

A dark blue vertical bar is on the left. A blue arrow points right from it, containing the date.

7/16/2025

# Secure Web App with Public Proxy + Private Backend on AWS

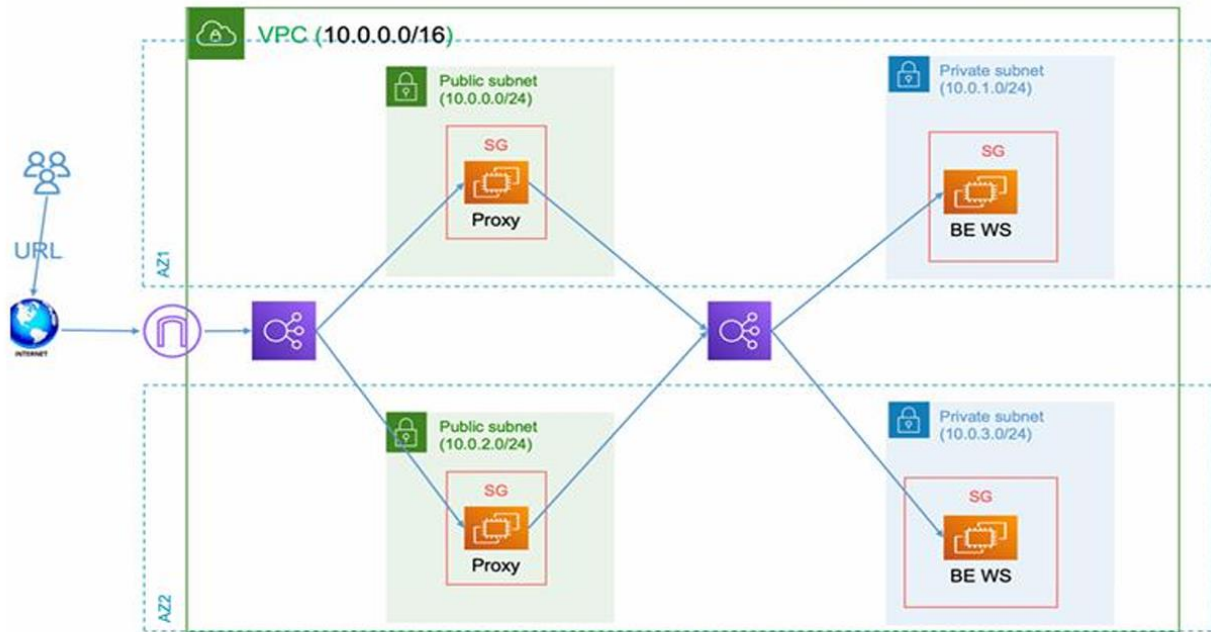
Terraform Project

Several thin, curved lines in dark blue and light grey originate from the bottom left corner and sweep upwards and to the right.

Diaa Qassem  
ITI -CAIRO UNIVERSITY

## **Project Description:**

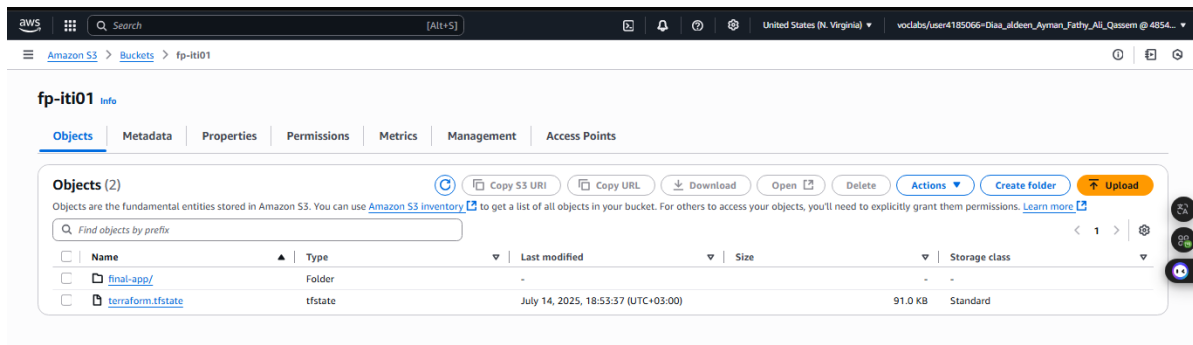
This project consists of a VPC that is deployed in two different availability zones (AZ1 & AZ2). It consists of two public subnets, each containing an EC2 instance that acts as a reverse proxy. They are both connected to a public ALB that takes requests from an internet gateway. The reverse proxy EC2s are connected to two EC2 instances in the two private subnets via an internal ALB.



