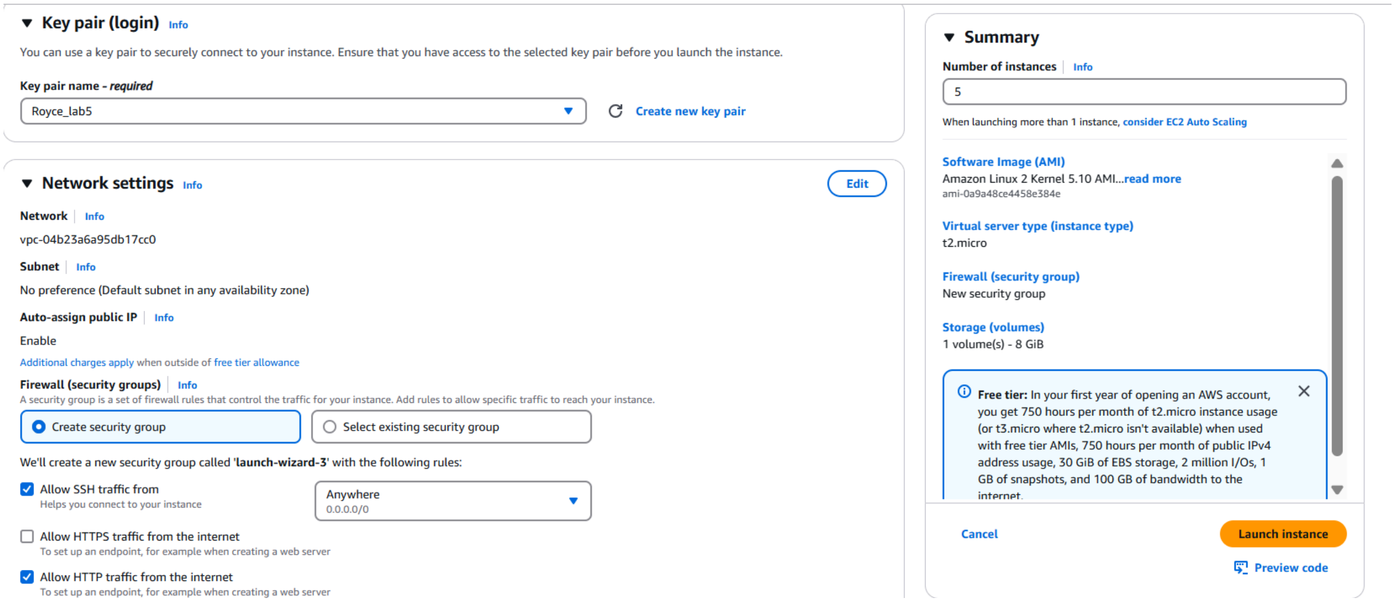
1) I created an EC2 instance using the Amazon Linux 2 AMI to serve as my web server. This instance was selected for its compatibility with the Nginx web server and its efficiency in handling network traffic.



▼ **Key pair (login)** Info

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - *required*

Royce_lab5 ▼ | Create new key pair

▼ **Network settings** Info | Edit

Network | Info

vpc-04b23a6a95db17cc0

Subnet | Info

No preference (Default subnet in any availability zone)

Auto-assign public IP | Info

Enable

Additional charges apply when outside of free tier allowance

Firewall (security groups) | Info

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

● Create security group ○ Select existing security group

We'll create a new security group called '**launch-wizard-3**' with the following rules:

☑ Allow SSH traffic from — Anywhere 0.0.0.0/0
Helps you connect to your instance

☐ Allow HTTPS traffic from the internet
To set up an endpoint, for example when creating a web server

☑ Allow HTTP traffic from the internet
To set up an endpoint, for example when creating a web server

▼ **Summary**

Number of instances | Info

5

When launching more than 1 instance, consider EC2 Auto Scaling

Software Image (AMI)
Amazon Linux 2 Kernel 5.10 AMI...read more
ami-0a9a48ce4458e384e

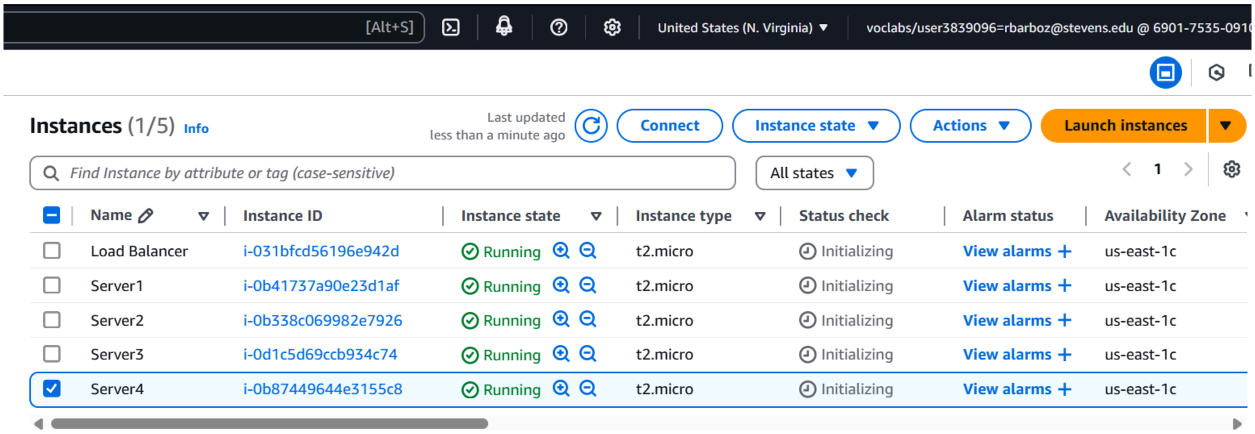
Virtual server type (instance type)
t2.micro

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 t2.micro isn't available) when used with free tier AMIs, 750 hours per month of public IPv4 address usage, 30 GiB of EBS storage, 2 million I/Os, 1 GB of snapshots, and 100 GB of bandwidth to the internet.

Cancel
Launch instance
Preview code



[Alt+S]
United States (N. Virginia) ▼
voclabs/user3839096=rbarboz@stevens.edu @ 6901-7535-091

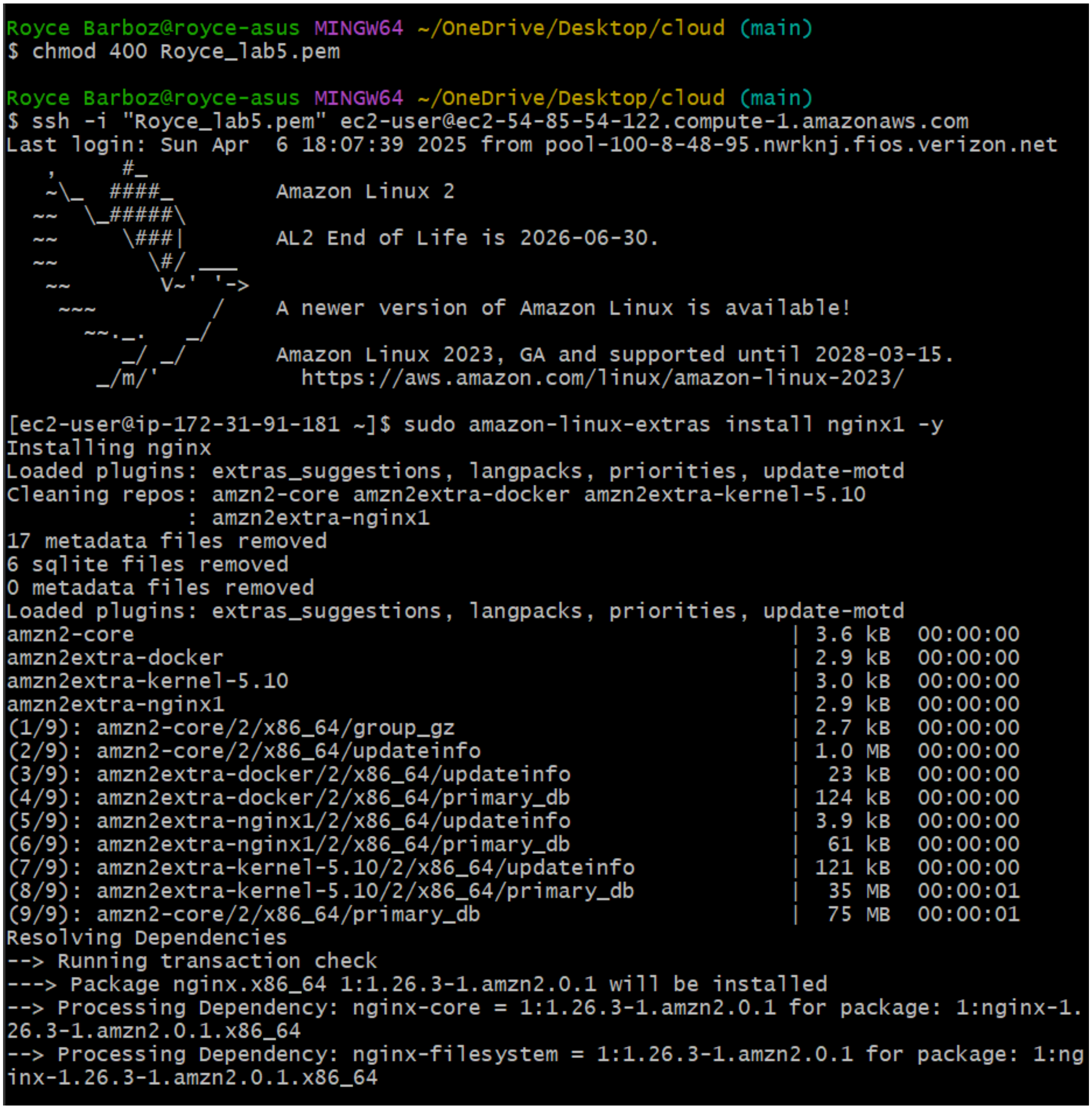
Instances (1/5) Info

Last updated less than a minute ago

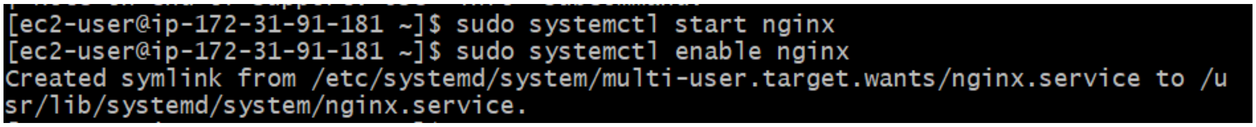
Connect
Instance state ▼
Actions ▼
Launch instances ▼
Find Instance by attribute or tag (case-sensitive)
All states ▼
1

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
☐	Load Balancer	i-031bfcd56196e942d	✓ Running	t2.micro	Initializing	View alarms +	us-east-1c
☐	Server1	i-0b41737a90e23d1af	✓ Running	t2.micro	Initializing	View alarms +	us-east-1c
☐	Server2	i-0b338c069982e7926	✓ Running	t2.micro	Initializing	View alarms +	us-east-1c
☐	Server3	i-0d1c5d69ccb934c74	✓ Running	t2.micro	Initializing	View alarms +	us-east-1c
☑	Server4	i-0b87449644e3155c8	✓ Running	t2.micro	Initializing	View alarms +	us-east-1c

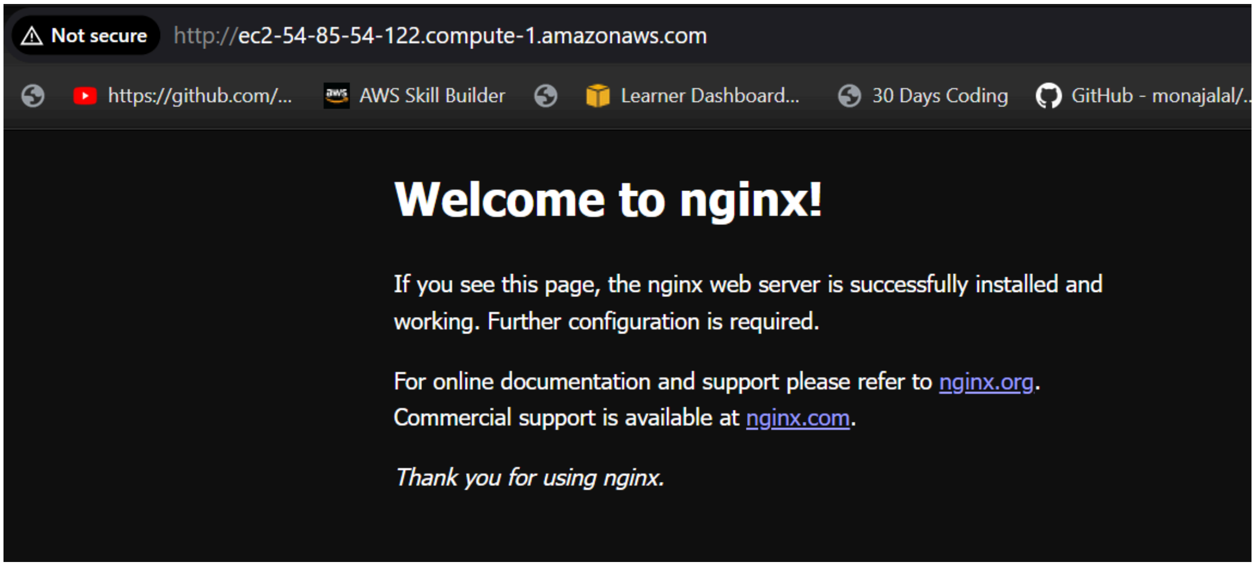
2) connected to Server 1,2,3,4 using SSH, and I installed Nginx using the command sudo yum install nginx -y. This setup will allow the server to handle incoming HTTP requests from the load balancer.



```
Royce Barboz@royce-asus MINGW64 ~/OneDrive/Desktop/cloud (main)
$ chmod 400 Royce_lab5.pem

Royce Barboz@royce-asus MINGW64 ~/OneDrive/Desktop/cloud (main)
$ ssh -i "Royce_lab5.pem" ec2-user@ec2-54-85-54-122.compute-1.amazonaws.com
Last login: Sun Apr  6 18:07:39 2025 from pool-100-8-48-95.nwrknj.fios.verizon.net
   ,     #_
   ~\_  ####_        Amazon Linux 2
  ~~  \_#####\
  ~~     \###|       AL2 End of Life is 2026-06-30.
  ~~       \#/ ___
   ~~       V~' '->
    ~~~         /    A newer version of Amazon Linux is available!
      ~~._.   _/
         _/ _/       Amazon Linux 2023, GA and supported until 2028-03-15.
       _/m/'           https://aws.amazon.com/linux/amazon-linux-2023/

[ec2-user@ip-172-31-91-181 ~]$ sudo amazon-linux-extras install nginx1 -y
Installing nginx
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Cleaning repos: amzn2-core amzn2extra-docker amzn2extra-kernel-5.10
              : amzn2extra-nginx1
17 metadata files removed
6 sqlite files removed
0 metadata files removed
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core                                                  | 3.6 kB  00:00:00
amzn2extra-docker                                           | 2.9 kB  00:00:00
amzn2extra-kernel-5.10                                      | 3.0 kB  00:00:00
amzn2extra-nginx1                                           | 2.9 kB  00:00:00
(1/9): amzn2-core/2/x86_64/group_gz                         | 2.7 kB  00:00:00
(2/9): amzn2-core/2/x86_64/updateinfo                       | 1.0 MB  00:00:00
(3/9): amzn2extra-docker/2/x86_64/updateinfo                |  23 kB  00:00:00
(4/9): amzn2extra-docker/2/x86_64/primary_db                | 124 kB  00:00:00
(5/9): amzn2extra-nginx1/2/x86_64/updateinfo                | 3.9 kB  00:00:00
(6/9): amzn2extra-nginx1/2/x86_64/primary_db                |  61 kB  00:00:00
(7/9): amzn2extra-kernel-5.10/2/x86_64/updateinfo           | 121 kB  00:00:00
(8/9): amzn2extra-kernel-5.10/2/x86_64/primary_db           |  35 MB  00:00:01
(9/9): amzn2-core/2/x86_64/primary_db                       |  75 MB  00:00:01
Resolving Dependencies
--> Running transaction check
---> Package nginx.x86_64 1:1.26.3-1.amzn2.0.1 will be installed
--> Processing Dependency: nginx-core = 1:1.26.3-1.amzn2.0.1 for package: 1:nginx-1.
26.3-1.amzn2.0.1.x86_64
--> Processing Dependency: nginx-filesystem = 1:1.26.3-1.amzn2.0.1 for package: 1:ng
inx-1.26.3-1.amzn2.0.1.x86_64
```



```
[ec2-user@ip-172-31-91-181 ~]$ sudo systemctl start nginx
[ec2-user@ip-172-31-91-181 ~]$ sudo systemctl enable nginx
Created symlink from /etc/systemd/system/multi-user.target.wants/nginx.service to /u
sr/lib/systemd/system/nginx.service.
```

