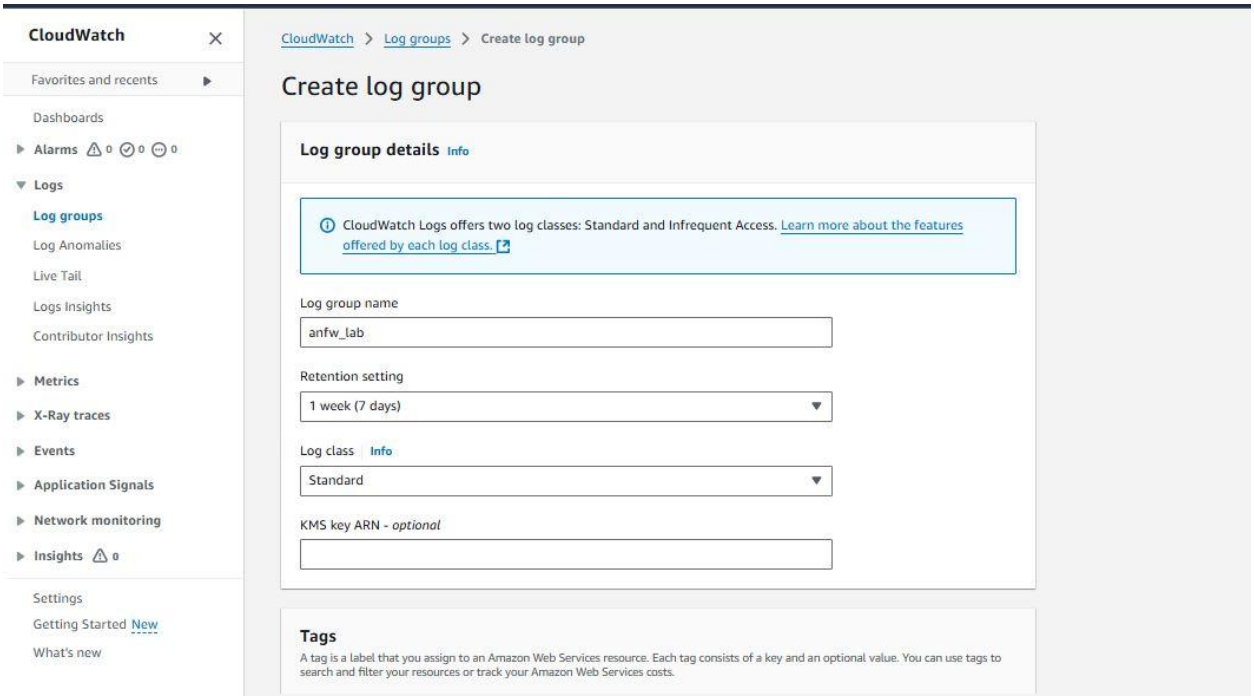
Now that all routes are configured and traffic is being inspected by the ANFW I set up logging to monitor ingress and egress traffic.

First navigate to the VPC Dashboard and select Firewalls in the left pane under Network Firewalls. Select the **Firewall Details** tab and scroll down to the **Logging** section, then click **Edit** as shown below:

Note: Set the flow log and alert log destination to cloudwatch logs.



On the Log groups page click on the **Create log group** button in the upper right hand corner from the tab shown on the diagram above:

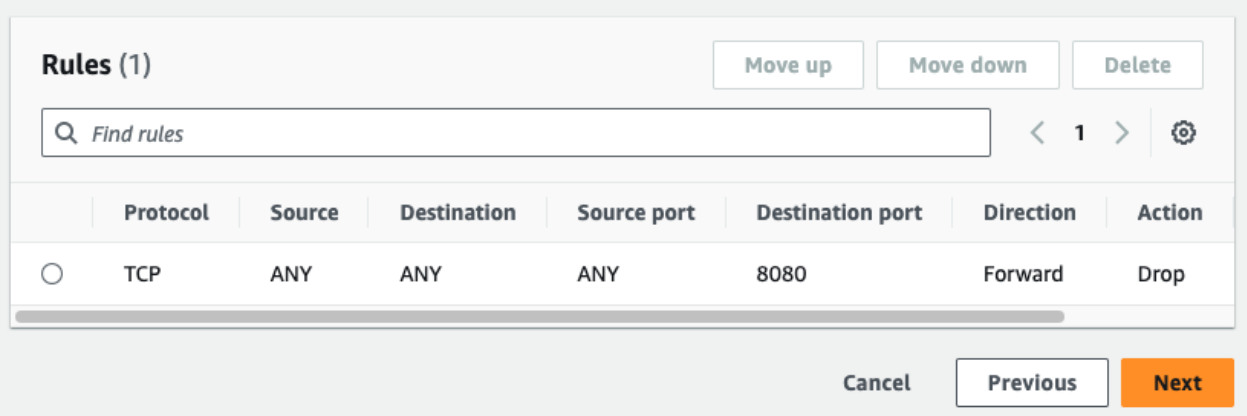


First, generate some flow logs to the Test Workload Web Server by navigating to it though a browser.Then check the log stream from cloudwatch.