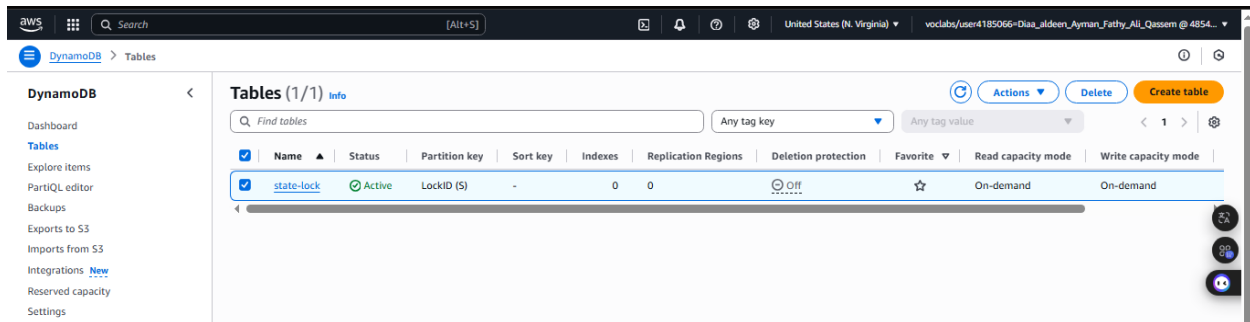
Since we are using modules to create this project, the following picture shows the hierarchy of the files that contain the project code.

```
[diaa@ITI final project]$ ls -la
total 32
drwxr-xr-x. 5 diaa diaa 4096 Jul 15 21:16 .
drwxr-xr-x. 7 diaa diaa  91 Jul 13 09:18 ..
-rw-r--r--. 1 diaa diaa  121 Jul 14 18:53 all-ips.txt
-rw-r--r--. 1 diaa diaa  186 Jul 15 21:23 backend.tf
drwxr-xr-x. 4 diaa diaa   75 Jul 14 02:47 files
-rw-r--r--. 1 diaa diaa 2009 Jul 14 03:05 main.tf
drwxr-xr-x. 8 diaa diaa   75 Jul 13 09:18 modules
-rw-r--r--. 1 diaa diaa  846 Jul 13 09:12 outputs.tf
drwxr-xr-x. 4 diaa diaa   63 Jul 14 02:34 .terraform
-rw-r--r--. 1 diaa diaa 2422 Jul 14 02:37 .terraform.lock.hcl
-rw-r--r--. 1 diaa diaa  180 Jul 15 21:15 terraform.tfvars
-rw-r--r--. 1 diaa diaa  774 Jul 14 03:05 variables.tf
[diaa@ITI final project]$ ls -l modules/
total 0
drwxr-xr-x. 2 diaa diaa 59 Jul 13 09:18 alb
drwxr-xr-x. 2 diaa diaa 59 Jul 13 09:18 ec2
drwxr-xr-x. 2 diaa diaa 59 Jul 13 09:18 nat
drwxr-xr-x. 2 diaa diaa 59 Jul 13 09:18 sg
drwxr-xr-x. 2 diaa diaa 59 Jul 13 09:18 subnets
drwxr-xr-x. 2 diaa diaa 59 Jul 13 09:18 vpc
[diaa@ITI final project]$ ls -l modules/ec2/
total 12
-rw-r--r--. 1 diaa diaa 3639 Jul 15 21:11 main.tf
-rw-r--r--. 1 diaa diaa  709 Jul 13 22:44 outputs.tf
-rw-r--r--. 1 diaa diaa  725 Jul 14 02:58 variables.tf
```

specified the backend as the S3 bucket to store the state file, we can view the state file here.



Also, I have created a DynamoDB table to maintain the state of the state file and prevent simultaneous edits on it.



Now, I will create a workspace named “dev”, and verify that it is created successfully.

```
[diaa@ITI final project]$ terraform workspace new dev
Created and switched to workspace "dev"!

You're now on a new, empty workspace. Workspaces isolate their state,
so if you run "terraform plan" Terraform will not see any existing state
for this configuration.
[diaa@ITI final project]$ terraform workspace show
dev
[diaa@ITI final project]$
```

I begin with running the “terraform init” command. It initializes the working directory and the backend, setting everything up so Terraform can function properly.

The backend is an S3 bucket and a DynamoDB table that contains a lock ID that prevents two users from making changes on the state file at the same time.

```
[diaa@ITI final project]$ terraform init
Initializing the backend...

Successfully configured the backend "s3"! Terraform will automatically
use this backend unless the backend configuration changes.
Initializing modules...
- alb in modules/alb
- ec2 in modules/ec2
- nat_gateway in modules/nat
- security_groups in modules/sg
- subnets in modules/subnets
- vpc in modules/vpc
Initializing provider plugins...
- Finding latest version of hashicorp/local...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/local v2.5.3...
- Installed hashicorp/local v2.5.3 (signed by HashiCorp)
- Installing hashicorp/aws v6.3.0...
```

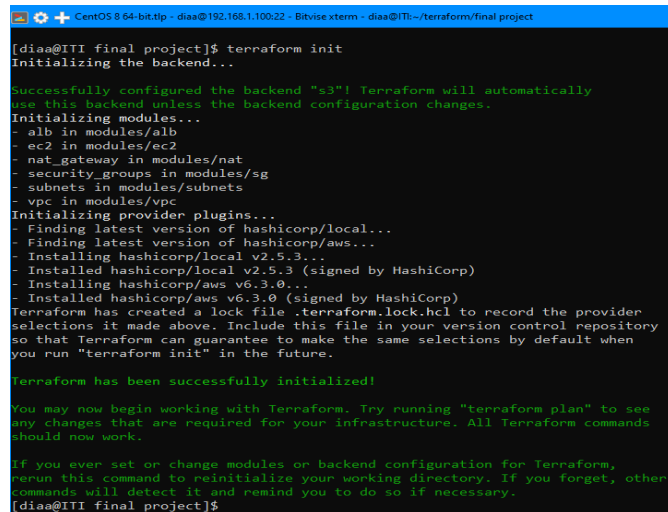
```
[diao@ITI final project]$ terraform init
Initializing the backend...

Successfully configured the backend "s3"! Terraform will automatically
use this backend unless the backend configuration changes.
Initializing modules...
- alb in modules/alb
- ec2 in modules/ec2
- nat_gateway in modules/nat
- security_groups in modules/sg
- subnets in modules/subnets
- vpc in modules/vpc
Initializing provider plugins...
- Finding latest version of hashicorp/local...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/local v2.5.3...
- Installed hashicorp/local v2.5.3 (signed by HashiCorp)
- Installing hashicorp/aws v6.3.0...
- Installed hashicorp/aws v6.3.0 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
[diao@ITI final project]$
```



```
CentOS 8 64-bit.tty - diao@192.168.1.100:22 - Bitvise xterm - diao@ITI--/terraform/final project

[diao@ITI final project]$ terraform init
Initializing the backend...

Successfully configured the backend "s3"! Terraform will automatically
use this backend unless the backend configuration changes.
Initializing modules...
- alb in modules/alb
- ec2 in modules/ec2
- nat_gateway in modules/nat
- security_groups in modules/sg
- subnets in modules/subnets
- vpc in modules/vpc
Initializing provider plugins...
- Finding latest version of hashicorp/local...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/local v2.5.3...
- Installed hashicorp/local v2.5.3 (signed by HashiCorp)
- Installing hashicorp/aws v6.3.0...
- Installed hashicorp/aws v6.3.0 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
[diao@ITI final project]$
```

