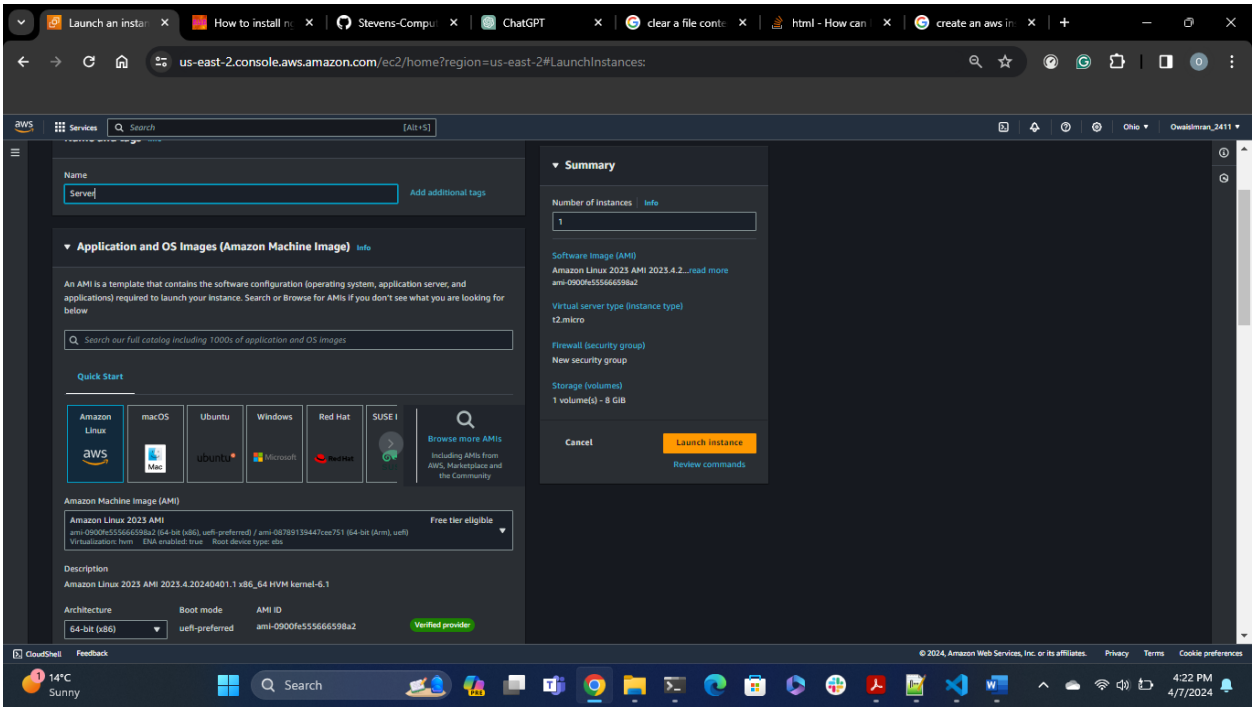
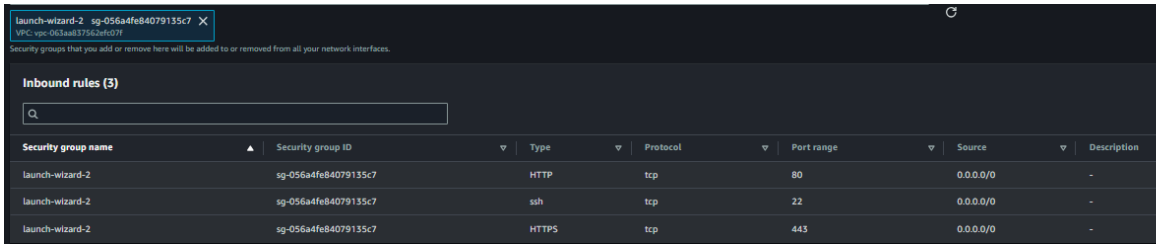
1. Launch an AWS Instance for Servers
    i. Name: Server1
    ii. AMI: Amazon Linux 2023 AMI
    iii. Architecture: 64bit (x86)



    iv. Instance Type: t2.micro (Free tier eligible)
    v. Key Pair Name: Owais_L02 (this was reused from previous lab)
    vi. Security group with the below config:



    With this configuration:
        There are three types of inbound rules:
        1. SSH at port 22 from anywhere on the internet (0.0.0.0/0)
        2. HTTP at port 80 from anywhere on the internet (0.0.0.0/0)
        3. HTTPS at port 443 from anywhere on the internet (0.0.0.0/0)
    vii. Launch the instance, using the aws cli command
    ``` aws ec2 run-instances --image-id ami-0900fe555666598a2 --security-group-ids sg-056a4fe84079135c7 --count 5 --instance-type t2.micro --key-name Owais_L02 ```
    After launching, Connect to the launched instance using SSH or AWS EC2 instance connect and execute the following commands:

```
sudo dnf install nginx -y

sudo cd /usr/share/nginx/html
sudo sed -i 's/Welcome to nginx/Owais Nginx SERVER_NUMBER/g' ./index.html
sudo systemctl start nginx
sudo systemctl enable nginx
```
```

```
sudo systemctl status nginx
```

Step by step output of each command is:

1. `sudo dnf install nginx -y #this command will install nginx on instance`

```
(3/7): gperftools-libs-2.9.1-1.amzn2023.0.3.x86_64.rpm
(4/7): nginx-1.24.0-1.amzn2023.0.2.x86_64.rpm
(5/7): nginx-filesystem-1.24.0-1.amzn2023.0.2.noarch.rpm
(6/7): nginx-mimetypes-2.1.49-3.amzn2023.0.3.noarch.rpm
(7/7): nginx-core-1.24.0-1.amzn2023.0.2.x86_64.rpm
--------------------------------------------------------------------------------
Total
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :
  Running scriptlet: nginx-filesystem-1:1.24.0-1.amzn2023.0.2.noarch
  Installing       : nginx-filesystem-1:1.24.0-1.amzn2023.0.2.noarch
  Installing       : nginx-mimetypes-2.1.49-3.amzn2023.0.3.noarch
  Installing       : libunwind-1.4.0-5.amzn2023.0.2.x86_64
  Installing       : gperftools-libs-2.9.1-1.amzn2023.0.3.x86_64
  Installing       : nginx-core-1:1.24.0-1.amzn2023.0.2.x86_64
  Installing       : generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch
  Installing       : nginx-1:1.24.0-1.amzn2023.0.2.x86_64
  Running scriptlet: nginx-1:1.24.0-1.amzn2023.0.2.x86_64
  Verifying        : generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch
  Verifying        : gperftools-libs-2.9.1-1.amzn2023.0.3.x86_64
  Verifying        : libunwind-1.4.0-5.amzn2023.0.2.x86_64
  Verifying        : nginx-1:1.24.0-1.amzn2023.0.2.x86_64
  Verifying        : nginx-core-1:1.24.0-1.amzn2023.0.2.x86_64
  Verifying        : nginx-filesystem-1:1.24.0-1.amzn2023.0.2.noarch
  Verifying        : nginx-mimetypes-2.1.49-3.amzn2023.0.3.noarch

Installed:
  generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch  gperftools-libs-2.9.1-1.amzn2023.0.3.x86_64   libunwind-1.4.0-5.amzn2023
  nginx-filesystem-1:1.24.0-1.amzn2023.0.2.noarch    nginx-mimetypes-2.1.49-3.amzn2023.0.3.noarch

Complete!
```

2. `sudo cd /usr/share/nginx/html #this site holds the host page of the html server when accessed from the internet. Update the title and h1 tag with the Server name tag to distinguish between different servers from the load balancer.`
3. `sudo sed -i 's/Welcome to nginx/Owais Nginx SERVER_NAME/g' ./index.html # this command replaces the Welcome to nginx with Owais Nginx SERVER_NAME in the file index.html`