4. Configure firewall policy

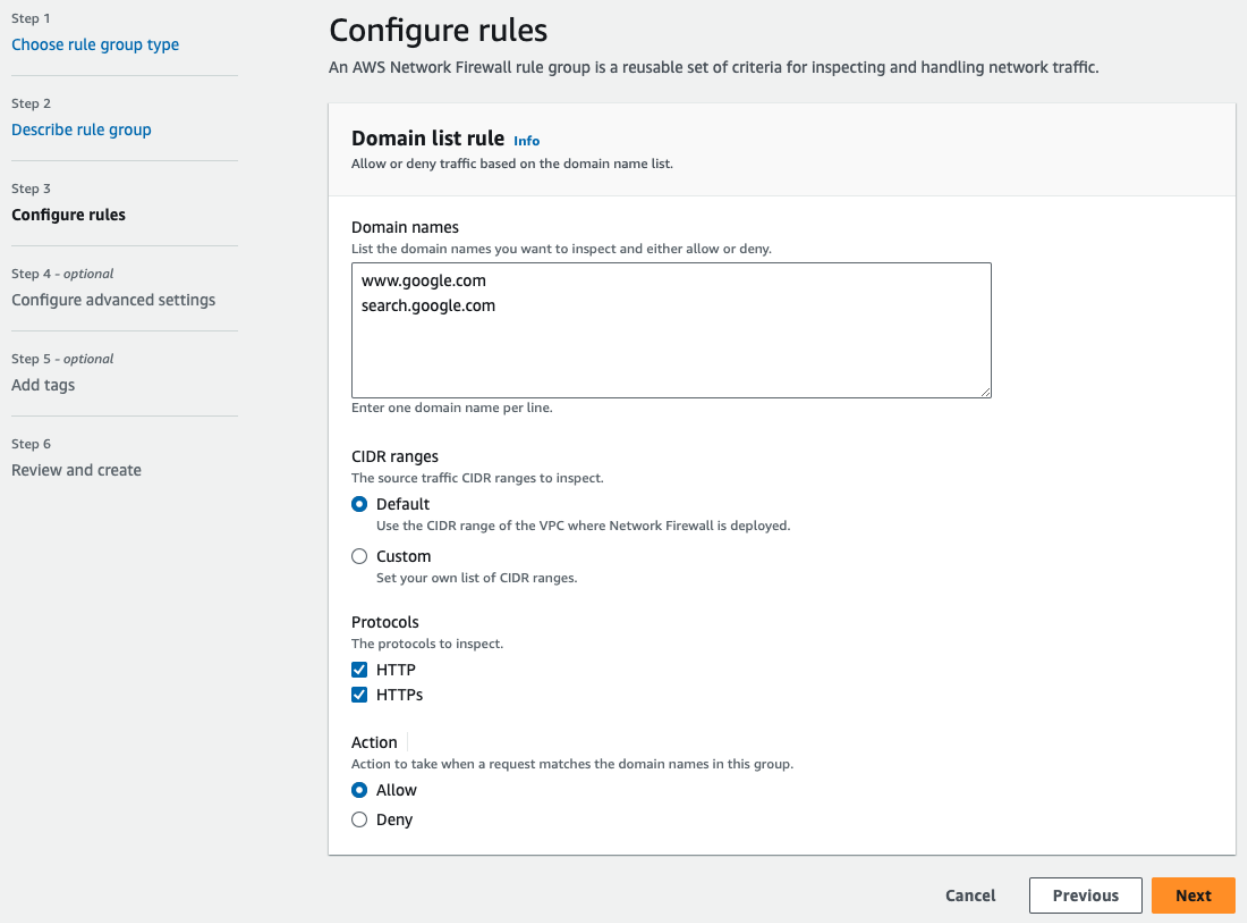
I created and test two types of stateful rule categories: standard and domain-list.

After creating the standard stateful rule, this is how it looked like:



I then generated some traffic to my web server on port 8080 and checked my **Alert** logs stream in the associated CloudWatch log group.

For stateful domain list, I configured the following:



This **Domain list** allows the domains added, and effectively denies all others except those in other rule groups associated with this policy.

Test the domain using the following command:

curl https://www.google.com --max-time 5

output:

```
sh-5.2$ curl http://www.google.com --max-time 5
<!doctype html><html itemscope="" itemtype="http://schema.org/WebPage" lang="en">
```

The output is successful since the domain name is specified in the approved list.One which is not approved is blocked as shown below:

```
sh-5.2$ curl http://google.com --max-time 5
curl: (28) Operation timed out after 5001 milliseconds with 0 bytes received
```

Lab 2: Protected VPC with Private Workload

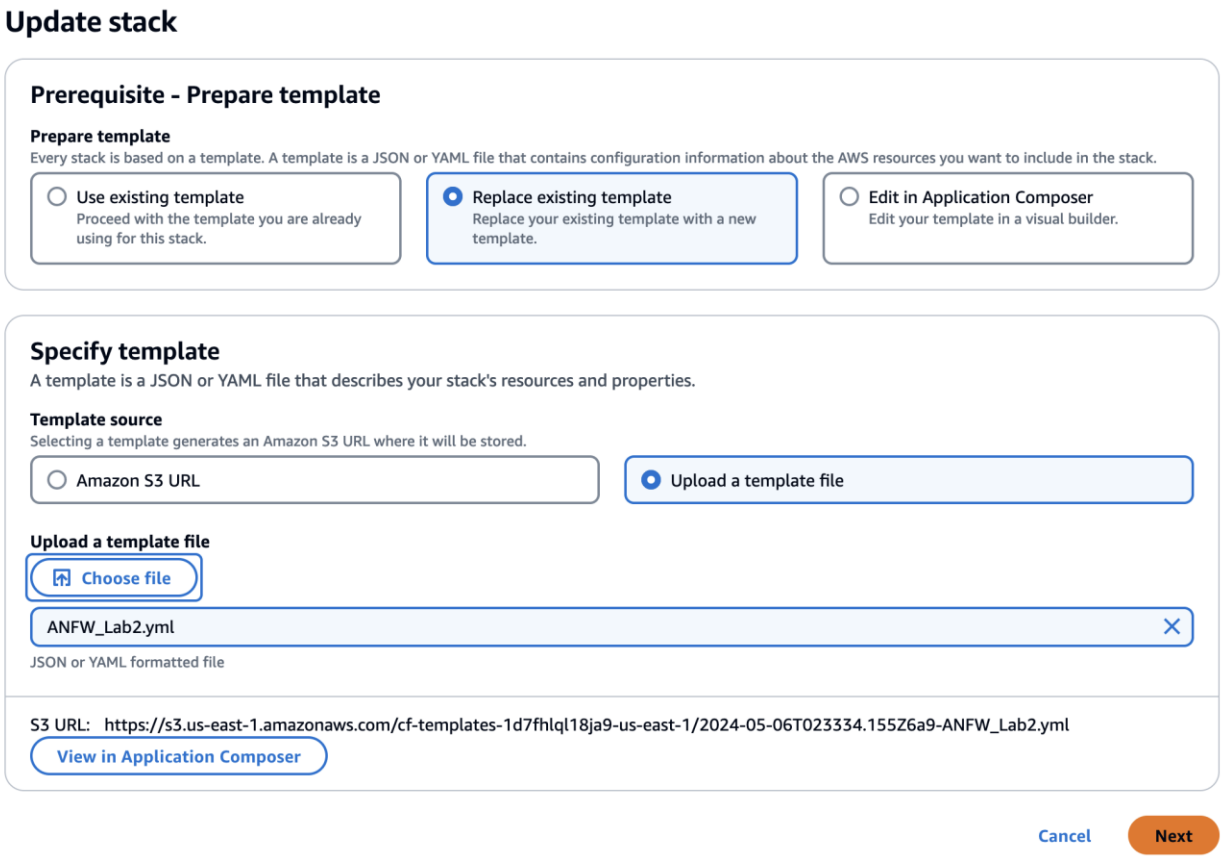
This lab focused on enhancing the security of the environment by transitioning the public workload into a protected private subnet and strengthening firewall policies to control access further.