After, I'm applying the terraform configurations using the "terraform apply command". Here the number of resources is 7 because I have already applied this file before taking this screenshot.

```
module.alb.aws_lb_target_group_attachment.backend_gateway_attachment[1]: Creating...
module.alb.aws_lb_target_group_attachment.backend_gateway_attachment[0]: Creating...
local_file.ips_file: Creating...
module.alb.aws_lb_target_group_attachment.backend_api_attachment[0]: Creating...
module.alb.aws_lb_target_group_attachment.backend_api_attachment[1]: Creating...
local_file.ips_file: Creation complete after 1s [id=f2fad5222497c4aba9a6971e42ba71683e71a7c4]
module.alb.aws_lb_target_group_attachment.backend_gateway_attachment[1]: Creation complete after 1s [id=arn:aws:elasticloadbalancing:us-east-1:485492729952:targetgroup/backend-gateway-tg/3491778092c4e9b7-20250714155352399400000003]
module.alb.aws_lb_target_group_attachment.backend_gateway_attachment[0]: Creation complete after 3s [id=arn:aws:elasticloadbalancing:us-east-1:485492729952:targetgroup/backend-gateway-tg/3491778092c4e9b7-20250714155352399400000004]
module.alb.aws_lb_target_group_attachment.backend_api_attachment[1]: Creation complete after 3s [id=arn:aws:elasticloadbalancing:us-east-1:485492729952:targetgroup/backend-api-tg/56277297458566bc-20250714155352526800000006]
module.alb.aws_lb_target_group_attachment.backend_api_attachment[0]: Creation complete after 3s [id=arn:aws:elasticloadbalancing:us-east-1:485492729952:targetgroup/backend-api-tg/56277297458566bc-20250714155352526800000005]

Apply complete! Resources: 7 added, 0 changed, 2 destroyed.

Outputs:

backend_private_ips = <sensitive>
igw_id = "igw-0c3ab555f4826b9c7"
nat_gateway_ips = [
  "34.194.52.75",
]
private_alb_dns_name = "internal-internalalb-521345524.us-east-1.elb.amazonaws.com"
proxy_public_ips = [
  "54.160.196.136",
  "3.89.73.134",
]
public_alb_dns_name = "public-alb-256643810.us-east-1.elb.amazonaws.com"
vpc_id = "vpc-0ba0194cd3d3586a8"
[diaa@ITI final project]$
```

this output file contains the IP associations.

```
[diaa@ITI final project]$ cat all-ips.txt
public-ip1 54.160.196.136
public-ip2 3.89.73.134
private-ip1 10.0.2.239
private-ip2 10.0.3.237

[diaa@ITI final project]$
```

The following picture shows the resources created by Terraform on the console.

## 1- Resource Map For VPC

The screenshot displays the AWS Management Console's VPC dashboard. The left sidebar shows navigation options like 'Virtual private cloud', 'Subnets', 'Route tables', and 'Security'. The main content area shows 'Your VPCs (1/2)' with a table listing VPCs. Below this, the 'Resource map' for VPC vpc-0ba0194cd3d3586a8 is shown, detailing its subnets, route tables, and network connections.

Name	VPC ID	State	Block Public...	IPv4 CIDR	IPv6 CIDR	DHCP option set	Main route table
lvpc	vpc-0ba0194cd3d3586a8	Available	Off	10.0.0.0/16	-	dhcp-099336b96133a6...	rtb-0a477ae998e7b5b0f
-	vpc-02df998ca9a8c71	Available	Off	172.31.0.0/16	-	dhcp-099336b96133a6...	rtb-0c3b7169b892595f6

**Resource map for vpc-0ba0194cd3d3586a8 / lvpc**

- VPC** (1): lvpc
- Subnets (4)**: us-east-1a (public-subnet-1, private-subnet-1), us-east-1b (public-subnet-2, private-subnet-2)
- Route tables (3)**: private-rt, rtb-0a477ae998e7b5b0f, public-rt
- Network connections (2)**: igw, vpc-nat

## 2- Resource Map For Public LB

The screenshot displays the AWS Management Console's Load Balancers dashboard. The left sidebar shows navigation options like 'Load Balancers', 'Target Groups', and 'Auto Scaling'. The main content area shows 'Load balancers (1/2)' with a table listing load balancers. Below this, the 'Resource map' for Load Balancer public-alb is shown, detailing its listeners, rules, target groups, and targets.