```
[ec2-user@ip-172-31-26-240 ~]$ cat /usr/share/nginx/html/index.html
<!DOCTYPE html>
<html>
<head>
<title>Owais Nginx LoadBalancer!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Owais Nginx LoadBalancer!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

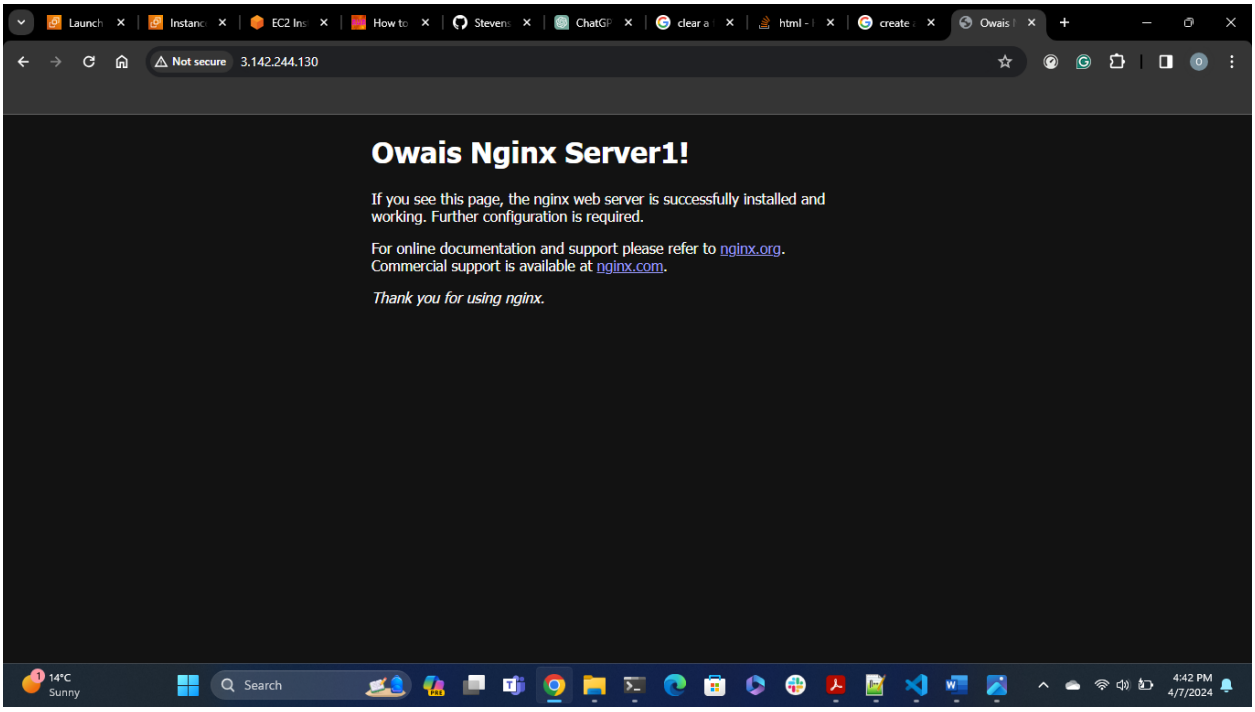
<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
[ec2-user@ip-172-31-26-240 ~]$
```

4. `sudo systemctl start nginx # this command starts the nginx service`
5. `sudo systemctl enable nginx #this commands enable the nginx service everytime the instance reboots`