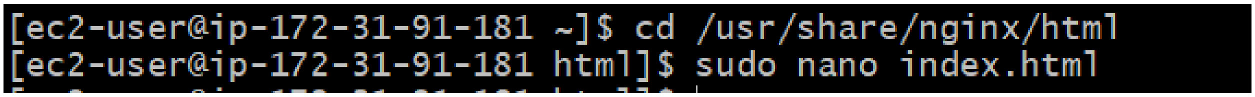


Welcome to nginx!

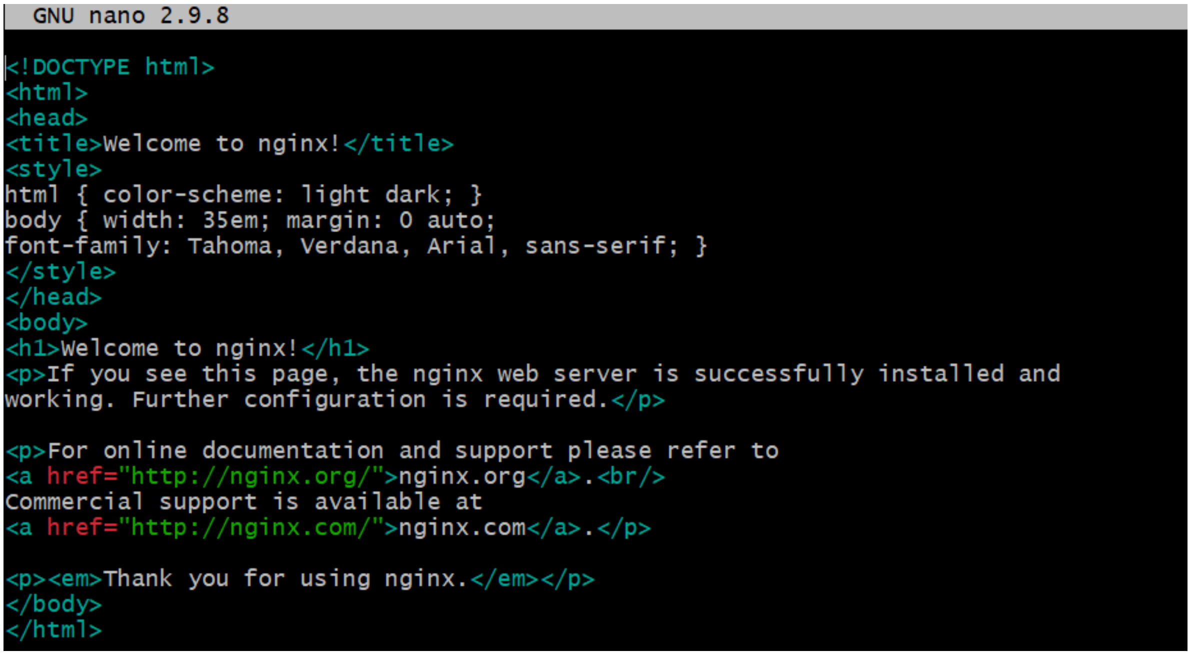
If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

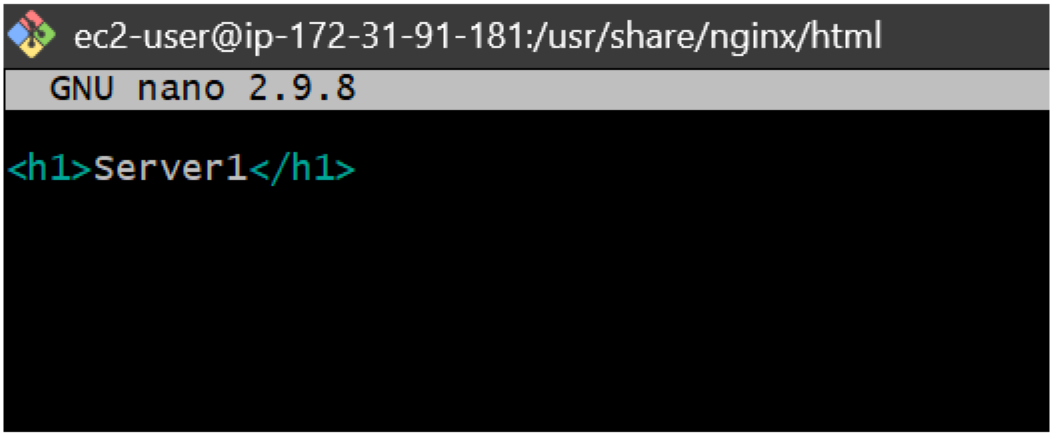
Thank you for using nginx.



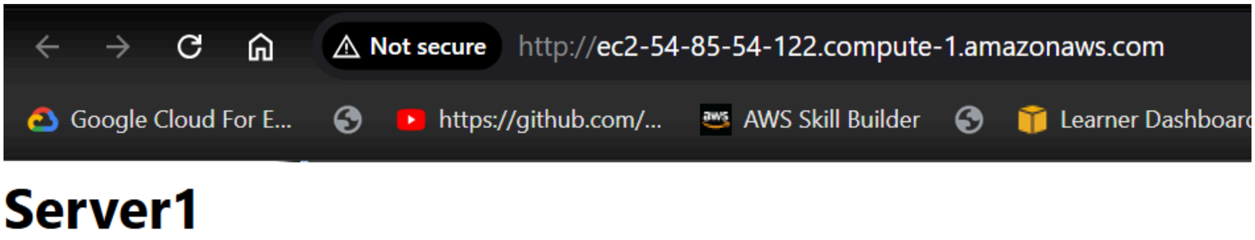
```
[ec2-user@ip-172-31-91-181 ~]$ cd /usr/share/nginx/html
[ec2-user@ip-172-31-91-181 html]$ sudo nano index.html
```



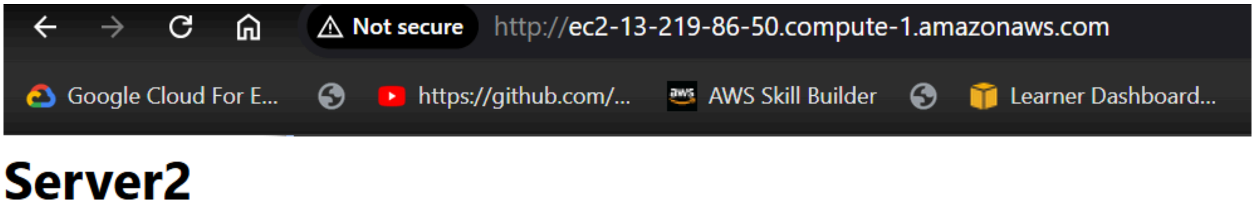
```
  GNU nano 2.9.8

<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```



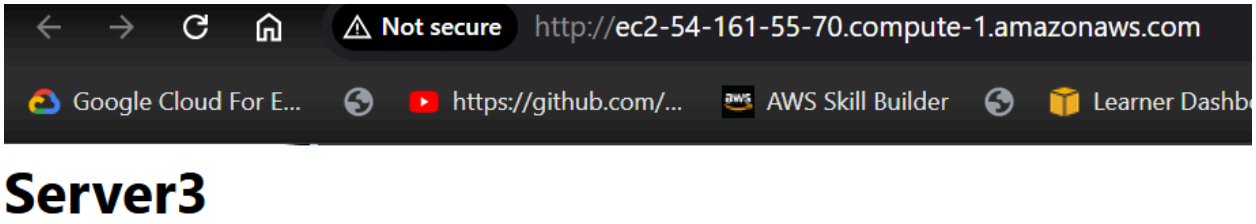
```
  GNU nano 2.9.8

<h1>Server1</h1>
```



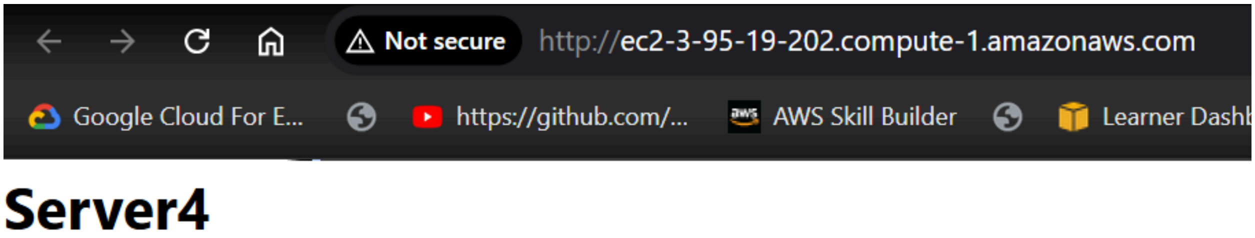
Server1



Server2

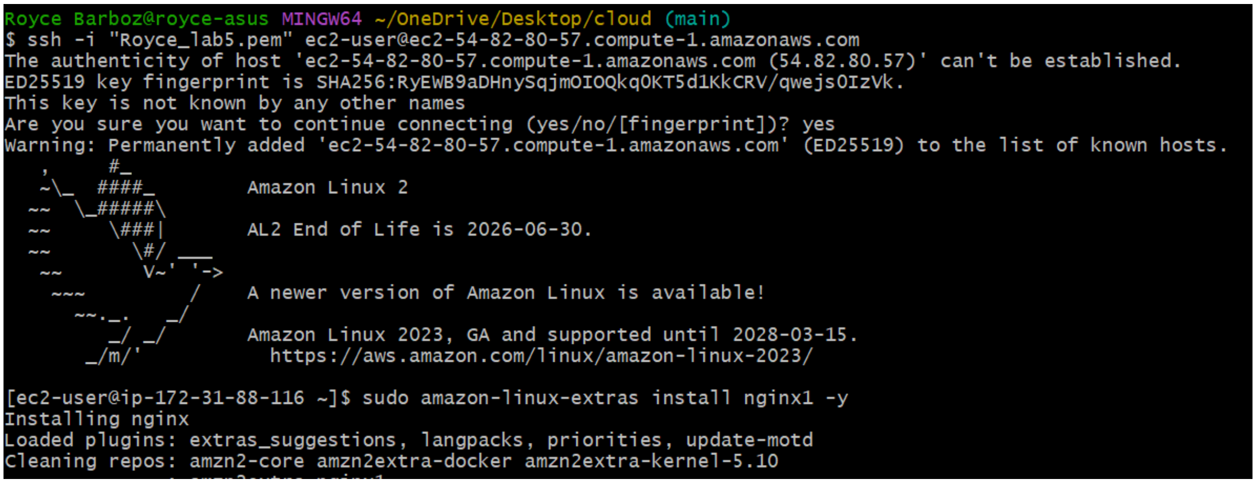


Server3

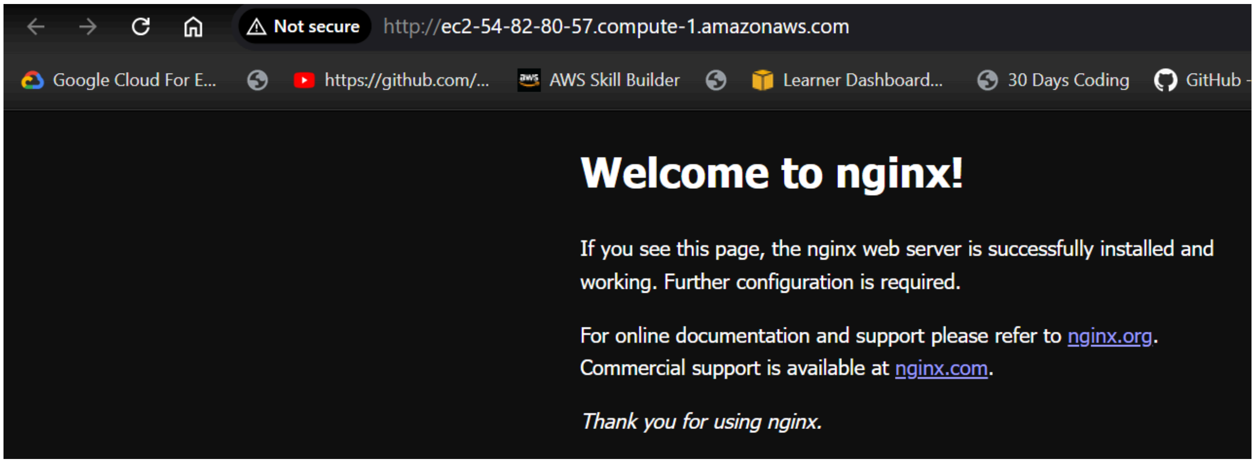


Server4

3) I configured the Nginx load balancer to distribute traffic across multiple servers. This was done by editing the nginx.conf file and defining an upstream block with multiple servers and weighted load balancing. Each server was assigned a specific weight to control the traffic distribution