Name	DNS name	State	VPC ID	Availability Zones	Type	Date created
public-alb	public-alb-256643810.us-east-1.elb.amazonaws.com	Active	vpc-0ba0194cd3d3586a8	2 Availability Zones	application	July 14, 2025, 02:40 (UTC+03:00)
internalalb	internal-internalalb-521345.us-east-1.elb.amazonaws.com	Active	vpc-0ba0194cd3d3586a8	2 Availability Zones	application	July 14, 2025, 17:16 (UTC+03:00)

**Load balancer: public-alb**

**Resource map**

- Listeners (1)**: HTTP:80
- Rules (1)**: Priority default, Forward to target group
- Target groups (1)**: Instance, HTTP, frontend-tg
- Targets (2)**: I-04913344d21aedb0e (Port 3000), I-0daa45c47100f118d (Port 3000)

### 3- Resource Map For Internal LB

The screenshot displays the AWS Management Console interface for the 'Load balancers' section. The left-hand navigation pane includes categories like 'Images', 'Elastic Block Store', 'Network & Security', 'Load Balancing', and 'Auto Scaling'. The main content area is titled 'Load balancers (1/2)' and shows a table with two load balancers: 'public-alb' and 'internalalb'. The 'internalalb' is selected, and its 'Resource map' is displayed. The resource map shows a flow from 'Listeners (1)' (HTTP:5000) through 'Rules (1)' (Priority default) to 'Target groups (1)' (Instance, HTTP backend-api-tg), which then points to 'Targets (2)' (I-0bd1aec21cf59024d and I-0f3c0c3d9d987a226, both Healthy). The bottom of the console shows the footer with copyright information and links to Privacy, Terms, and Cookie preferences.

This screenshot shows the same AWS Management Console interface, but the 'Load balancers' table now displays two load balancers: 'public-alb' and 'internalalb'. Below the table, a message states '0 load balancers selected' and 'Select a load balancer above.' The interface is consistent with the previous screenshot, showing the same navigation pane and console layout.



The following picture shows the VPC created.

The screenshot shows the AWS VPC console interface. On the left, there's a navigation menu with 'VPC dashboard' and 'Virtual private cloud' sections. The main area displays 'Your VPCs (1/2)' with a table listing VPCs. The selected VPC is 'lvpc' with ID 'vpc-0ba0194cd3d3586a8'. Below the table, the 'Details' tab is active, showing various attributes of the VPC.

Name	VPC ID	State	Block Public...	IPv4 CIDR	IPv6 CIDR	DHCP option set
lvpc	vpc-0ba0194cd3d3586a8	Available	Off	10.0.0.0/16	-	dopt-099336b96
-	vpc-02df4998ca9a8ccf1	Available	Off	172.31.0.0/16	-	dopt-099336b96

**vpc-0ba0194cd3d3586a8 / lvpc**

**Details**

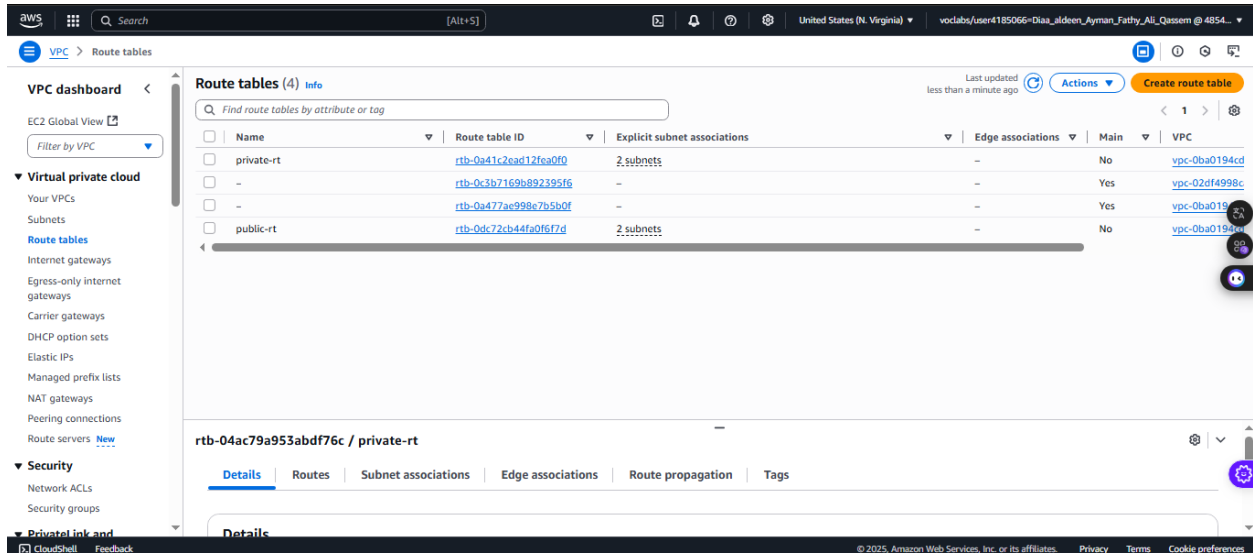
- VPC ID: vpc-0ba0194cd3d3586a8
- State: Available
- Tenancy: default
- Block Public Access: Off
- DNS resolution: Enabled
- DHCP option set: dopt-099336b96133a651f
- Main network ACL: acl-0a25dc897a5efa12f
- Default VPC: No
- IPv4 CIDR: 10.0.0.0/16
- IPv6 CIDR (Network border group): -
- DNS hostnames: Enabled
- Main route table: rtb-0a477ae998e7b5b0f
- Owner ID: 406403730063

The VPC contains four subnets. Two private subnets and two public subnets.