Steps Taken:

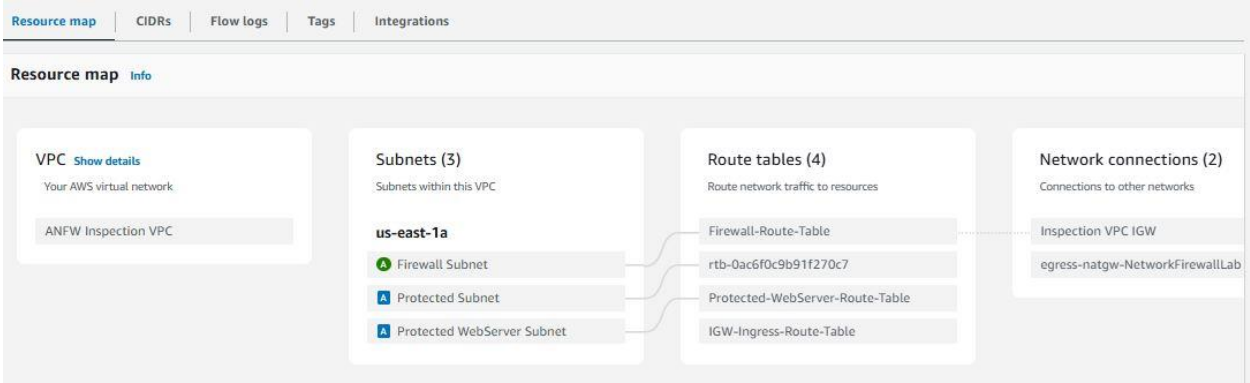
1.Setting Up the environment for lab 2.

Similar to Lab 1, the CloudFormation template set up the necessary infrastructure and resources, with a focus on updating the VPC configuration to support private access.

- Navigate to the CloudFormation console by following this link. and select the **NetworkFirewallLab** stack radio button. Once the stack is selected click the **Update** button to update the stack.



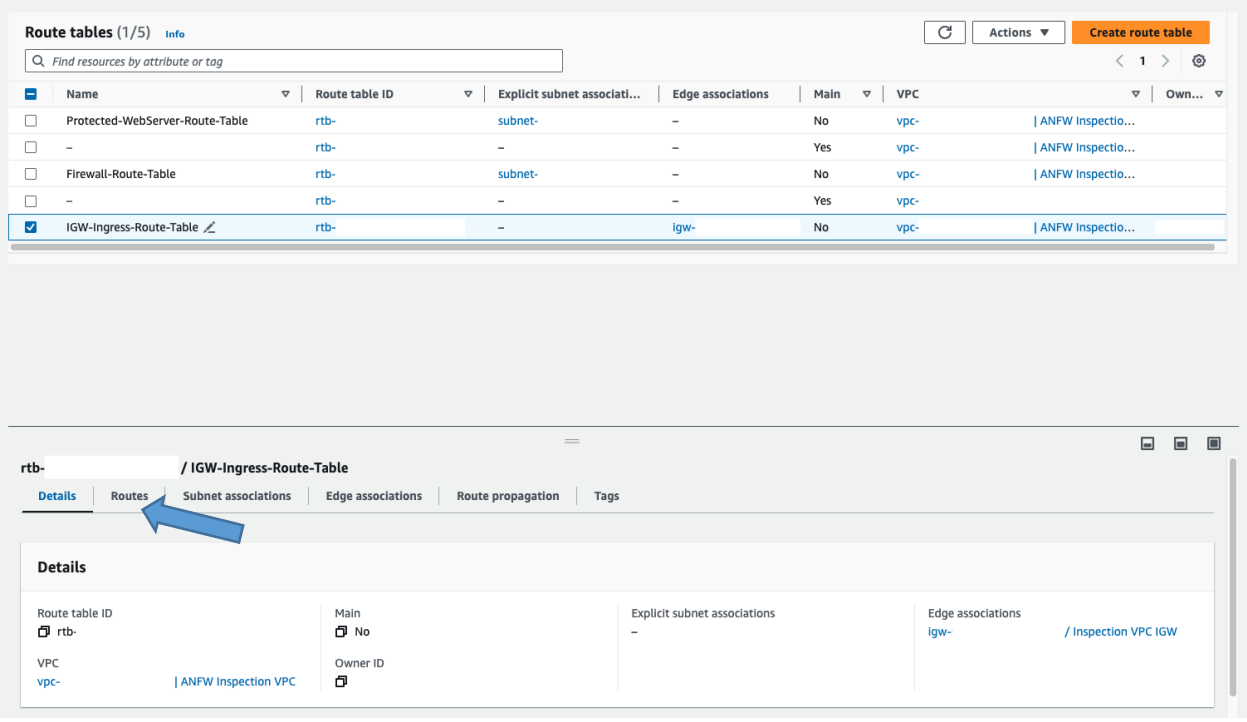
After it is updated, this is how the vpc resource map will look like:



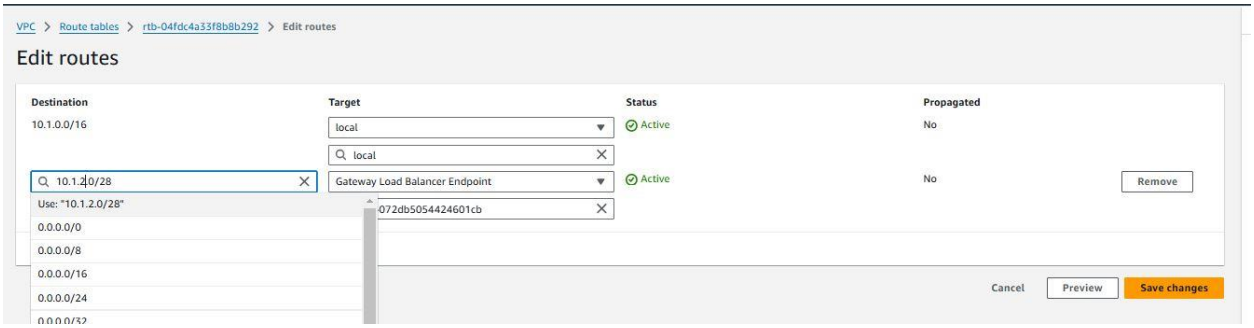
2. Update route tables.