```
Royce Barboz@royce-asus MINGW64 ~/OneDrive/Desktop/cloud (main)
$ ssh -i "Royce_lab5.pem" ec2-user@ec2-54-82-80-57.compute-1.amazonaws.com
The authenticity of host 'ec2-54-82-80-57.compute-1.amazonaws.com (54.82.80.57)' can't be established.
ED25519 key fingerprint is SHA256:RyEWB9aDHnySqjmOIOQkqOKT5d1KkCRV/qwejsOIzVk.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-54-82-80-57.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
   ,     #_
   ~\_  ####_        Amazon Linux 2
  ~~  \_#####\
  ~~     \###|       AL2 End of Life is 2026-06-30.
  ~~       \#/ ___
   ~~       V~' '->
    ~~~         /    A newer version of Amazon Linux is available!
      ~~._.   _/
         _/ _/       Amazon Linux 2023, GA and supported until 2028-03-15.
       _/m/'           https://aws.amazon.com/linux/amazon-linux-2023/

[ec2-user@ip-172-31-88-116 ~]$ sudo amazon-linux-extras install nginx1 -y
Installing nginx
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Cleaning repos: amzn2-core amzn2extra-docker amzn2extra-kernel-5.10
```

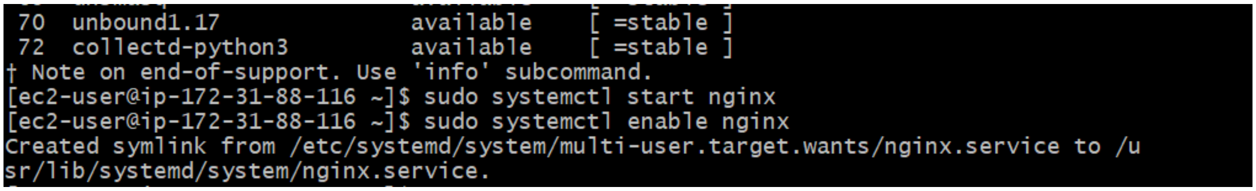


Welcome to nginx!

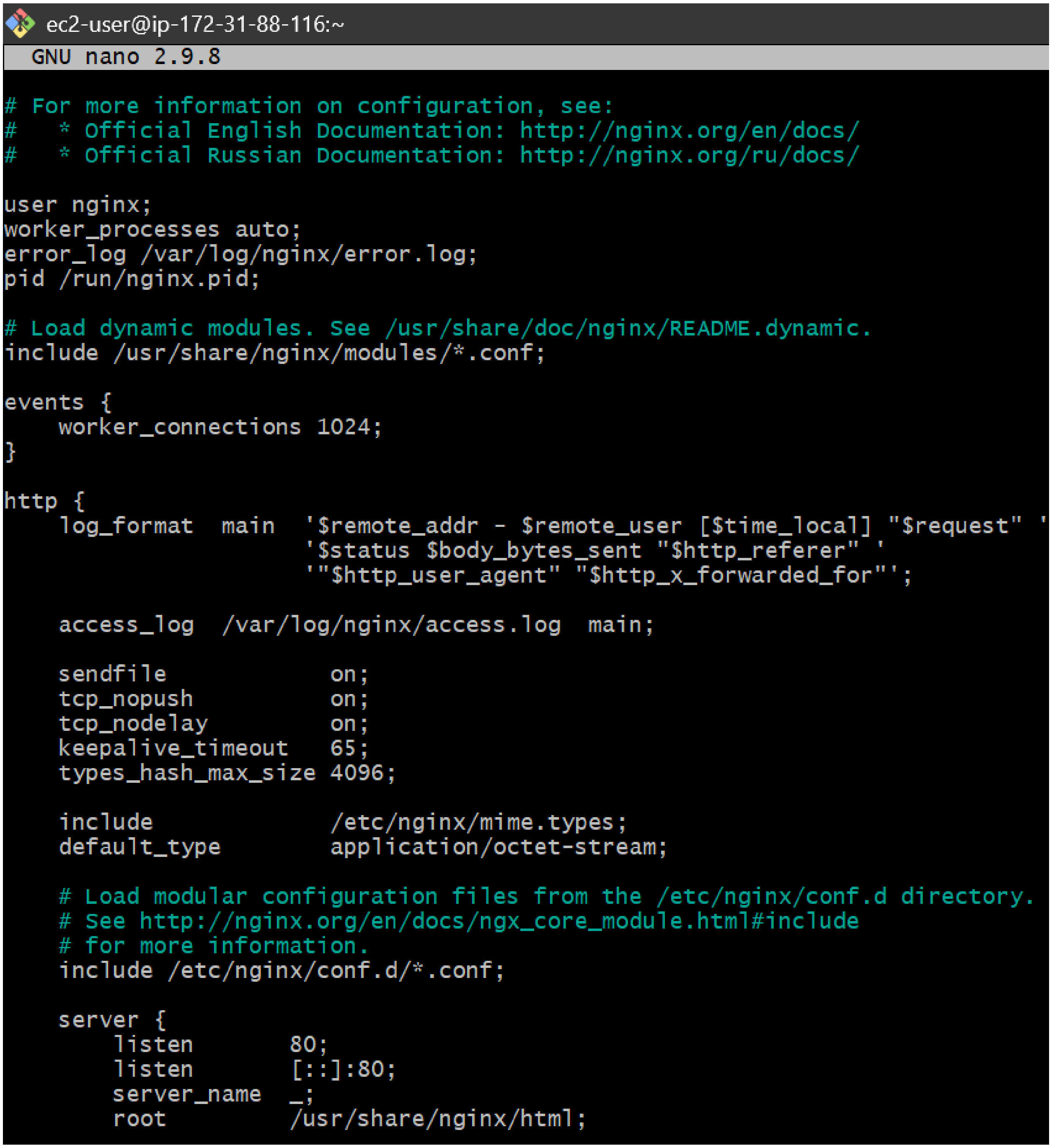
If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

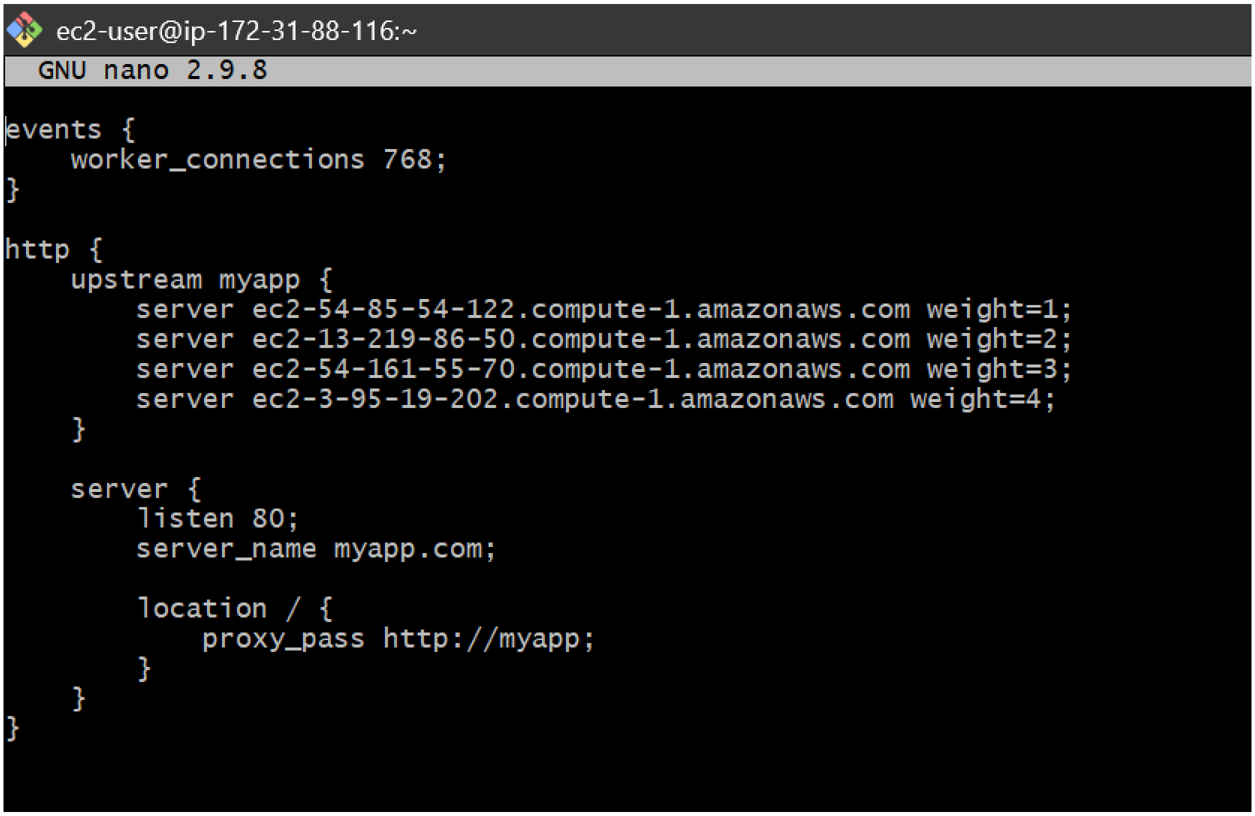


```
 70  unbound1.17              available    [ =stable ]
 72  collectd-python3         available    [ =stable ]
† Note on end-of-support. Use 'info' subcommand.
[ec2-user@ip-172-31-88-116 ~]$ sudo systemctl start nginx
[ec2-user@ip-172-31-88-116 ~]$ sudo systemctl enable nginx
Created symlink from /etc/systemd/system/multi-user.target.wants/nginx.service to /u
sr/lib/systemd/system/nginx.service.
```



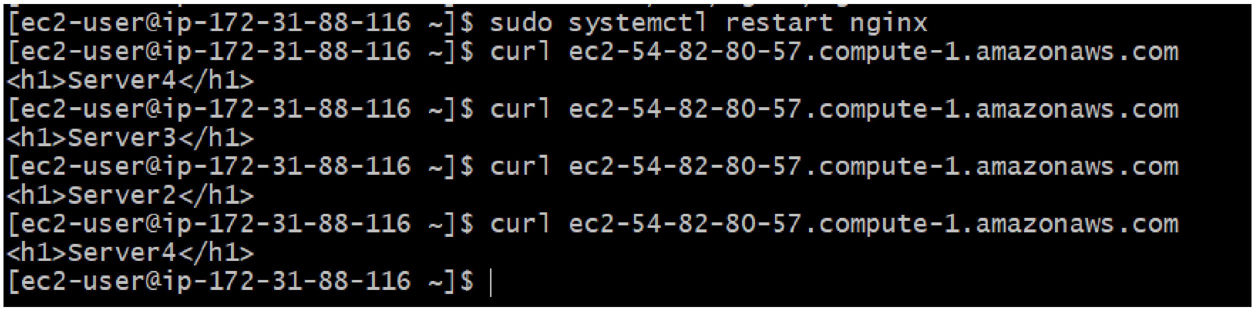
ec2-user@ip-172-31-88-116:~

```
  GNU nano 2.9.8

# For more information on configuration, see:
#   * Official English Documentation: http://nginx.org/en/docs/
#   * Official Russian Documentation: http://nginx.org/ru/docs/

user nginx;
worker_processes auto;
error_log /var/log/nginx/error.log;
pid /run/nginx.pid;

# Load dynamic modules. See /usr/share/doc/nginx/README.dynamic.
include /usr/share/nginx/modules/*.conf;

events {
    worker_connections 1024;
}

http {
    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';

    access_log  /var/log/nginx/access.log  main;

    sendfile            on;
    tcp_nopush          on;
    tcp_nodelay         on;
    keepalive_timeout   65;
    types_hash_max_size 4096;

    include             /etc/nginx/mime.types;
    default_type        application/octet-stream;

    # Load modular configuration files from the /etc/nginx/conf.d directory.
    # See http://nginx.org/en/docs/ngx_core_module.html#include
    # for more information.
    include /etc/nginx/conf.d/*.conf;

    server {
        listen       80;
        listen       [::]:80;
        server_name  _;
        root         /usr/share/nginx/html;
```



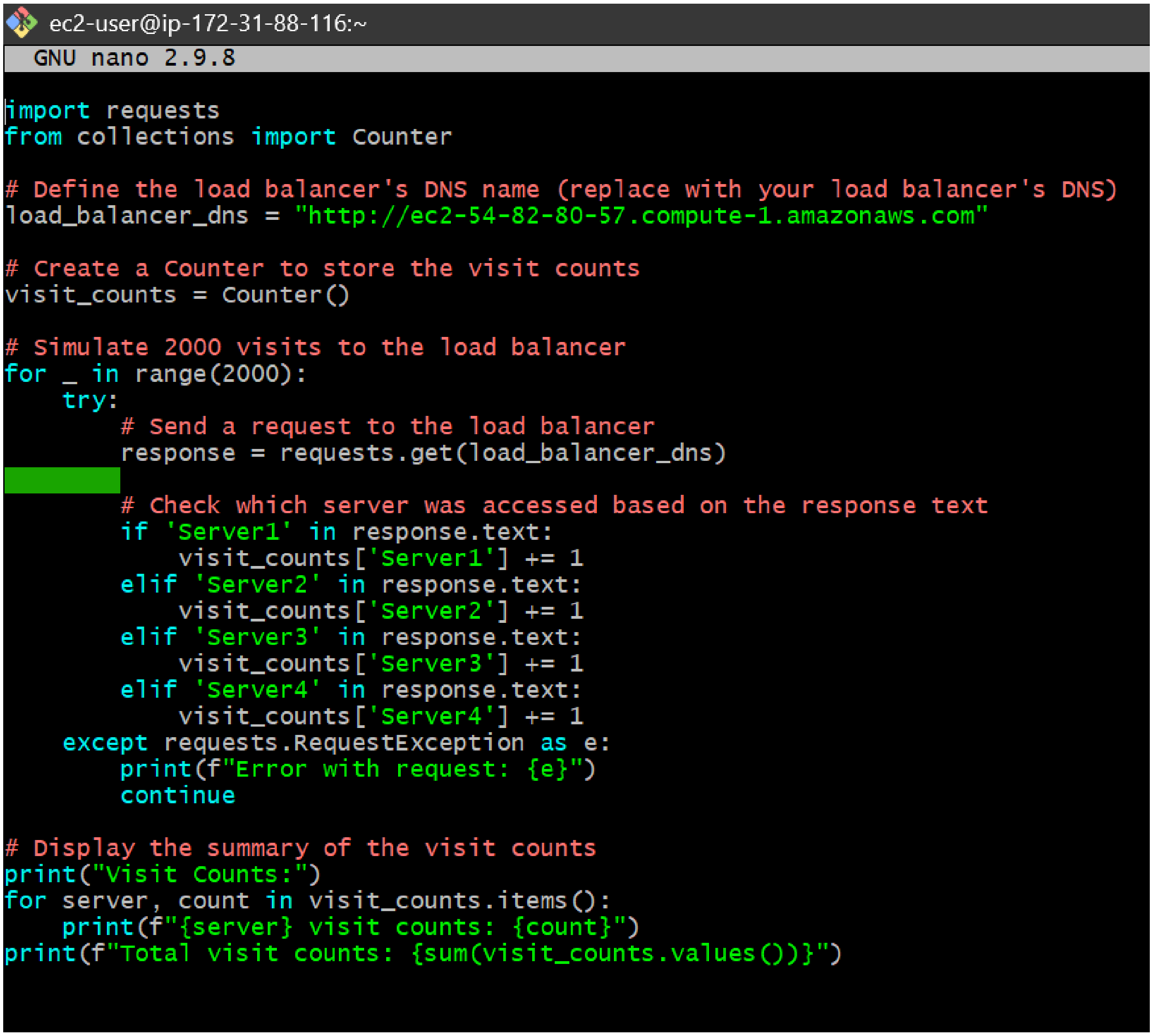
```
events {
    worker_connections 768;
}