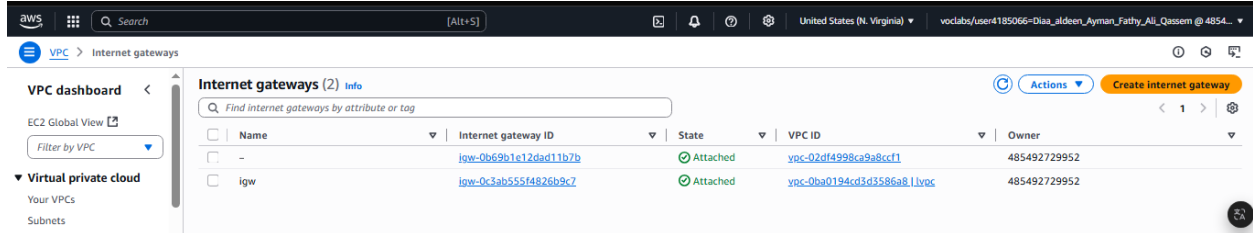
The screenshot shows the AWS VPC console interface, specifically the 'Subnets' page. It displays a table of subnets associated with the VPC 'lvpc'. The subnets are categorized as private or public based on their 'Block Public Access' setting.

Name	Subnet ID	State	VPC	Block Public...	IPv4 CIDR	IPv6 CIDR
-	subnet-050b460e2957c5a63	Available	vpc-02df4998ca9a8ccf1	Off	172.31.80.0/20	-
-	subnet-07b8cc721ba61ffa	Available	vpc-02df4998ca9a8ccf1	Off	172.31.48.0/20	-
private-subnet-2	subnet-0c1b0f7938816500b	Available	vpc-0ba0194cd3d3586a8   lvpc	Off	10.0.3.0/24	-
-	subnet-0c7fa7a971e55b1	Available	vpc-02df4998ca9a8ccf1	Off	172.31.16.0/20	-
-	subnet-0eaecf7d26f21d6c4	Available	vpc-02df4998ca9a8ccf1	Off	172.31.32.0/20	-
public-subnet-1	subnet-0512521278b8de6f9	Available	vpc-0ba0194cd3d3586a8   lvpc	Off	10.0.0.0/24	-
public-subnet-2	subnet-053cd68e650c534b	Available	vpc-0ba0194cd3d3586a8   lvpc	Off	10.0.1.0/24	-
-	subnet-021e343e7f85ff355	Available	vpc-02df4998ca9a8ccf1	Off	172.31.64.0/20	-
-	subnet-08a1e17be028eaba7	Available	vpc-02df4998ca9a8ccf1	Off	172.31.0.0/20	-
private-subnet-1	subnet-039072d5af539cb3b	Available	vpc-0ba0194cd3d3586a8   lvpc	Off	10.0.2.0/24	-

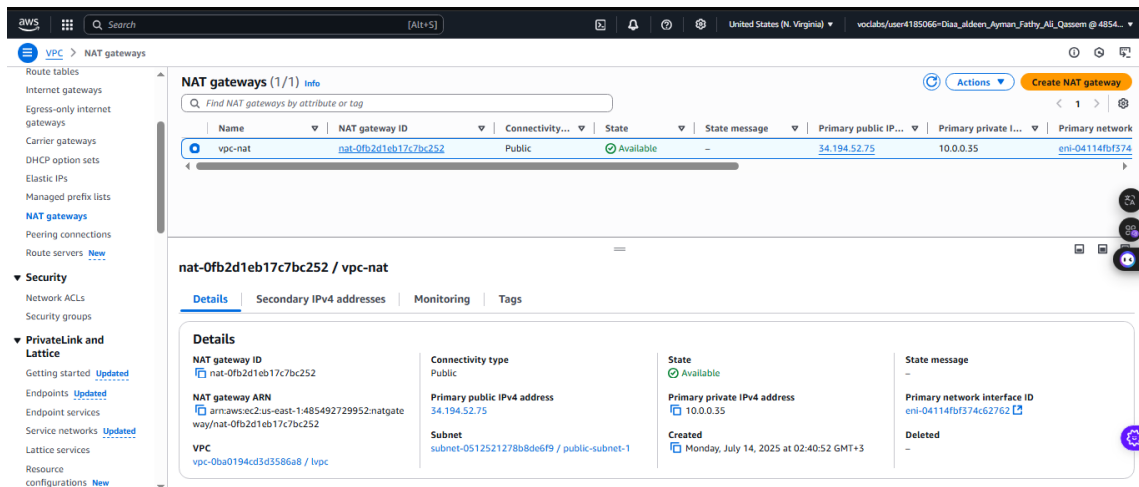
I also associated the public subnet with a route table called “public-rt”, and the private subnet with another route table called “private-rt”



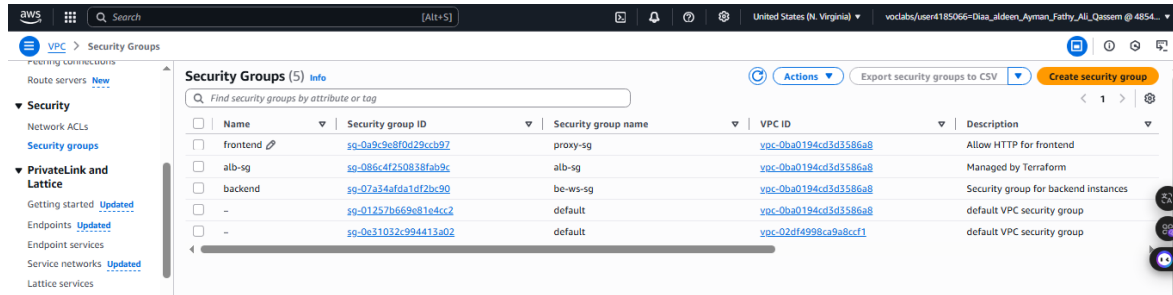
After, I created an internet gateway to allow access for the public EC2 instances on the internet.