I updated the IGW-Ingress-Route-Table as it now needs to point to the new Protected Subnet instead of the Protected Workload Subnet. This is so incoming traffic to the Network Load Balancer will be routed to the firewall.

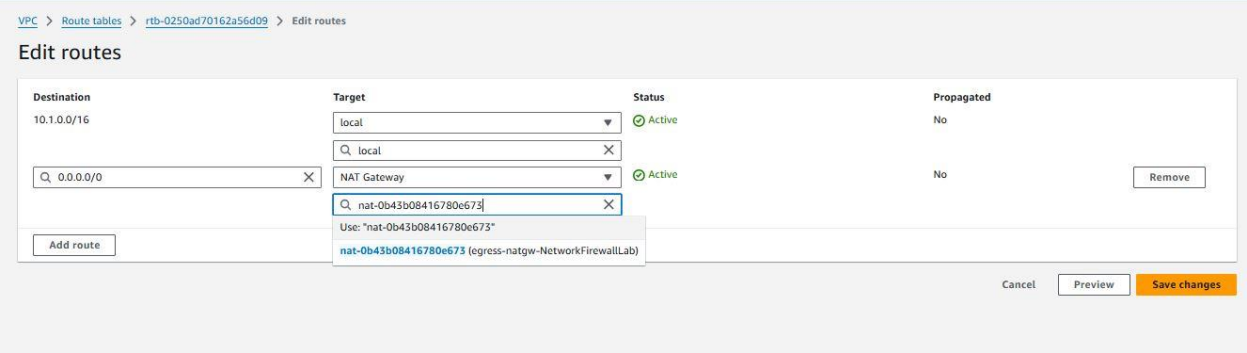
Select routes to modify it.



Change the destination as shown below:



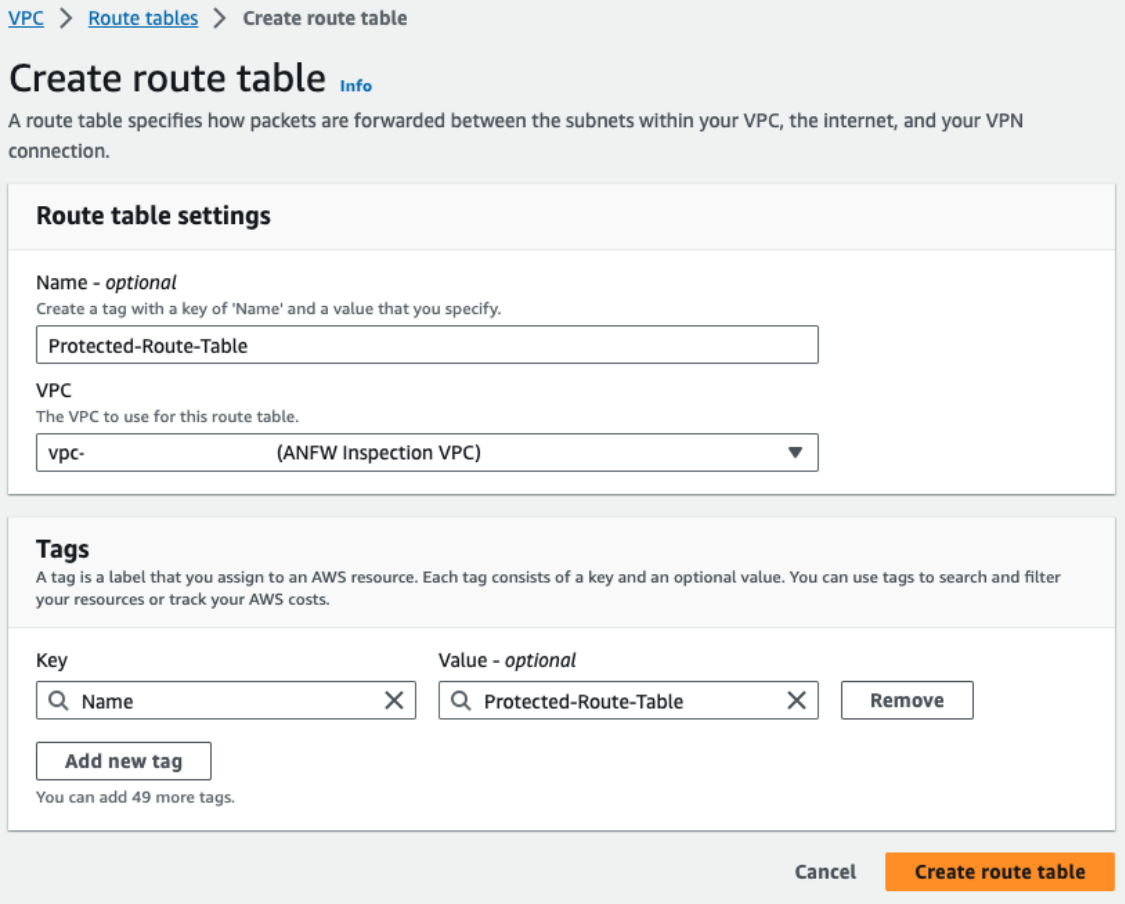
I update the **Protected-WebServer-Route-Table** to send all its Internet-bound traffic to the NAT Gateway in the protected subnet.