http {
    upstream myapp {
        server ec2-54-85-54-122.compute-1.amazonaws.com weight=1;
        server ec2-13-219-86-50.compute-1.amazonaws.com weight=2;
        server ec2-54-161-55-70.compute-1.amazonaws.com weight=3;
        server ec2-3-95-19-202.compute-1.amazonaws.com weight=4;
    }

    server {
        listen 80;
        server_name myapp.com;

        location / {
            proxy_pass http://myapp;
        }
    }
}
```

4. After configuring the load balancer, I tested its functionality by accessing the load balancer's public DNS. The responses were correctly routed to different web servers, verifying that the load balancer was distributing traffic as intended.



```
[ec2-user@ip-172-31-88-116 ~]$ sudo systemctl restart nginx
[ec2-user@ip-172-31-88-116 ~]$ curl ec2-54-82-80-57.compute-1.amazonaws.com
<h1>Server4</h1>
[ec2-user@ip-172-31-88-116 ~]$ curl ec2-54-82-80-57.compute-1.amazonaws.com
<h1>Server3</h1>
[ec2-user@ip-172-31-88-116 ~]$ curl ec2-54-82-80-57.compute-1.amazonaws.com
<h1>Server2</h1>
[ec2-user@ip-172-31-88-116 ~]$ curl ec2-54-82-80-57.compute-1.amazonaws.com
<h1>Server4</h1>
[ec2-user@ip-172-31-88-116 ~]$ |
```

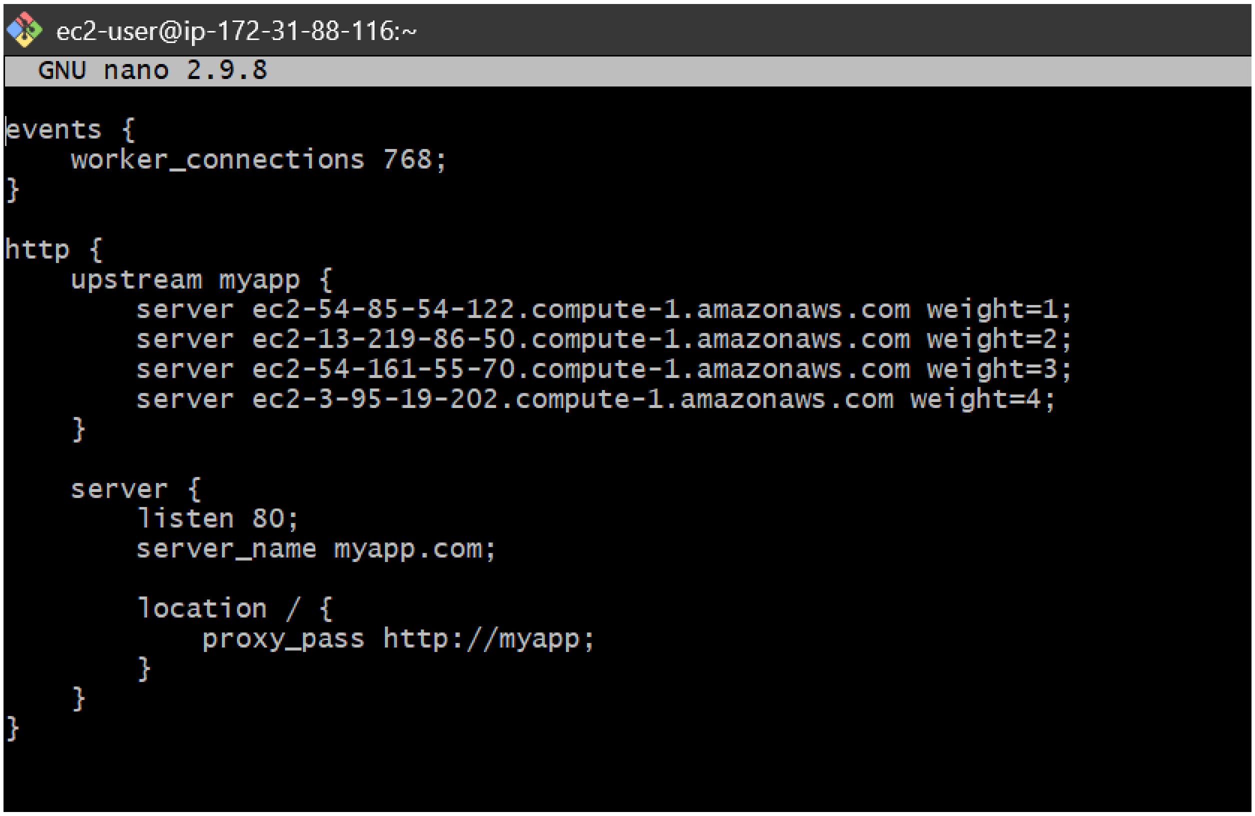
5. I created a Python script to simulate 2000 visits to the load balancer. This allowed me to collect data on how the load balancer distributed traffic across the web servers. The script successfully recorded the number of visits to each server.



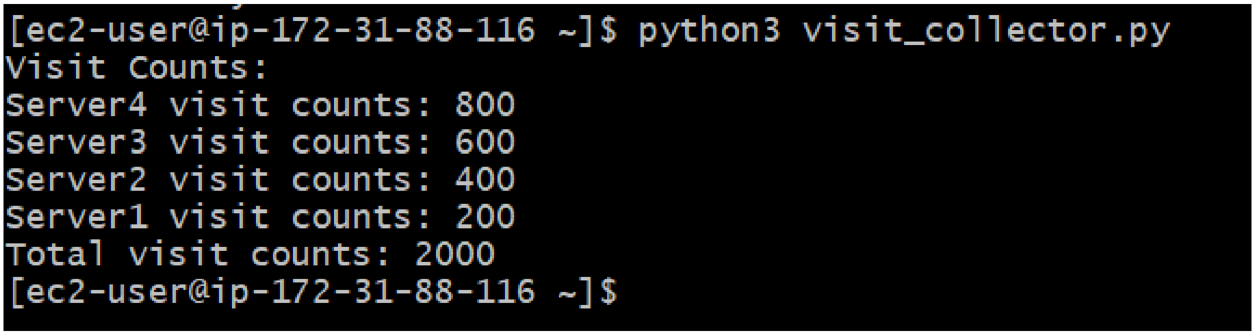
```
import requests
from collections import Counter

# Define the load balancer's DNS name (replace with your load balancer's DNS)
load_balancer_dns = "http://ec2-54-82-80-57.compute-1.amazonaws.com"

# Create a Counter to store the visit counts
visit_counts = Counter()

# Simulate 2000 visits to the load balancer
for _ in range(2000):
    try:
        # Send a request to the load balancer
        response = requests.get(load_balancer_dns)

        # Check which server was accessed based on the response text
        if 'Server1' in response.text:
            visit_counts['Server1'] += 1
        elif 'Server2' in response.text:
            visit_counts['Server2'] += 1
        elif 'Server3' in response.text:
            visit_counts['Server3'] += 1
        elif 'Server4' in response.text:
            visit_counts['Server4'] += 1
    except requests.RequestException as e:
        print(f"Error with request: {e}")
        continue

# Display the summary of the visit counts
print("Visit Counts:")
for server, count in visit_counts.items():
    print(f"{server} visit counts: {count}")
print(f"Total visit counts: {sum(visit_counts.values())}")
```

Scenario 1: Weighted Traffic Distribution (Increasing Weights for Servers)

In this scenario, the weights of the servers were set as follows: Server 1 (weight 1), Server 2 (weight 2), Server 3 (weight 3), and Server 4 (weight 4). As a result, the traffic distribution was not equal. Server 4, with the highest weight, handled the most traffic, followed by Server 3, Server 2, and Server 1. The load balancer routed the majority of requests to Server 4, which handled approximately 40% of the total traffic, with the other servers handling traffic in proportion to their assigned weights. This confirmed that the load balancer was accurately respecting the weight distribution.



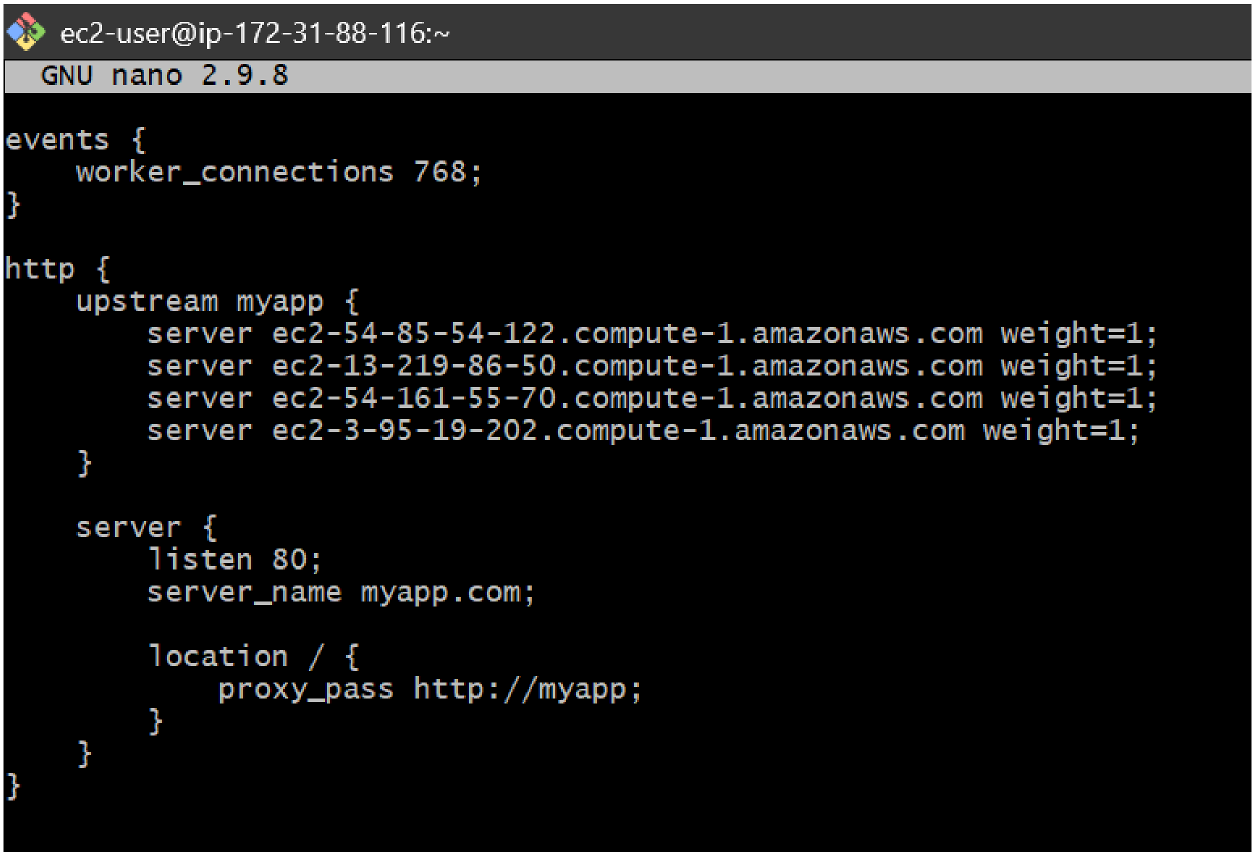
```
events {
    worker_connections 768;
}

http {
    upstream myapp {
        server ec2-54-85-54-122.compute-1.amazonaws.com weight=1;
        server ec2-13-219-86-50.compute-1.amazonaws.com weight=2;
        server ec2-54-161-55-70.compute-1.amazonaws.com weight=3;
        server ec2-3-95-19-202.compute-1.amazonaws.com weight=4;
    }

    server {
        listen 80;
        server_name myapp.com;