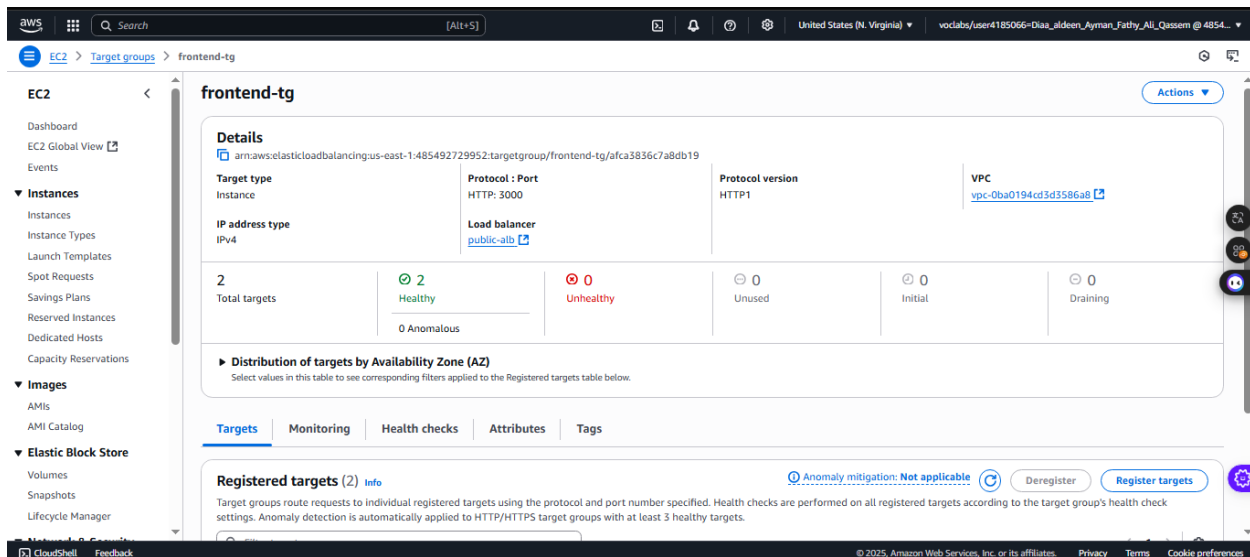
The, I created a NAT gateway as well to allow communication for the private EC2 instances



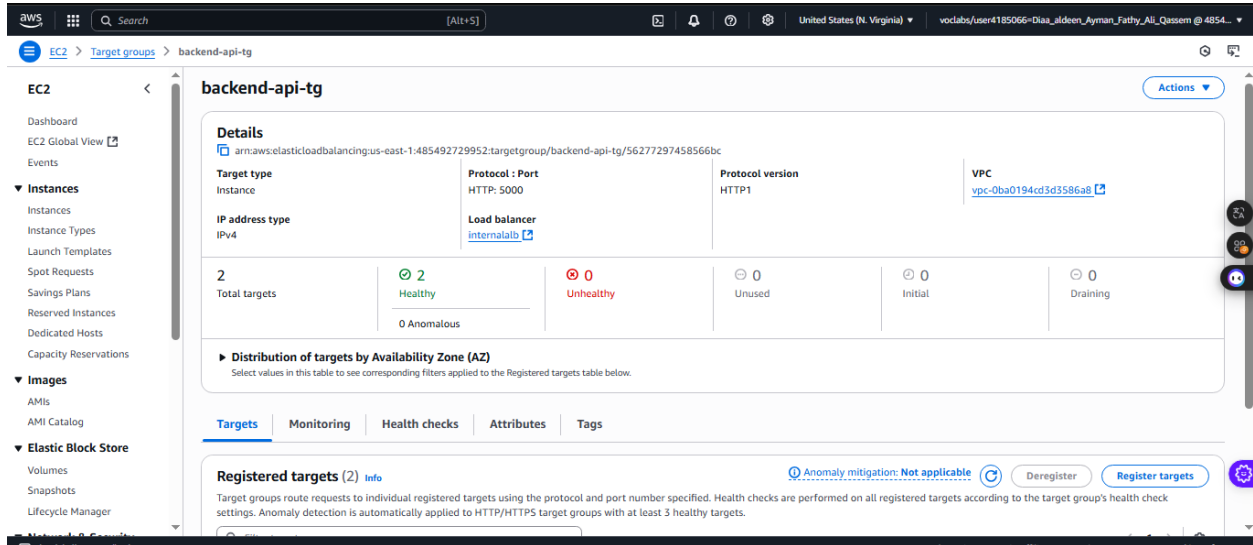
Further, I created security groups for the frontend, the backend, and the ALB, and allowed the needed rules on these SGs.