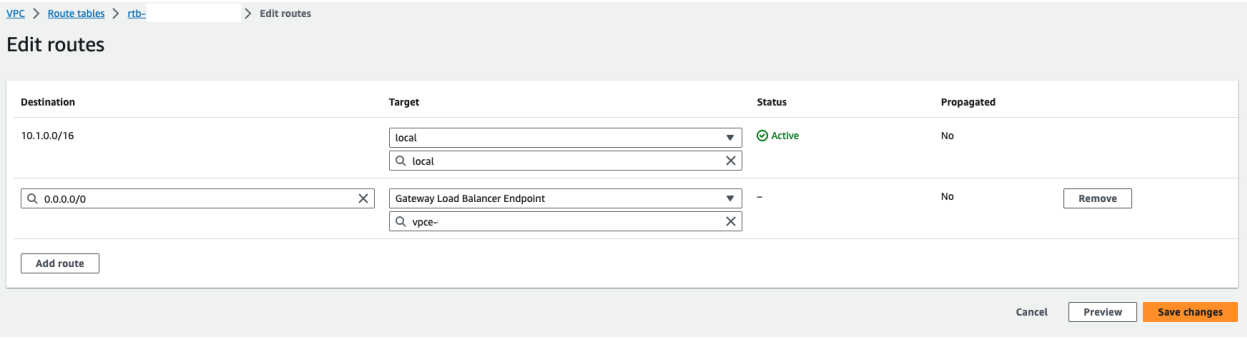


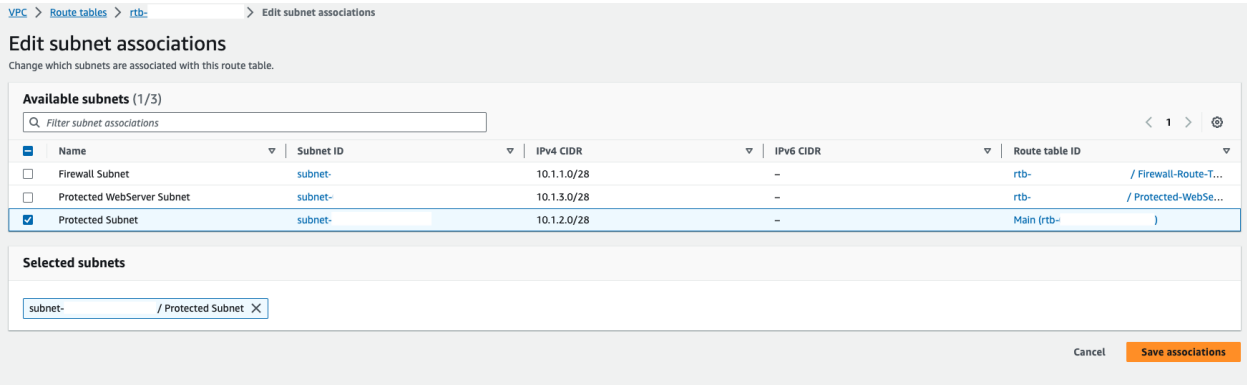
3. Create protected subnet routes.

From the Route tables screen, click on the Create route table button on the top left of the screen. On the Create route table form set the Name to be Protected-Route-Table.Select ANFW Inspection VPC from the Select a VPC dropdown. Then, click on the Create route table button, edit the routes and add a subnet association as shown below:

This is to allow inbound connectivity to the protected web server in the protected subnet through the Network Load Balancer. Outbound connectivity should be established through the NAT Gateway.


VPC > Route tables > Create route table
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.
Protected-Route-Table
VPC
The VPC to use for this route table.
vpc- (ANFW Inspection VPC)
Tags
A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.
Key
Value - optional
Name
Protected-Route-Table
Remove
Add new tag
You can add 49 more tags.
Cancel
Create route table

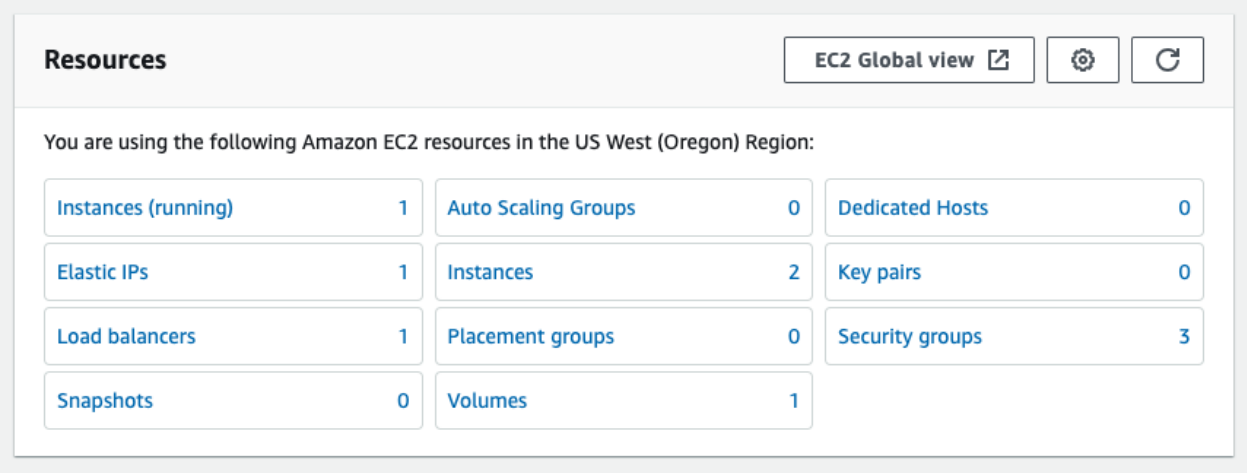

VPC > Route tables > rtb- > Edit routes
Edit routes
Destination
Target
Status
Propagated
10.1.0.0/16
local
local
Active
No
0.0.0.0/0
Gateway Load Balancer Endpoint
vpce-
–
No
Remove
Add route
Cancel
Preview
Save changes


VPC > Route tables > rtb- > Edit subnet associations
Edit subnet associations
Change which subnets are associated with this route table.
Available subnets (1/3)
Filter subnet associations
Name
Subnet ID
IPv4 CIDR
IPv6 CIDR
Route table ID
Firewall Subnet
subnet-
10.1.1.0/28
–
rtb-
/ Firewall-Route-T...
Protected WebServer Subnet
subnet-
10.1.3.0/28
–
rtb-
/ Protected-WebSe...
Protected Subnet
subnet-
10.1.2.0/28
–
Main (rtb-
)
Selected subnets
subnet-
/ Protected Subnet
Cancel
Save associations