        location / {
            proxy_pass http://myapp;
        }
    }
}
```



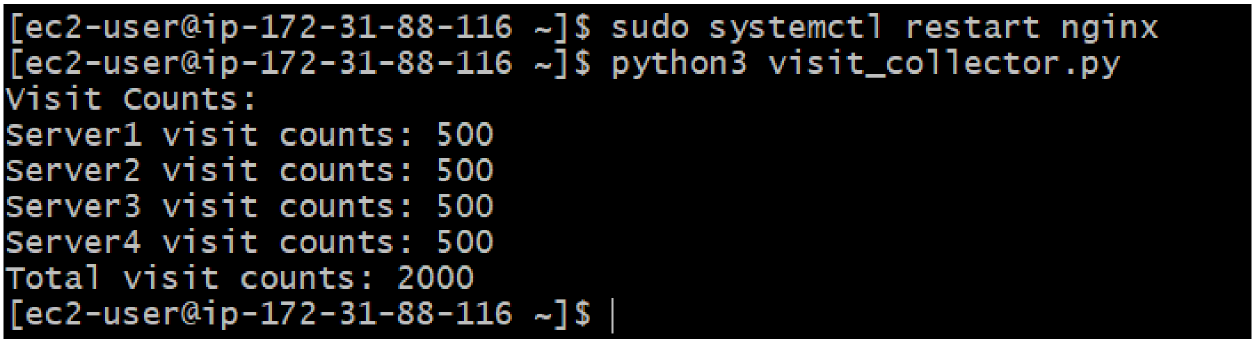
```
[ec2-user@ip-172-31-88-116 ~]$ python3 visit_collector.py
Visit Counts:
Server4 visit counts: 800
Server3 visit counts: 600
Server2 visit counts: 400
Server1 visit counts: 200
Total visit counts: 2000
[ec2-user@ip-172-31-88-116 ~]$
```

Scenario 2: Equal Traffic Distribution (Weight = 1 for All Servers)

In this scenario, each server received an equal amount of traffic, as all servers were assigned the same weight (1). The results showed that the load balancer distributed the incoming traffic evenly across all four servers. Each server handled approximately 25% of the total traffic, demonstrating that the load balancer was working as expected to equally balance the load.



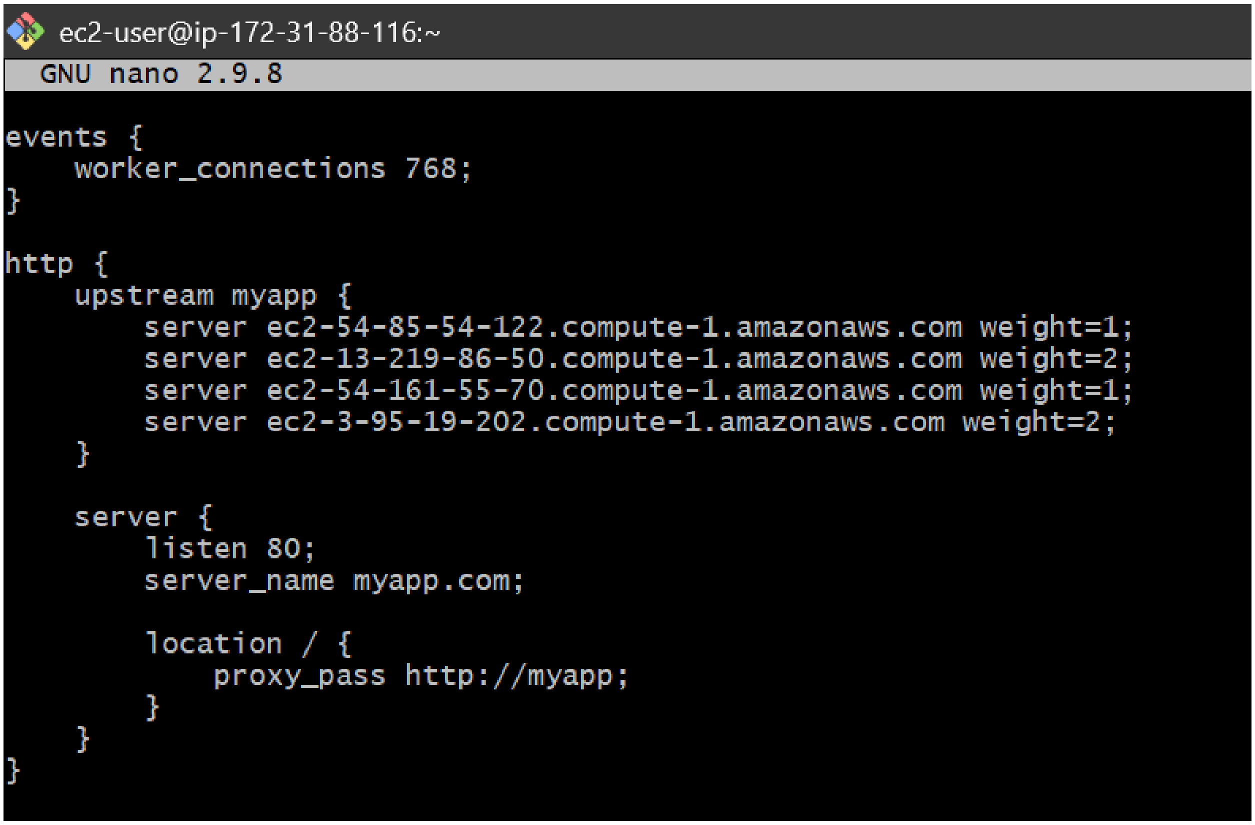
```
events {
    worker_connections 768;
}

http {
    upstream myapp {
        server ec2-54-85-54-122.compute-1.amazonaws.com weight=1;
        server ec2-13-219-86-50.compute-1.amazonaws.com weight=1;
        server ec2-54-161-55-70.compute-1.amazonaws.com weight=1;
        server ec2-3-95-19-202.compute-1.amazonaws.com weight=1;
    }

    server {
        listen 80;
        server_name myapp.com;

        location / {
            proxy_pass http://myapp;
        }
    }
}
```



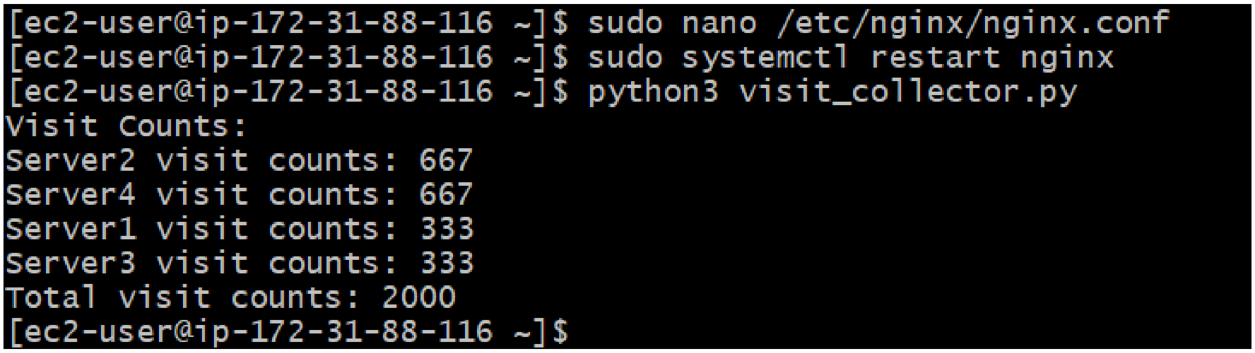
```
[ec2-user@ip-172-31-88-116 ~]$ sudo systemctl restart nginx
[ec2-user@ip-172-31-88-116 ~]$ python3 visit_collector.py
Visit Counts:
Server1 visit counts: 500
Server2 visit counts: 500
Server3 visit counts: 500
Server4 visit counts: 500
Total visit counts: 2000
[ec2-user@ip-172-31-88-116 ~]$
```

Scenario 3: Mixed Weight Distribution (Weight = 1, 2, 1, 2)

For this scenario, the weights were set as follows: Server 1 (weight 1), Server 2 (weight 2), Server 3 (weight 1), and Server 4 (weight 2). The traffic distribution was not as even as Scenario 1, but the traffic was distributed in a manner that reflected the weights. Servers 2 and 4, with higher weights, received a larger share of the traffic, while Servers 1 and 3 handled fewer requests. This scenario demonstrated the load balancer's ability to allocate traffic based on weighted values, with Servers 2 and 4 handling a larger portion of the load compared to Servers 1 and 3.



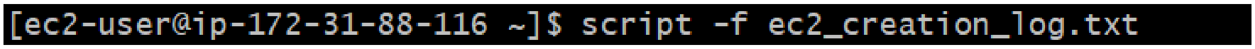
```
events {
    worker_connections 768;
}

http {
    upstream myapp {
        server ec2-54-85-54-122.compute-1.amazonaws.com weight=1;
        server ec2-13-219-86-50.compute-1.amazonaws.com weight=2;
        server ec2-54-161-55-70.compute-1.amazonaws.com weight=1;
        server ec2-3-95-19-202.compute-1.amazonaws.com weight=2;
    }

    server {
        listen 80;
        server_name myapp.com;

        location / {
            proxy_pass http://myapp;
        }
    }
}
```



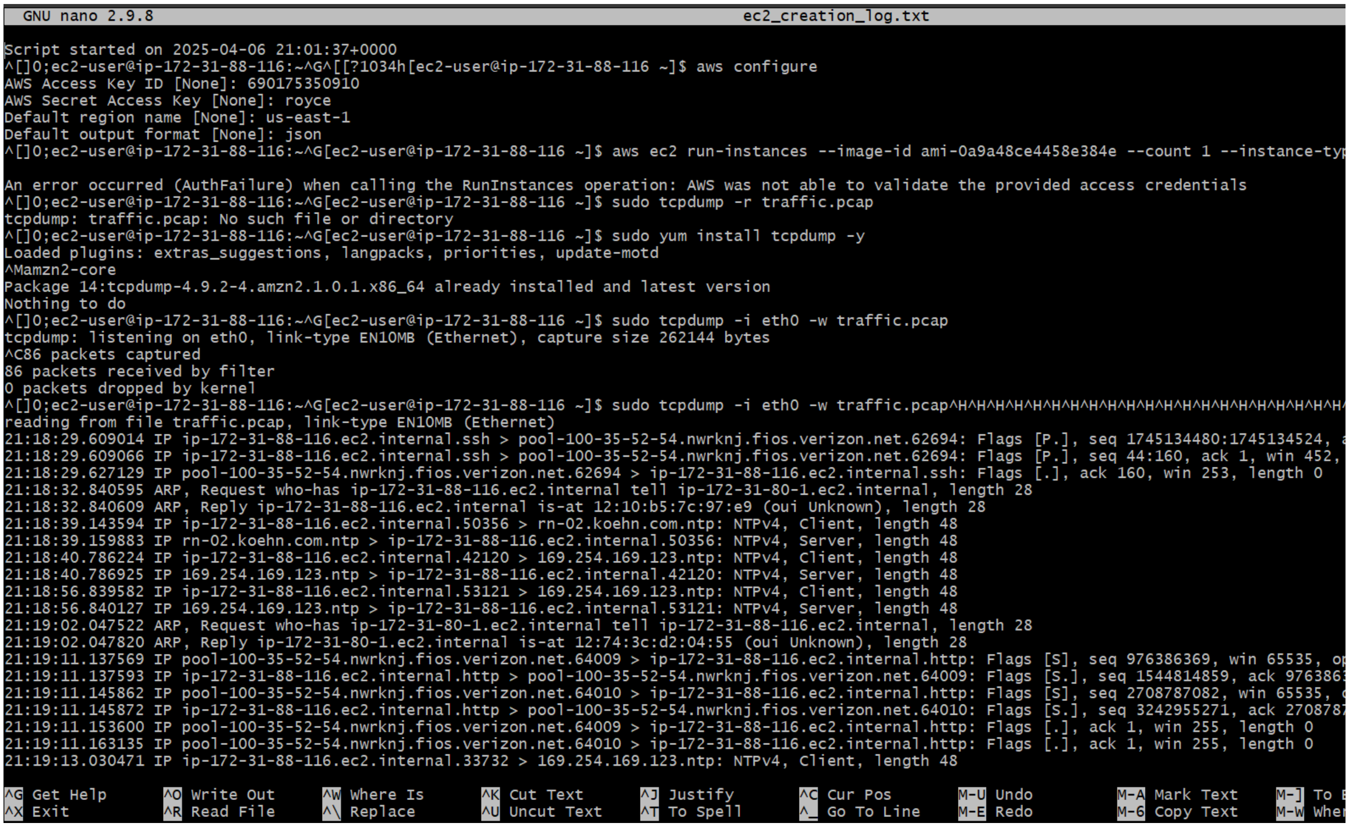
```
[ec2-user@ip-172-31-88-116 ~]$ sudo nano /etc/nginx/nginx.conf
[ec2-user@ip-172-31-88-116 ~]$ sudo systemctl restart nginx
[ec2-user@ip-172-31-88-116 ~]$ python3 visit_collector.py
Visit Counts:
Server2 visit counts: 667
Server4 visit counts: 667
Server1 visit counts: 333
Server3 visit counts: 333
Total visit counts: 2000
[ec2-user@ip-172-31-88-116 ~]$
```

Additional steps

6. Used the `script` command to record all the commands I used

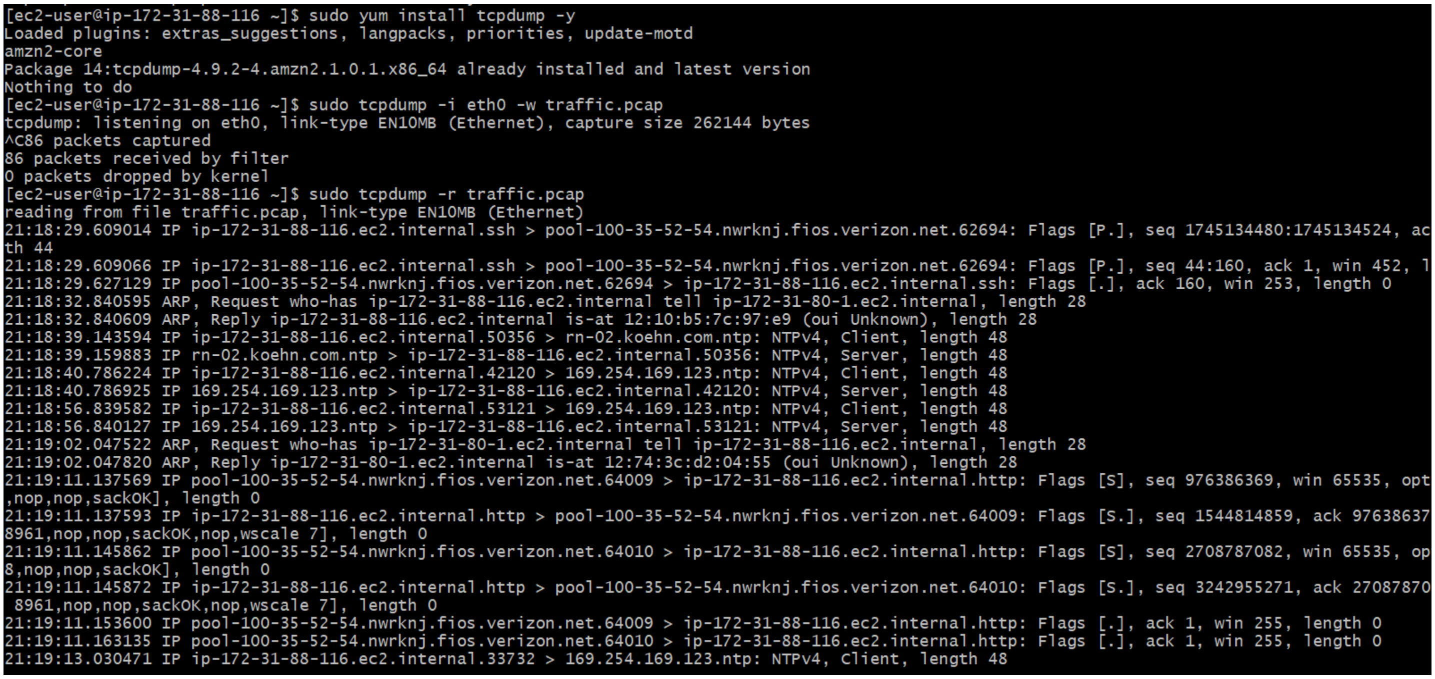