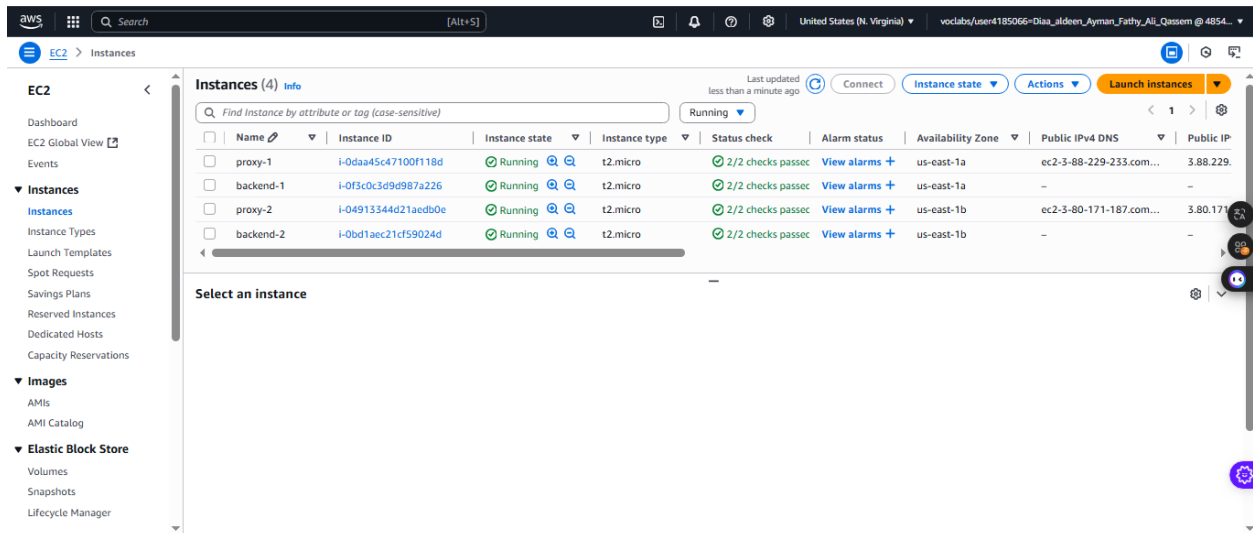
The frontend target group here shows that both servers are in a healthy state.



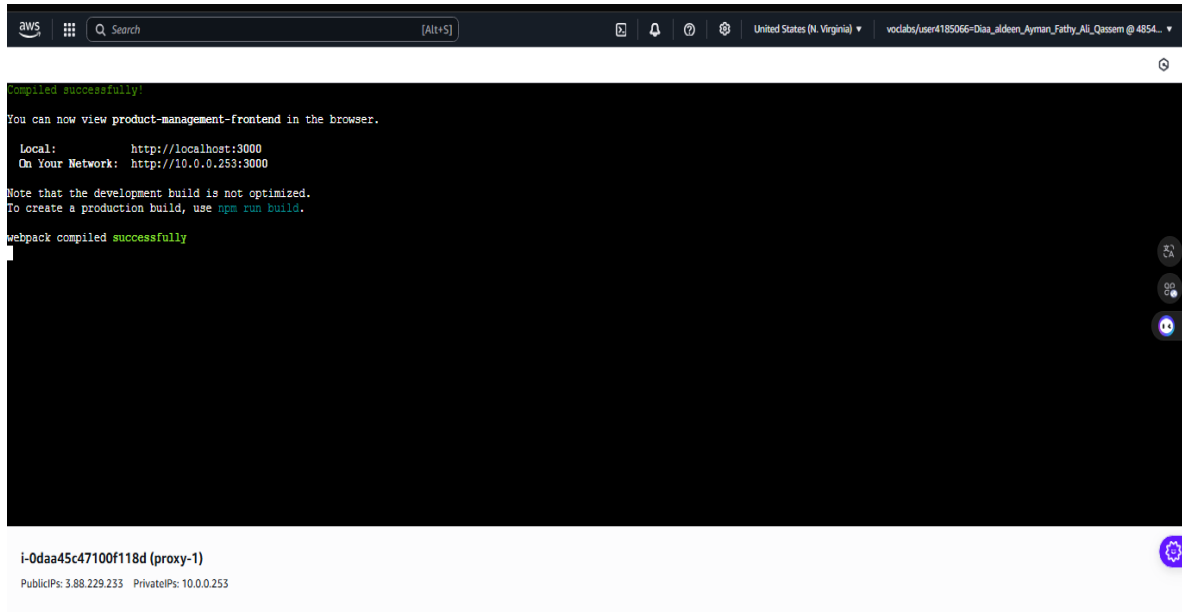
Also, the backend target group shows that both servers are in a healthy state.



Here, there are four EC2 instances created. Two for the reverse proxy and two for the backend.