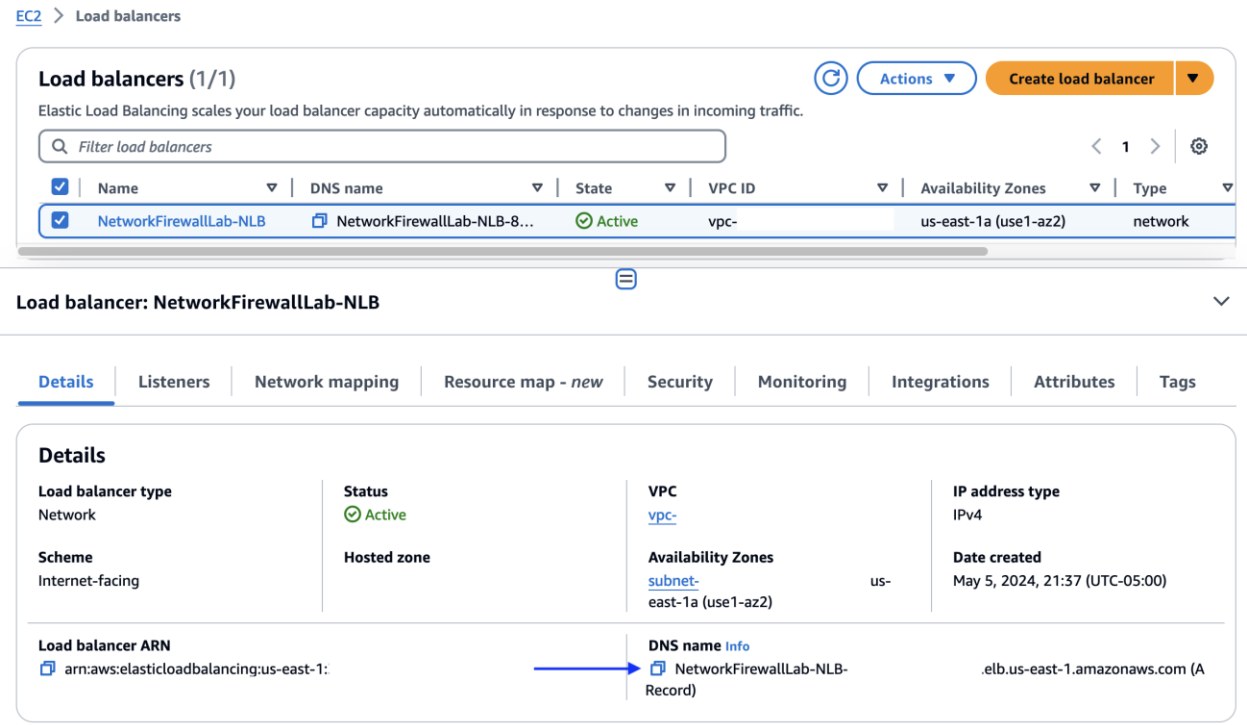
Testing the setup.

I tested the new setup with the private workload and confirm network connectivity via the Network Load Balancer (NLB).

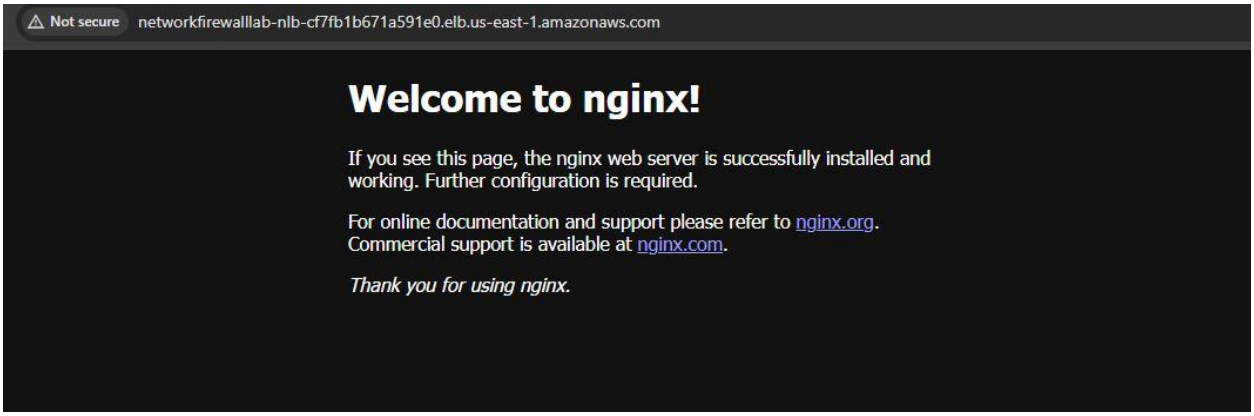
I navigated to to the EC2 Dashboard. Then in the Resources window, as seen below, I select Load balancers.



I selected the NLB for this lab and copied the **DNS name** from the **Description** tab in the configuration details as seen below.



I copied it on my web browser and got the output as shown below:



After that I checked the logs generated from my activity on cloudwatch.

I can validate the correct logs are delivered as the the destination IP address changed from 10.1.3.4 To that of my NLB, 10.1.2.4.

4.Strengthen Firewall Policy.

I configured both Stateless rules and Stateful IPS rules to strengthen the configured firewall policy in **Lab 1** which consisted of Stateful 5-tuple rules and Stateful Domain list rules only.

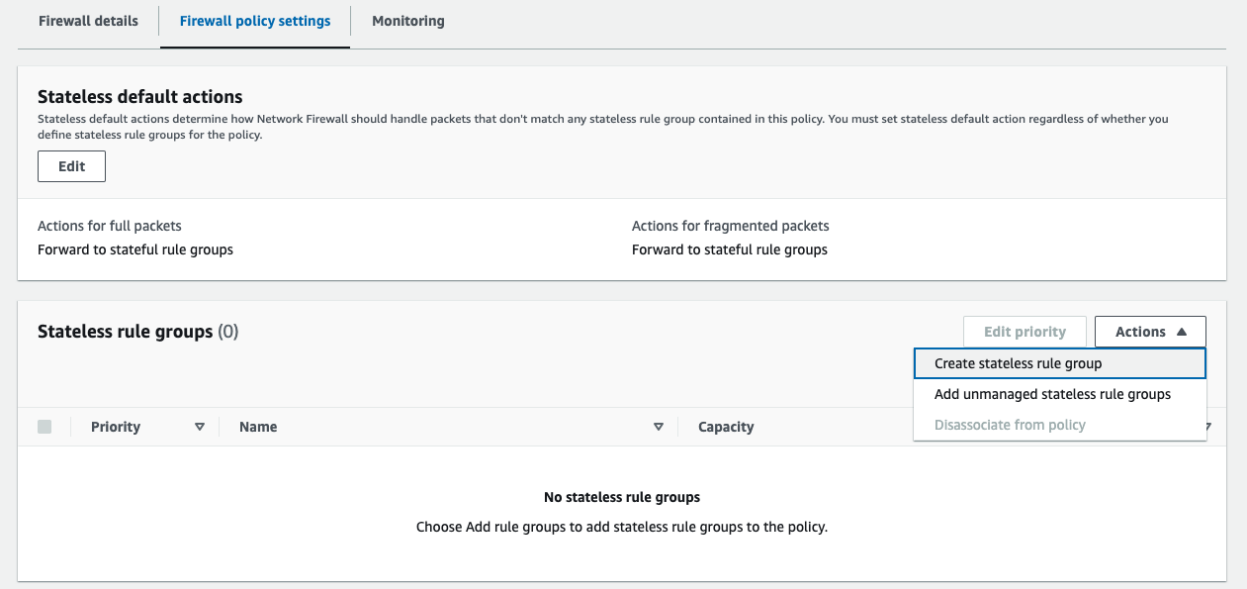
Stateless rule groups evaluate packets in isolation, while stateful rule groups evaluate them in the context of their traffic workflow.