```
[ec2-user@ip-172-31-88-116 ~]$ script -f ec2_creation_log.txt
```



```
Script started on 2025-04-06 21:01:37+0000
^[]0;ec2-user@ip-172-31-88-116:~^G^[[?1034h[ec2-user@ip-172-31-88-116 ~]$ aws configure
AWS Access Key ID [None]: 690175350910
AWS Secret Access Key [None]: royce
Default region name [None]: us-east-1
Default output format [None]: json
^[]0;ec2-user@ip-172-31-88-116:~^G[ec2-user@ip-172-31-88-116 ~]$ aws ec2 run-instances --image-id ami-0a9a48ce4458e384e --count 1 --instance-typ

An error occurred (AuthFailure) when calling the RunInstances operation: AWS was not able to validate the provided access credentials
^[]0;ec2-user@ip-172-31-88-116:~^G[ec2-user@ip-172-31-88-116 ~]$ sudo tcpdump -r traffic.pcap
tcpdump: traffic.pcap: No such file or directory
^[]0;ec2-user@ip-172-31-88-116:~^G[ec2-user@ip-172-31-88-116 ~]$ sudo yum install tcpdump -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
^Mamzn2-core
Package 14:tcpdump-4.9.2-4.amzn2.1.0.1.x86_64 already installed and latest version
Nothing to do
^[]0;ec2-user@ip-172-31-88-116:~^G[ec2-user@ip-172-31-88-116 ~]$ sudo tcpdump -i eth0 -w traffic.pcap
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
^C86 packets captured
86 packets received by filter
0 packets dropped by kernel
^[]0;ec2-user@ip-172-31-88-116:~^G[ec2-user@ip-172-31-88-116 ~]$ sudo tcpdump -i eth0 -w traffic.pcap^H^H^H^H^H^H^H^H^H^H^H^H^H^H^H^H^H^H^H^H^
reading from file traffic.pcap, link-type EN10MB (Ethernet)
21:18:29.609014 IP ip-172-31-88-116.ec2.internal.ssh > pool-100-35-52-54.nwrknj.fios.verizon.net.62694: Flags [P.], seq 1745134480:1745134524, a
21:18:29.609066 IP ip-172-31-88-116.ec2.internal.ssh > pool-100-35-52-54.nwrknj.fios.verizon.net.62694: Flags [P.], seq 44:160, ack 1, win 452,
21:18:29.627129 IP pool-100-35-52-54.nwrknj.fios.verizon.net.62694 > ip-172-31-88-116.ec2.internal.ssh: Flags [.], ack 160, win 253, length 0
21:18:32.840595 ARP, Request who-has ip-172-31-88-116.ec2.internal tell ip-172-31-80-1.ec2.internal, length 28
21:18:32.840609 ARP, Reply ip-172-31-88-116.ec2.internal is-at 12:10:b5:7c:97:e9 (oui Unknown), length 28
21:18:39.143594 IP ip-172-31-88-116.ec2.internal.50356 > rn-02.koehn.com.ntp: NTPv4, Client, length 48
21:18:39.159883 IP rn-02.koehn.com.ntp > ip-172-31-88-116.ec2.internal.50356: NTPv4, Server, length 48
21:18:40.786224 IP ip-172-31-88-116.ec2.internal.42120 > 169.254.169.123.ntp: NTPv4, Client, length 48
21:18:40.786925 IP 169.254.169.123.ntp > ip-172-31-88-116.ec2.internal.42120: NTPv4, Server, length 48
21:18:56.839582 IP ip-172-31-88-116.ec2.internal.53121 > 169.254.169.123.ntp: NTPv4, Client, length 48
21:18:56.840127 IP 169.254.169.123.ntp > ip-172-31-88-116.ec2.internal.53121: NTPv4, Server, length 48
21:19:02.047522 ARP, Request who-has ip-172-31-80-1.ec2.internal tell ip-172-31-88-116.ec2.internal, length 28
21:19:02.047820 ARP, Reply ip-172-31-80-1.ec2.internal is-at 12:74:3c:d2:04:55 (oui Unknown), length 28
21:19:11.137569 IP pool-100-35-52-54.nwrknj.fios.verizon.net.64009 > ip-172-31-88-116.ec2.internal.http: Flags [S], seq 976386369, win 65535, op
21:19:11.137593 IP ip-172-31-88-116.ec2.internal.http > pool-100-35-52-54.nwrknj.fios.verizon.net.64009: Flags [S.], seq 1544814859, ack 976386
21:19:11.145862 IP pool-100-35-52-54.nwrknj.fios.verizon.net.64010 > ip-172-31-88-116.ec2.internal.http: Flags [S], seq 2708787082, win 65535, c
21:19:11.145872 IP ip-172-31-88-116.ec2.internal.http > pool-100-35-52-54.nwrknj.fios.verizon.net.64010: Flags [S.], seq 3242955271, ack 2708787
21:19:11.153600 IP pool-100-35-52-54.nwrknj.fios.verizon.net.64009 > ip-172-31-88-116.ec2.internal.http: Flags [.], ack 1, win 255, length 0
21:19:11.163135 IP pool-100-35-52-54.nwrknj.fios.verizon.net.64010 > ip-172-31-88-116.ec2.internal.http: Flags [.], ack 1, win 255, length 0
21:19:13.030471 IP ip-172-31-88-116.ec2.internal.33732 > 169.254.169.123.ntp: NTPv4, Client, length 48
```

7. To analyze the traffic, I used the tcpdump command (sudo tcpdump -i eth0 -w traffic.pcap) to capture all network packets exchanged between the load balancer and the web servers. This allowed me to observe the types of protocols being used and ensure proper communication between the instances.

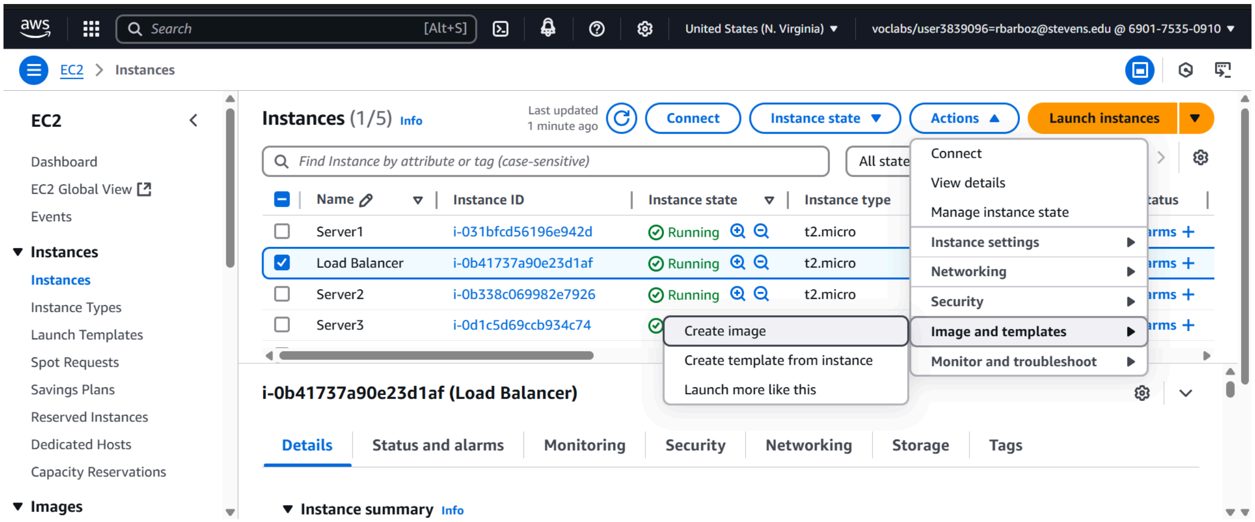