```
[ec2-user@ip-172-31-26-240 ~]$ sudo systemctl start nginx
[ec2-user@ip-172-31-26-240 ~]$ sudo systemctl enable nginx
Created symlink /etc/systemd/system/multi-user.target.wants/nginx.service → /usr/lib/systemd/system/ngi
```

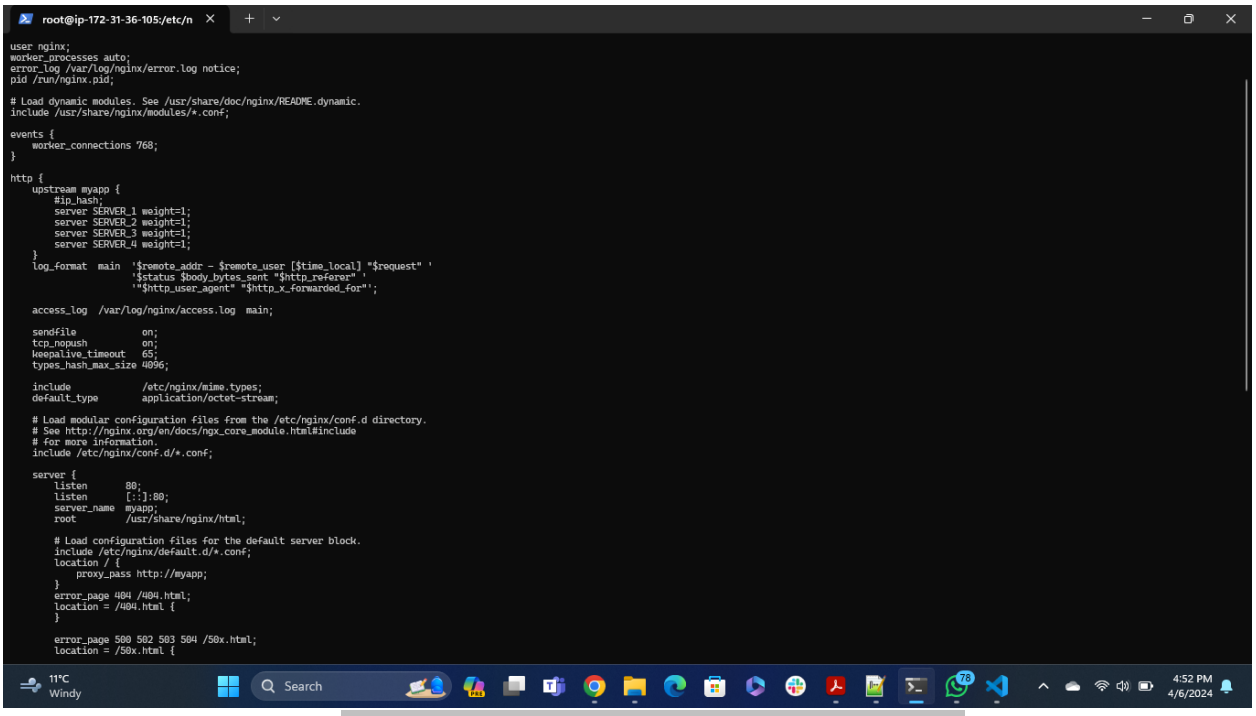
After executing the above code, your nginx service is accessible on the public IP address of your EC2 instance (screenshot attached). Repeat the above steps for configuring Server2, Server3, Server4.



Load Balancer configuration has a slightly different procedure which is given below:

1. execute the command ```` ```sudo dnf install nginx -y``` ````
2. Use any text editor and edit the file located at ```` ```/etc/nginx/nginx.conf``` ````, and make the following changes:
    a. Change the value of server connections to 768. (This can be found under the events indentation)
    b. Add an upsteam block under http indentation
    upstream myapp {
        #ip_hash
        server SERVER1_PUBLIC_DNS weight=1
        server SERVER2_PUBLIC_DNS weight=1
        server SERVER3_PUBLIC_DNS weight=1
        server SERVER4_PUBLIC_DNS weight=1
    }
    c. Replace the server name under server indentation as myapp.com and add a location block as
    location / {
      proxy_pass http://myapp;
    }

3. After modification the file should look something as:



4. Execute the command ```` ```sudo systemctl start nginx && sudo systemctl enable nginx``` ````
5. Load Balancer is now configured.
   NOTE: SERVER_1 SERVER_2 SERVER_3 SERVER_4 should be replaced with the respective IP Address or Public DNS Name of each EC2 instance.

METRICS COLLECTION:

The following code was used to collect metrics, the code is written in Python:

```
import lxml.html
counter = {
    "Owais Nginx Server1!": 0,
    "Owais Nginx Server2!": 0,
    "Owais Nginx Server3!": 0,
    "Owais Nginx Server4!": 0,
}
print(counter)
for i in range(2000):
    t = lxml.html.parse("http://ec2-3-14-8-219.us-east-2.compute.amazonaws.com")
    title = t.find(".//title").text
    counter[title]+=1
print("=== Usage Statistics ===")
for key in counter:
    print("Server Name: ", key, '--- TOTAL VISITS: ', counter[key])
```

```
for changing weights in each distribution execute the following commands and edit the weight value as specified in upstream block against each server:

1. sudo vi /etc/config/nginx.conf #modify the weights in this file
2. sudo systemctl reload nginx #this command will restart the service of nginx
3. wait for a few minutes so that the changes are propagated and execute the above python code to collect metrics.
```

4. **CONFIG 1: (EQUAL WEIGHT DISTRIBUTION)**

```
http {
    upstream myapp {
        #ip_hash
        server ec2-18-218-2-103.us-east-2.compute.amazonaws.com weight=1;
        server ec2-18-222-145-176.us-east-2.compute.amazonaws.com weight=1;
        server ec2-3-22-99-174.us-east-2.compute.amazonaws.com weight=1;
        server ec2-13-59-85-227.us-east-2.compute.amazonaws.com weight=1;
    }
    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
```

```
{'Owais Nginx Server1!': 0, 'Owais Nginx Server2!': 0, 'Owais Nginx Server3!': 0, 'Owais Nginx Server4!': 0}
=== Usage Statistics ===
Server Name:  Owais Nginx Server1! --- TOTAL VISITS:  500
Server Name:  Owais Nginx Server2! --- TOTAL VISITS:  500
Server Name:  Owais Nginx Server3! --- TOTAL VISITS:  500
Server Name:  Owais Nginx Server4! --- TOTAL VISITS:  500
```

5. **CONFIG 2: (WEIGHT=1 FOR SERVER1, WEIGHT=2 FOR SERVER2, WEIGHT=3 FOR SERVER3, WEIGHT=4 FOR SERVER4)**

```
http {
    upstream myapp {
        #ip_hash
        server ec2-18-218-2-103.us-east-2.compute.amazonaws.com weight=1;
        server ec2-18-222-145-176.us-east-2.compute.amazonaws.com weight=2;
        server ec2-3-22-99-174.us-east-2.compute.amazonaws.com weight=3;
        server ec2-13-59-85-227.us-east-2.compute.amazonaws.com weight=4;
    }
    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';
```