The following two snapshots include accessing the two proxy EC2 instances via the console. These instances include the frontend of the application.



```
Compiled successfully!

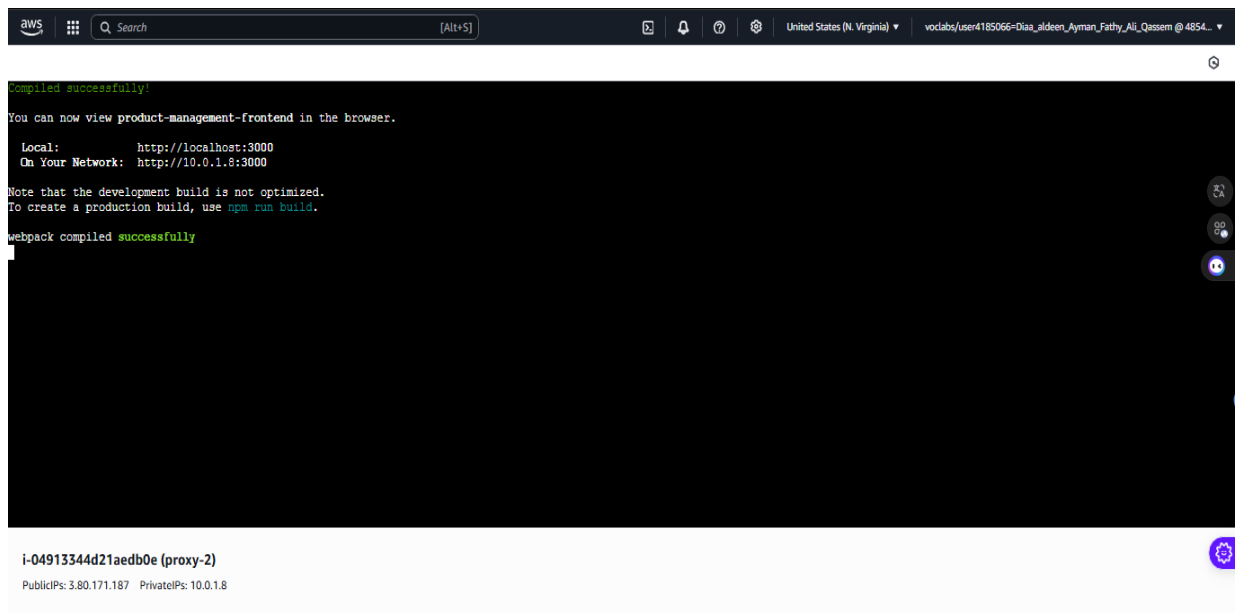
You can now view product-management-frontend in the browser.

Local:      http://localhost:3000
On Your Network: http://10.0.0.253:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```

**i-0daa45c47100f118d (proxy-1)**  
PublicIPs: 3.88.229.233 PrivateIPs: 10.0.0.253



```
Compiled successfully!

You can now view product-management-frontend in the browser.

Local:      http://localhost:3000
On Your Network: http://10.0.1.8:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```

**i-04913344d21aedb0e (proxy-2)**  
PublicIPs: 3.80.171.187 PrivateIPs: 10.0.1.8

Here, I have accessed the backend instances via SSH from the proxy instances by adding the public key of the instances for passwordless access.

```
[ec2-user@ip-10-0-3-237 backend-app]$ npm run start
> backend@1.0.0 start
> node server.js

[dotenv@17.2.0] injecting env (2) from .env (tip: ✨ write to custom object with { processEnv: myObject })
(node:9358) [MONGODB DRIVER] Warning: useUrlParser is a deprecated option: useUrlParser has no effect since M
ion
(Use `node --trace-warnings ...` to show where the warning was created)
(node:9358) [MONGODB DRIVER] Warning: useUnifiedTopology is a deprecated option: useUnifiedTopology has no effect s
r version
Server running on port 5000
Connected to MongoDB
```

```
[ec2-user@ip-10-0-3-237 backend-app]$ nvm install 17.2.0
#####
Computing checksum with sha256sum
Checksums matched!
Now using node v17.2.0 (npm v8.1.4)
[ec2-user@ip-10-0-3-237 backend-app]$ npm run start
> backend@1.0.0 start
> node server.js

[dotenv@17.2.0] injecting env (2) from .env (tip: ✨ write to custom object with { processEnv: myObject })
(node:9358) [MONGODB DRIVER] Warning: useUrlParser is a deprecated option: useUrlParser has no effect since Node.js
ion
(Use `node --trace-warnings ...` to show where the warning was created)
(node:9358) [MONGODB DRIVER] Warning: useUnifiedTopology is a deprecated option: useUnifiedTopology has no effect since M
r version
Server running on port 5000
Connected to MongoDB
```

Here, I have configured the Reverse proxy for the instances.

```
server {
    listen 4000;

    location / {
        proxy_pass http://internal-internalalb-521345524.us-east-1.elb.amazonaws.com:5000/;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }
}

"/etc/nginx/nginx.conf" [readonly] 97L, 2692B
```

i-04913344d21aedb0e (proxy-2)

PublicIPs: 3.80.171.187 PrivateIPs: 10.0.1.8

```

server {
listen 4000;

location / {
proxy_pass http://internal-internalalb-521345524.us-east-1.elb.amazonaws.com:5000/;
proxy_http_version 1.1;
proxy_set_header Upgrade $http_upgrade;
proxy_set_header Connection 'upgrade';
proxy_set_header Host $host;
proxy_cache_bypass $http_upgrade;
}

# Settings for a TLS enabled server.
#
# server {
#     listen      443 ssl;
#     listen      [::]:443 ssl;
#     http2       on;
# }
"/etc/nginx/nginx.conf" 97L, 2696B

```

**i-0daa45c47100f118d (proxy-1)**

PublicIPs: 3.88.229.233 PrivateIPs: 10.0.0.253

The below screenshot shows the application running.

**Product Management**

### Add New Product

Product Name

Price

Description

**Add Product**

### Product List

NAME	PRICE	DESCRIPTION	ACTIONS
test	3636	la2a	<button>Edit</button> <button>Delete</button>
new	9999	yes	<button>Edit</button> <button>Delete</button>
watermelon	200	tohfa	<button>Edit</button> <button>Delete</button>
kiwi	100	sour	<button>Edit</button> <button>Delete</button>