```
[ec2-user@ip-172-31-88-116 ~]$ sudo yum install tcpdump -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core
Package 14:tcpdump-4.9.2-4.amzn2.1.0.1.x86_64 already installed and latest version
Nothing to do
[ec2-user@ip-172-31-88-116 ~]$ sudo tcpdump -i eth0 -w traffic.pcap
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
^C86 packets captured
86 packets received by filter
0 packets dropped by kernel
[ec2-user@ip-172-31-88-116 ~]$ sudo tcpdump -r traffic.pcap
reading from file traffic.pcap, link-type EN10MB (Ethernet)
21:18:29.609014 IP ip-172-31-88-116.ec2.internal.ssh > pool-100-35-52-54.nwrknj.fios.verizon.net.62694: Flags [P.], seq 1745134480:1745134524, ac
th 44
21:18:29.609066 IP ip-172-31-88-116.ec2.internal.ssh > pool-100-35-52-54.nwrknj.fios.verizon.net.62694: Flags [P.], seq 44:160, ack 1, win 452, l
21:18:29.627129 IP pool-100-35-52-54.nwrknj.fios.verizon.net.62694 > ip-172-31-88-116.ec2.internal.ssh: Flags [.], ack 160, win 253, length 0
21:18:32.840595 ARP, Request who-has ip-172-31-88-116.ec2.internal tell ip-172-31-80-1.ec2.internal, length 28
21:18:32.840609 ARP, Reply ip-172-31-88-116.ec2.internal is-at 12:10:b5:7c:97:e9 (oui Unknown), length 28
21:18:39.143594 IP ip-172-31-88-116.ec2.internal.50356 > rn-02.koehn.com.ntp: NTPv4, Client, length 48
21:18:39.159883 IP rn-02.koehn.com.ntp > ip-172-31-88-116.ec2.internal.50356: NTPv4, Server, length 48
21:18:40.786224 IP ip-172-31-88-116.ec2.internal.42120 > 169.254.169.123.ntp: NTPv4, Client, length 48
21:18:40.786925 IP 169.254.169.123.ntp > ip-172-31-88-116.ec2.internal.42120: NTPv4, Server, length 48
21:18:56.839582 IP ip-172-31-88-116.ec2.internal.53121 > 169.254.169.123.ntp: NTPv4, Client, length 48
21:18:56.840127 IP 169.254.169.123.ntp > ip-172-31-88-116.ec2.internal.53121: NTPv4, Server, length 48
21:19:02.047522 ARP, Request who-has ip-172-31-80-1.ec2.internal tell ip-172-31-88-116.ec2.internal, length 28
21:19:02.047820 ARP, Reply ip-172-31-80-1.ec2.internal is-at 12:74:3c:d2:04:55 (oui Unknown), length 28
21:19:11.137569 IP pool-100-35-52-54.nwrknj.fios.verizon.net.64009 > ip-172-31-88-116.ec2.internal.http: Flags [S], seq 976386369, win 65535, opt
,nop,nop,sackOK], length 0
21:19:11.137593 IP ip-172-31-88-116.ec2.internal.http > pool-100-35-52-54.nwrknj.fios.verizon.net.64009: Flags [S.], seq 1544814859, ack 97638637
8961,nop,nop,sackOK,nop,wscale 7], length 0
21:19:11.145862 IP pool-100-35-52-54.nwrknj.fios.verizon.net.64010 > ip-172-31-88-116.ec2.internal.http: Flags [S], seq 2708787082, win 65535, op
8,nop,nop,sackOK], length 0
21:19:11.145872 IP ip-172-31-88-116.ec2.internal.http > pool-100-35-52-54.nwrknj.fios.verizon.net.64010: Flags [S.], seq 3242955271, ack 27087870
 8961,nop,nop,sackOK,nop,wscale 7], length 0
21:19:11.153600 IP pool-100-35-52-54.nwrknj.fios.verizon.net.64009 > ip-172-31-88-116.ec2.internal.http: Flags [.], ack 1, win 255, length 0
21:19:11.163135 IP pool-100-35-52-54.nwrknj.fios.verizon.net.64010 > ip-172-31-88-116.ec2.internal.http: Flags [.], ack 1, win 255, length 0
21:19:13.030471 IP ip-172-31-88-116.ec2.internal.33732 > 169.254.169.123.ntp: NTPv4, Client, length 48
```

Packet Capture Analysis Summary:

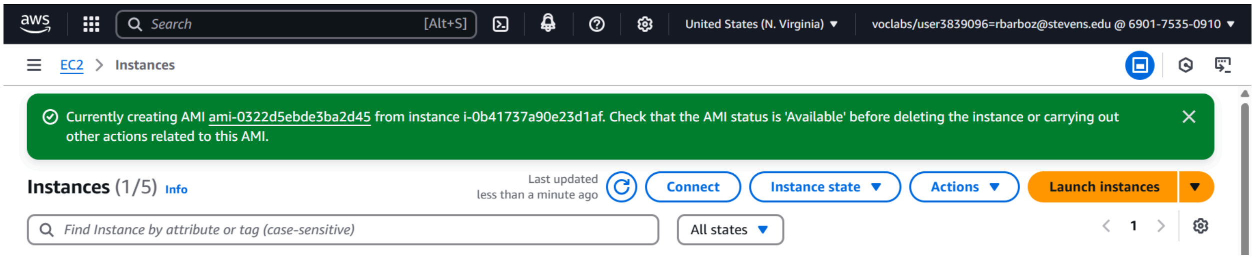
The captured packets show various types of network activity occurring on the EC2 instance. SSH communication is observed between the instance and a remote host, indicating secure remote access. ARP requests and replies show the process of the instance discovering MAC addresses of other devices on the local network. The system is also interacting with NTP servers to synchronize its time. HTTP requests reveal client-server communication over port 80, which is typical for web servers. DHCP traffic shows the instance requesting and receiving an IP address, ensuring it's connected to the network. Additionally, IPv6 DHCP requests indicate that the system is also seeking an IPv6 address. The TCP flags (SYN, ACK, FIN) demonstrate the regular connection setup and teardown processes. Overall, the packet capture confirms that the EC2 instance is performing standard network operations, including secure communication, time synchronization, and network configuration.

8. To create a backup of the load balancer instance, I registered an AMI from the instance. I navigated to the EC2 console, selected the load balancer instance, and created an image of it. This image will serve as a backup and can be used to restore the instance if needed.



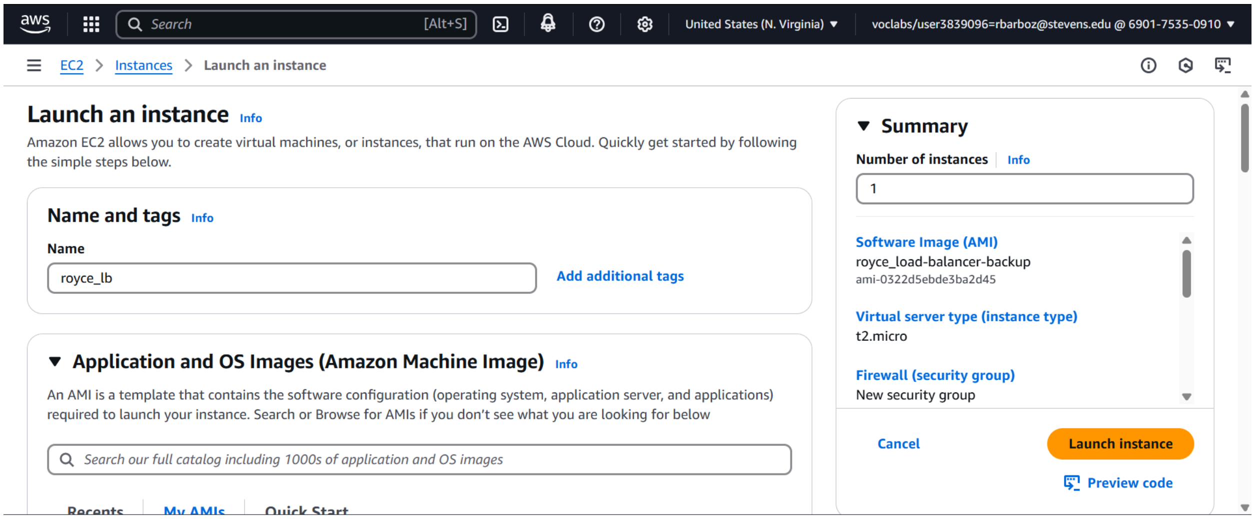
	Name	Instance ID	Instance state	Instance type
□	Server1	i-031bfcd56196e942d	Running	t2.micro
☑	Load Balancer	i-0b41737a90e23d1af	Running	t2.micro
□	Server2	i-0b338c069982e7926	Running	t2.micro
□	Server3	i-0d1c5d69ccb934c74		

i-0b41737a90e23d1af (Load Balancer)


aws
Search [Alt+S]
United States (N. Virginia) ▼
voclabs/user3839096=rbarboz@stevens.edu @ 6901-7535-0910 ▼
EC2 > Instances
Instances (1/5) Info
Last updated less than a minute ago
Connect
Instance state ▼
Actions ▼
Launch instances ▼
Find Instance by attribute or tag (case-sensitive)
All states ▼
1

Currently creating AMI ami-0322d5ebde3ba2d45 from instance i-0b41737a90e23d1af. Check that the AMI status is 'Available' before deleting the instance or carrying out other actions related to this AMI.

9. I launched a new EC2 instance using the AMI I created. This new instance was configured with the same settings and data as the original load balancer instance, ensuring that it can serve as a backup or replacement in case of failure. After launching the new instance, I verified that all files and configurations were intact



aws
Search [Alt+S]
United States (N. Virginia) ▾
voclabs/user3839096=rbarboz@stevens.edu @ 6901-7535-0910 ▾
EC2 > Instances > Launch an instance

Launch an instance Info

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags Info

Name: royce_lb

Add additional tags

▼ 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

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

Recents | My AMIs | Quick Start

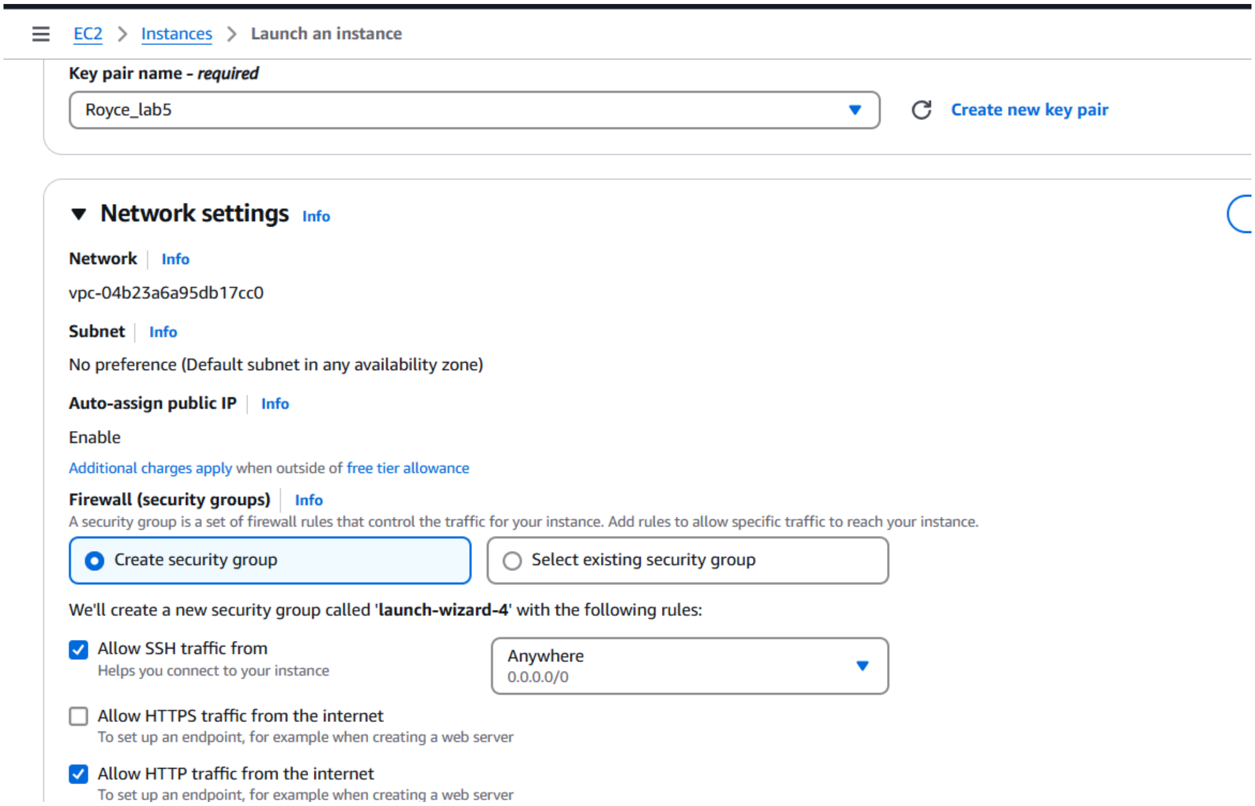
▼ Summary

Number of instances Info: 1

Software Image (AMI)
royce_load-balancer-backup
ami-0322d5ebde3ba2d45

Virtual server type (instance type)
t2.micro

Firewall (security group)
New security group



EC2 > Instances > Launch an instance

Key pair name - *required*

Royce_lab5 ▼ Create new key pair

▼ Network settings Info

Network Info

vpc-04b23a6a95db17cc0

Subnet Info

No preference (Default subnet in any availability zone)

Auto-assign public IP Info

Enable

Additional charges apply when outside of free tier allowance

Firewall (security groups) Info

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

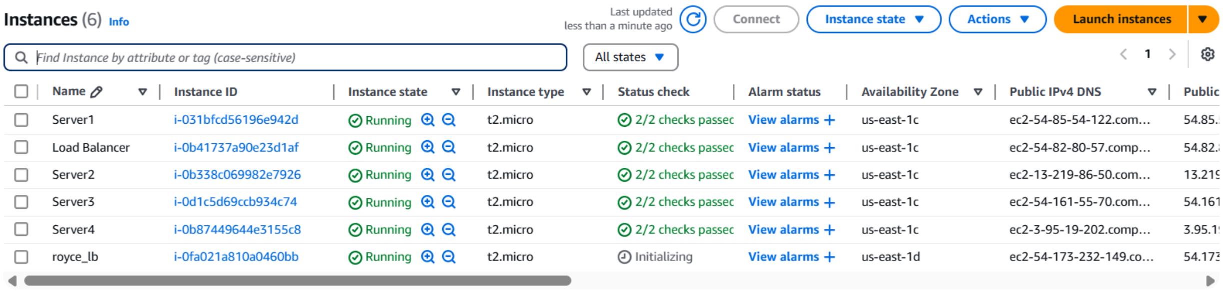
● Create security group ○ Select existing security group

We'll create a new security group called '**launch-wizard-4**' with the following rules:

☑ Allow SSH traffic from — Anywhere 0.0.0.0/0 ▼
Helps you connect to your instance

□ Allow HTTPS traffic from the internet
To set up an endpoint, for example when creating a web server

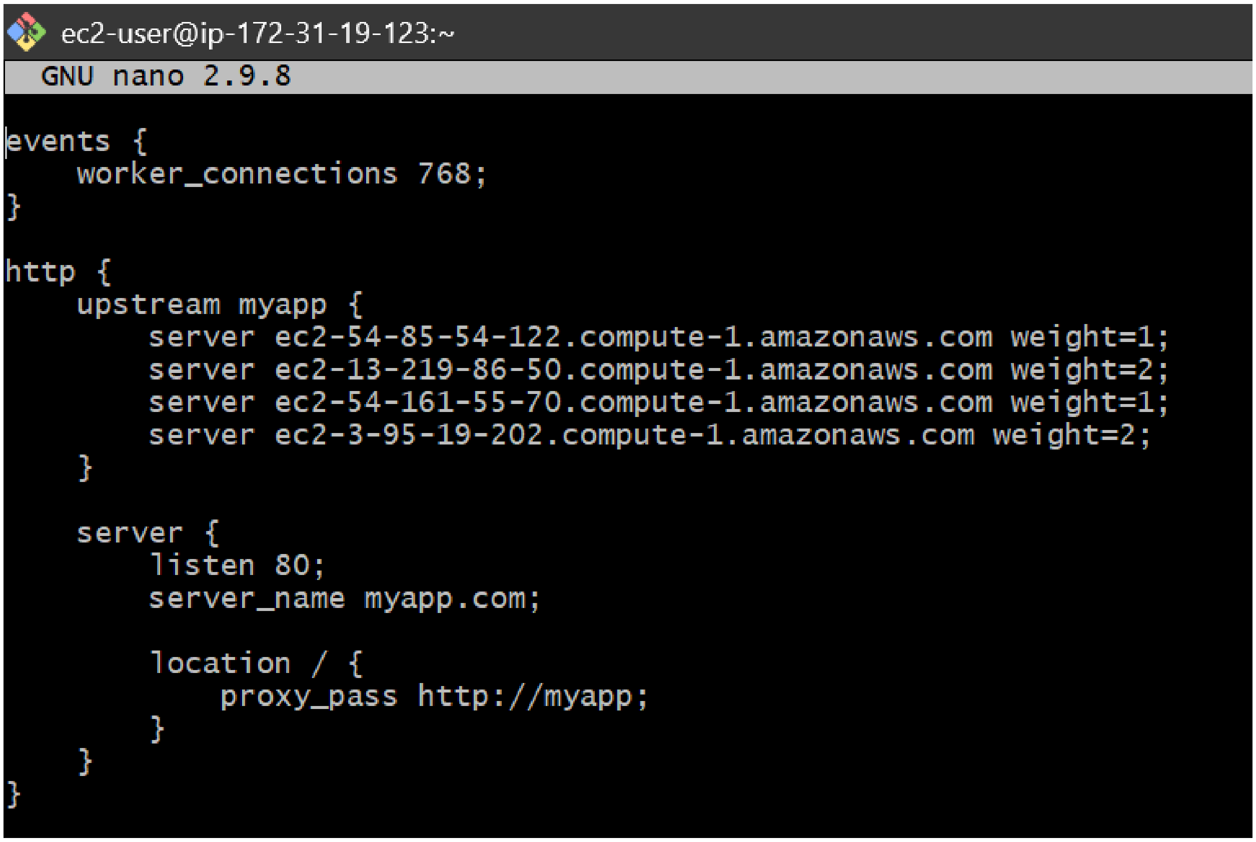
☑ Allow HTTP traffic from the internet
To set up an endpoint, for example when creating a web server



Instances (6) Info

Last updated less than a minute ago

□	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public
□	Server1	i-031bfcd56196e942d	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1c	ec2-54-85-54-122.com…	54.85.
□	Load Balancer	i-0b41737a90e23d1af	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1c	ec2-54-82-80-57.comp…	54.82.
□	Server2	i-0b338c069982e7926	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1c	ec2-13-219-86-50.com…	13.219
□	Server3	i-0d1c5d69ccb934c74	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1c	ec2-54-161-55-70.com…	54.161
□	Server4	i-0b87449644e3155c8	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1c	ec2-3-95-19-202.comp…	3.95.1
□	royce_lb	i-0fa021a810a0460bb	Running	t2.micro	Initializing	View alarms +	us-east-1d	ec2-54-173-232-149.co…	54.173

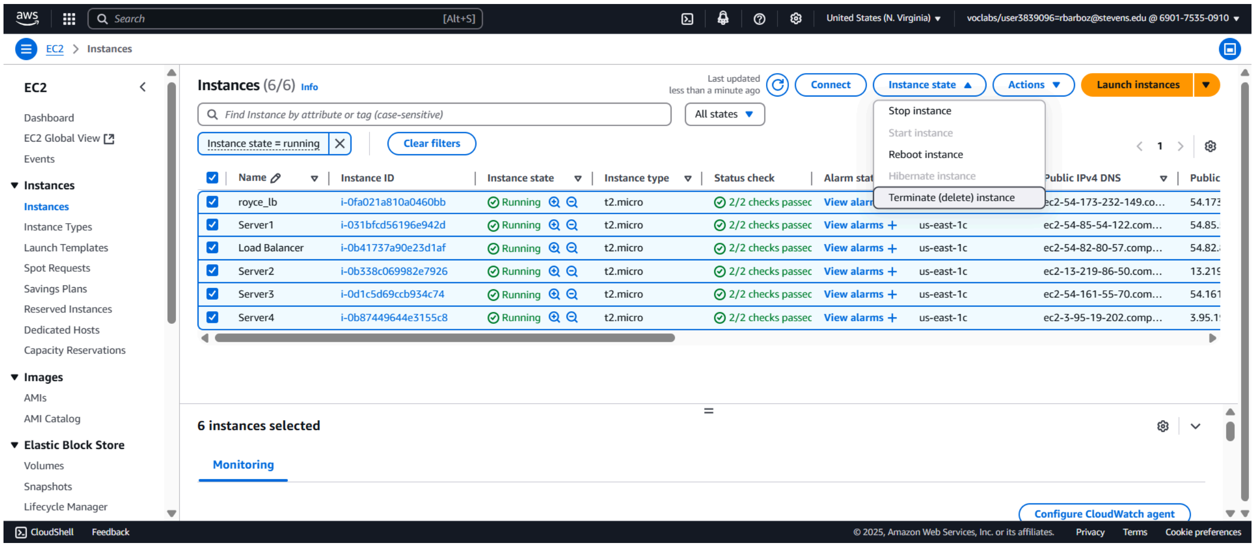


ec2-user@ip-172-31-19-123:~

GNU nano 2.9.8