- **Stateless** – Defines standard, 5-tuple criteria for examining a packet on its own, with no additional context.
- **Stateful Suricata-compatible IPS rules** – Defines intrusion prevention system (IPS) rules in the rule group in an open source compatible format (e.g. Suricata, Snort).

Benefit of combining rules: Stateless rules are evaluated first, and can provide a coarse layer of security protecting you from accidentally making Stateful rules too permissive. Stateful IPS network traffic inspection enables identification of malicious activity from protocol anomalies to malware and botnets using signature-based detection. AWS Network Firewall IPS rules use an open source Suricata syntax with thousands of existing community maintained rulesets available.

Configure stateless rules.

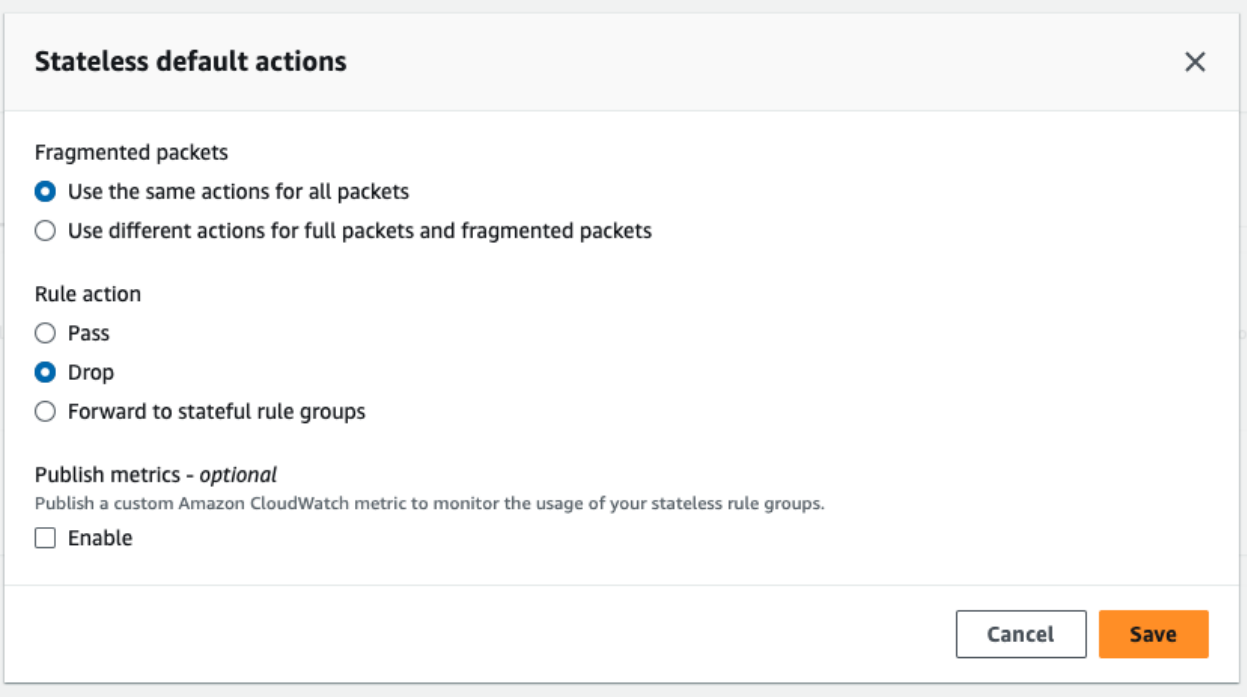
First navigate to VPC → Firewalls → *Your Lab Firewall*. This will bring you to the Firewall Overview page. Select the Firewall policy settings tab, then select Create stateless rule group in the Actions tab on the top right of the Stateless rule groups section.