```
events {
    worker_connections 768;
}

http {
    upstream myapp {
        server ec2-54-85-54-122.compute-1.amazonaws.com weight=1;
        server ec2-13-219-86-50.compute-1.amazonaws.com weight=2;
        server ec2-54-161-55-70.compute-1.amazonaws.com weight=1;
        server ec2-3-95-19-202.compute-1.amazonaws.com weight=2;
    }

    server {
        listen 80;
        server_name myapp.com;

        location / {
            proxy_pass http://myapp;
        }
    }
}
```

Terminated all the resources (EC2, security groups, key pairs, custom AMI from snapshot of Loadbalancer):

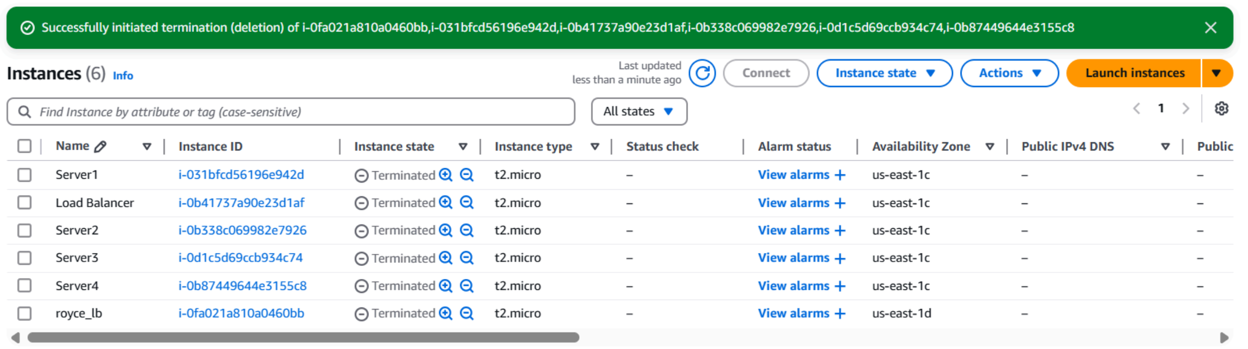


Instances (6/6) Info

Instance state = running

☑	Name	Instance ID	Instance state	Instance type	Status check	Alarm stat		Public IPv4 DNS	Public
☑	royce_lb	i-0fa021a810a0460bb	Running	t2.micro	2/2 checks passec	View alarr		ec2-54-173-232-149.co...	54.173
☑	Server1	i-031bfcd56196e942d	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1c	ec2-54-85-54-122.com...	54.85.
☑	Load Balancer	i-0b41737a90e23d1af	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1c	ec2-54-82-80-57.comp...	54.82.
☑	Server2	i-0b338c069982e7926	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1c	ec2-13-219-86-50.com...	13.219
☑	Server3	i-0d1c5d69ccb934c74	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1c	ec2-54-161-55-70.com...	54.161
☑	Server4	i-0b87449644e3155c8	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1c	ec2-3-95-19-202.comp...	3.95.1

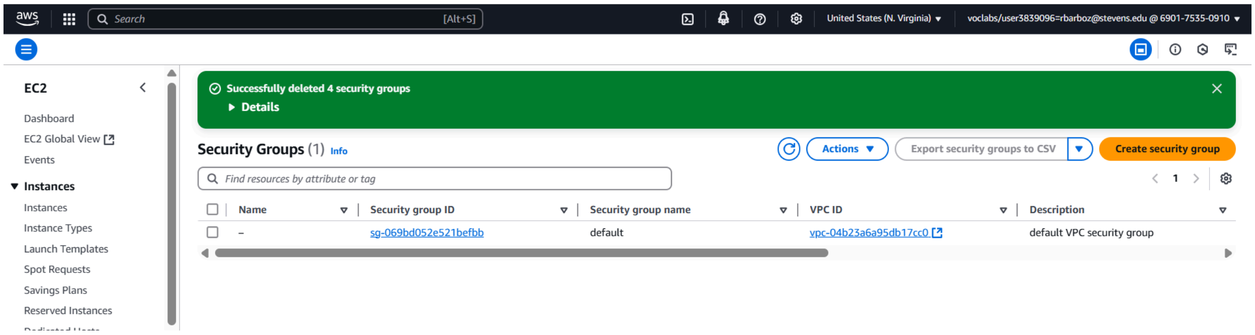
6 instances selected


Successfully initiated termination (deletion) of i-0fa021a810a0460bb,i-031bfcd56196e942d,i-0b41737a90e23d1af,i-0b338c069982e7926,i-0d1c5d69ccb934c74,i-0b87449644e3155c8

Instances (6)

Info
Last updated less than a minute ago
Connect
Instance state ▼
Actions ▼
Launch instances
Find Instance by attribute or tag (case-sensitive)
All states ▼
1

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public
Server1	i-031bfcd56196e942d	Terminated	t2.micro	–	View alarms +	us-east-1c	–	–
Load Balancer	i-0b41737a90e23d1af	Terminated	t2.micro	–	View alarms +	us-east-1c	–	–
Server2	i-0b338c069982e7926	Terminated	t2.micro	–	View alarms +	us-east-1c	–	–
Server3	i-0d1c5d69ccb934c74	Terminated	t2.micro	–	View alarms +	us-east-1c	–	–
Server4	i-0b87449644e3155c8	Terminated	t2.micro	–	View alarms +	us-east-1c	–	–
royce_lb	i-0fa021a810a0460bb	Terminated	t2.micro	–	View alarms +	us-east-1d	–	–

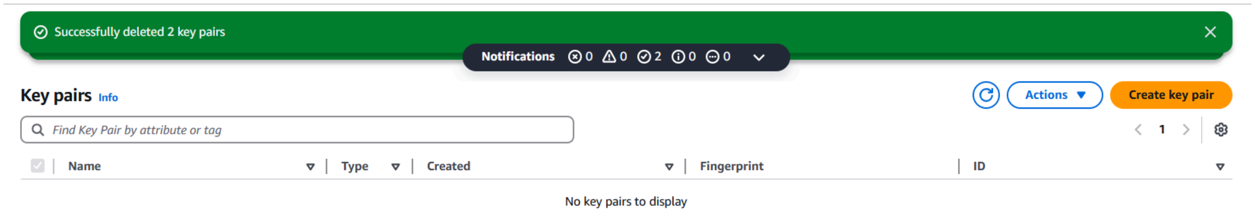


Successfully deleted 4 security groups

▸ Details

Security Groups (1)

Name	Security group ID	Security group name	VPC ID	Description
–	sg-069bd052e521befbb	default	vpc-04b23a6a95db17cc0	default VPC security group

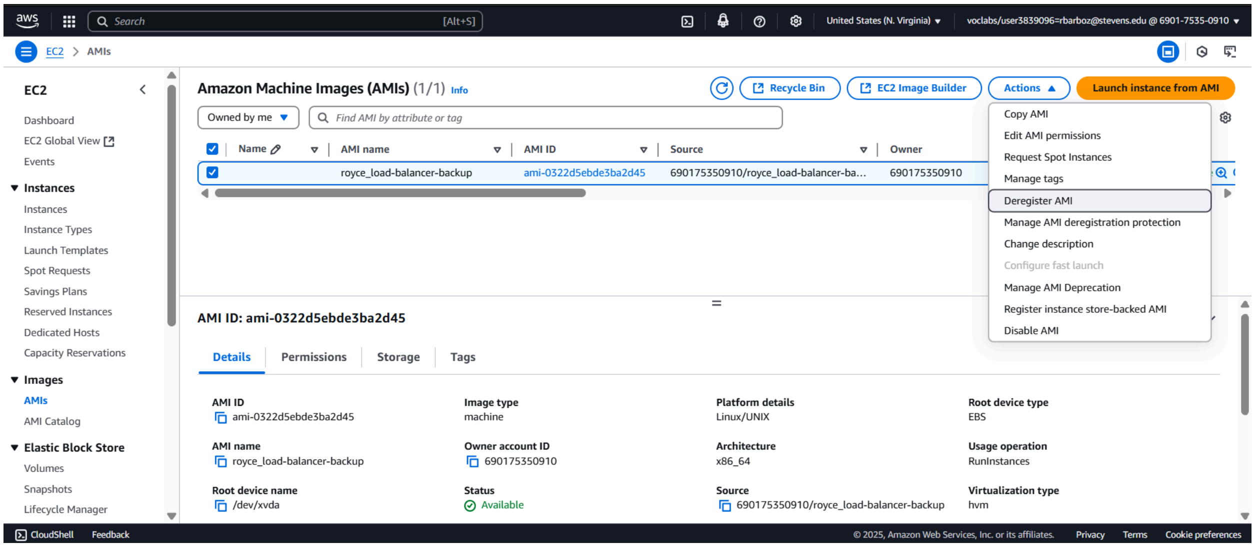

Successfully deleted 2 key pairs

Notifications ⊗0 ⚠0 ✓2 ⓘ0 ⊖0
Info
Actions ▼
Create key pair
Find Key Pair by attribute or tag
1

Key pairs

Name	Type	Created	Fingerprint	ID

No key pairs to display

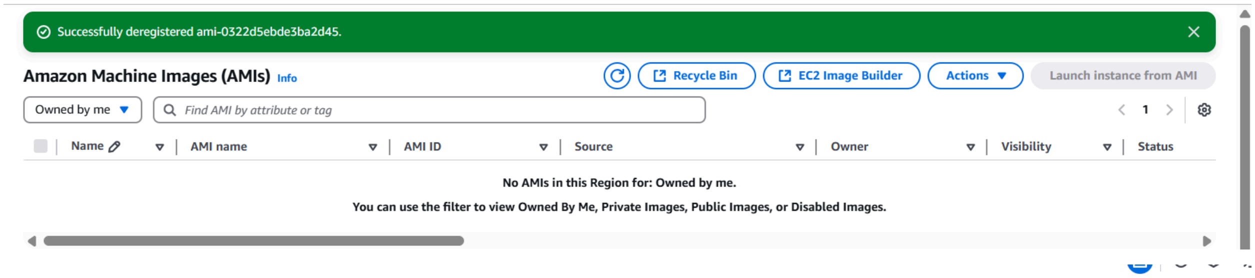


Amazon Machine Images (AMIs) (1/1)

☑	Name	AMI name	AMI ID	Source	Owner
☑		royce_load-balancer-backup	ami-0322d5ebde3ba2d45	690175350910/royce_load-balancer-ba...	690175350910

AMI ID: ami-0322d5ebde3ba2d45

AMI ID ami-0322d5ebde3ba2d45	Image type machine	Platform details Linux/UNIX	Root device type EBS
AMI name royce_load-balancer-backup	Owner account ID 690175350910	Architecture x86_64	Usage operation RunInstances
Root device name /dev/xvda	Status Available	Source 690175350910/royce_load-balancer-backup	Virtualization type hvm

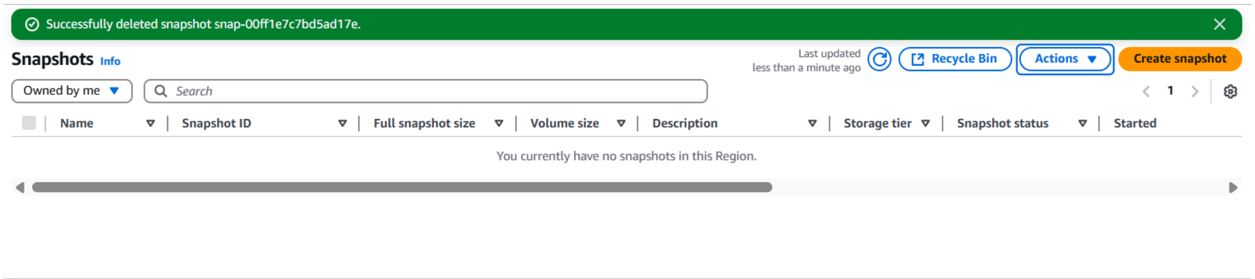


Successfully deregistered ami-0322d5ebde3ba2d45.

□	Name	AMI name	AMI ID	Source	Owner	Visibility	Status

No AMIs in this Region for: Owned by me.

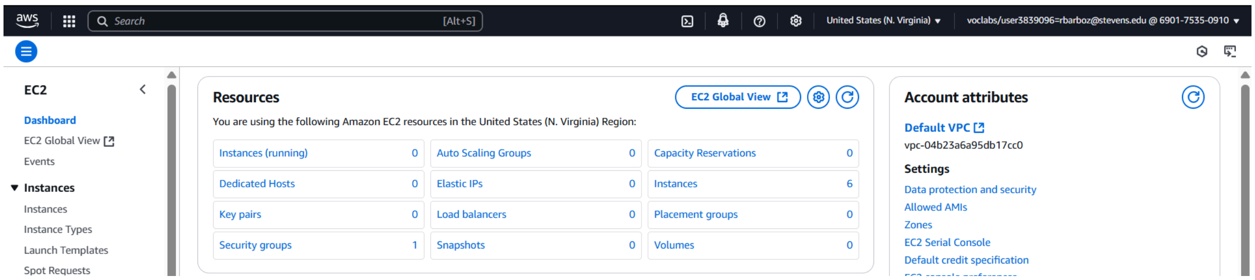
You can use the filter to view Owned By Me, Private Images, Public Images, or Disabled Images.



Successfully deleted snapshot snap-00ff1e7c7bd5ad17e.

□	Name	Snapshot ID	Full snapshot size	Volume size	Description	Storage tier	Snapshot status	Started

You currently have no snapshots in this Region.



Resources

You are using the following Amazon EC2 resources in the United States (N. Virginia) Region:

Instances (running)	0	Auto Scaling Groups	0	Capacity Reservations	0
Dedicated Hosts	0	Elastic IPs	0	Instances	6
Key pairs	0	Load balancers	0	Placement groups	0
Security groups	1	Snapshots	0	Volumes	0

Account attributes

Default VPC

vpc-04b23a6a95db17cc0

Settings

Data protection and security
Allowed AMIs
Zones
EC2 Serial Console
Default credit specification