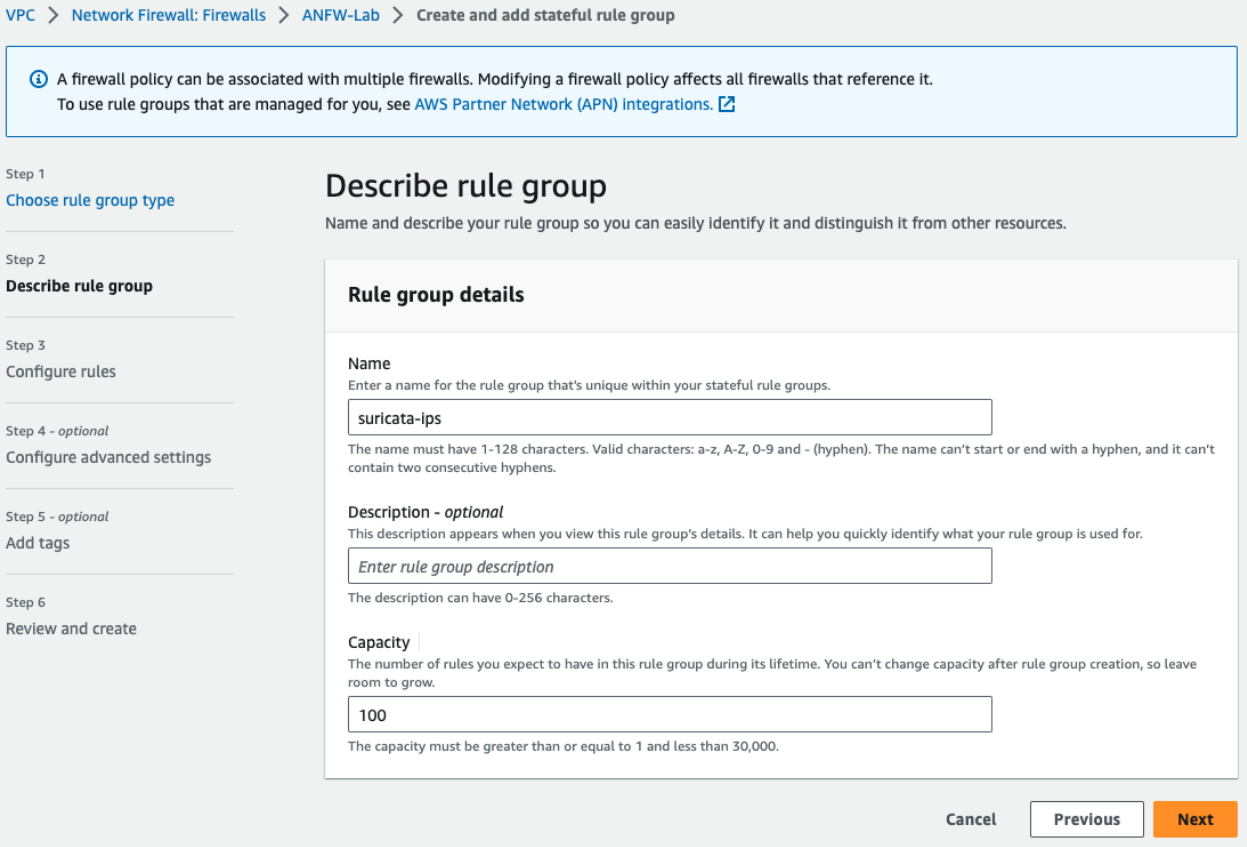
Add three rules as shown below:

Priority	Protocol	Source	Destination	Source port range	Destination port range	Action	Custom action	Masks	Flags
1	TCP	0.0.0.0/0	0.0.0.0/0	0:65535	80	Pass	-	-	-
2	TCP	0.0.0.0/0	0.0.0.0/0	80	0:65535	Pass	-	-	-
3	ICMP	0.0.0.0/0	0.0.0.0/0	-	-	Forward	-	-	-

Navigate to the **Stateless default actions** section and click **Edit.** Then save it after editing.

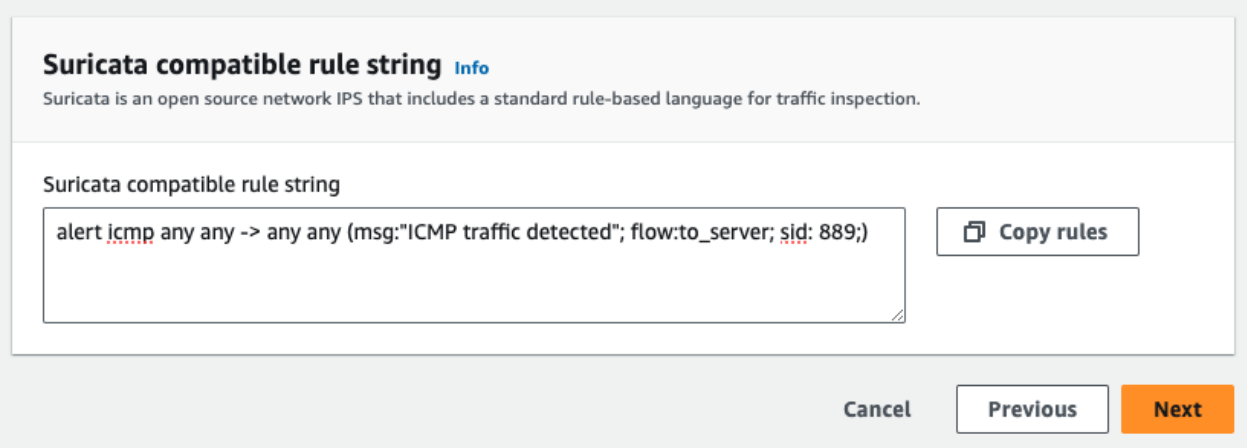


Configure stateful IPS Rules as shown below:



For step 3, scroll down to the **Suricata compatible rule string** section and copy and paste the following Suricata IPS rule into the text box:

alert icmp any any -> any any (msg:"ICMP traffic detected"; flow:to_server; sid: 889;)



The Suricata IPS rule will enable the firewall to log an alert in response to any ICMP traffic with the following message: "ICMP traffic detected".

To test, connect to EC2 Instance (Test Workload) via Session Manager as previously done in the lab. Then copy and paste the following ping command into your secure browser-based shell window.

ping -c 5 www.amazon.com

```
sh-5.2$ ping -c 5 www.amazon.com
PING d3ag4hukkh62yn.cloudfront.net (23.62.25.178) 56(84) bytes of data.

--- d3ag4hukkh62yn.cloudfront.net ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4123ms

sh-5.2$
```

The pings timed out. This is because a stateless **alert** rule will generate an alert and then pass the traffic to the stateful rules engine. Since the stateful rules engine has no rules defined for ICMP traffic, it will drop the packet in accordance with its **Drop established** default action.

Below is the log activity from cloud watch on the traffic generated above.

Key Take away in lab 1.

Through this lab, I learned the basics of AWS Network Firewall and how to enforce access control for public-facing resources. The integration with CloudWatch provided insights into network traffic and potential security events**.**

Key take away in lab 2.

Lab 2 reinforced the importance of a layered security approach by isolating workloads in private subnets and tightening firewall rules. It highlighted the flexibility of AWS Network Firewall policies to adapt as security needs evolve.

Conclusion.

Completing Labs 1 and 2 of the AWS Hands-on Network Firewall Workshop gave me hands-on experience in setting up and securing AWS environments with a network firewall. Through these labs, I gained practical skills in configuring and logging firewall traffic, managing route tables, and applying security best practices to both public and private workloads. Additionally, I learned how to use a **CloudFormation template** to automate the creation of essential AWS resources, streamlining the setup process and ensuring consistency across environments. I look forward to exploring more advanced topics in AWS Network Firewall to deepen my knowledge and further secure cloud-based applications.