```
{'Owais Nginx Server1!': 0, 'Owais Nginx Server2!': 0, 'Owais Nginx Server3!': 0, 'Owais Nginx Server4!': 0}
=== Usage Statistics ===
Server Name:  Owais Nginx Server1! --- TOTAL VISITS:  200
Server Name:  Owais Nginx Server2! --- TOTAL VISITS:  400
Server Name:  Owais Nginx Server3! --- TOTAL VISITS:  600
Server Name:  Owais Nginx Server4! --- TOTAL VISITS:  800
```

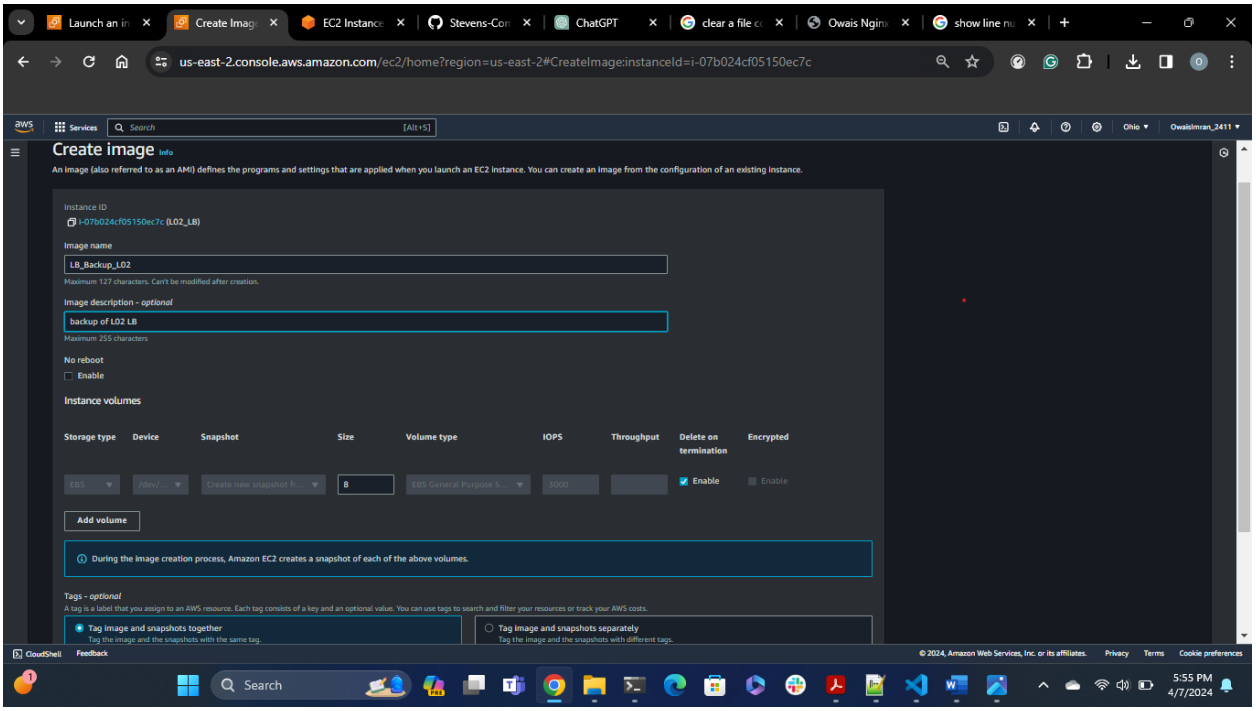
6. **CONFIG 3: (WEIGHT=1 FOR SERVER 1 AND SERVER3, WEIGHT=2 FOR SERVER2 AND SERVER4)**

```
http {
    upstream myapp {
        #ip_hash
        server ec2-18-218-2-103.us-east-2.compute.amazonaws.com weight=1;
        server ec2-18-222-145-176.us-east-2.compute.amazonaws.com weight=2;
        server ec2-3-22-99-174.us-east-2.compute.amazonaws.com weight=1;
        server ec2-13-59-85-227.us-east-2.compute.amazonaws.com weight=2;
    }
    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';
```

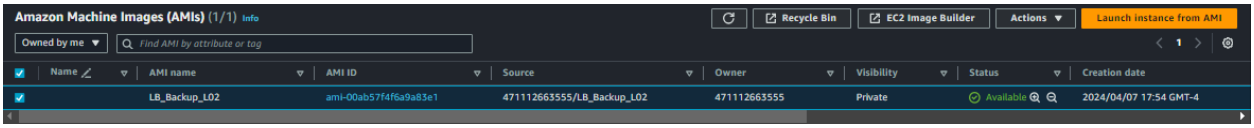
```
{'Owais Nginx Server1!': 0, 'Owais Nginx Server2!': 0, 'Owais Nginx Server3!': 0, 'Owais Nginx Server4!': 0}
=== Usage Statistics ===
Server Name:  Owais Nginx Server1! --- TOTAL VISITS:  333
Server Name:  Owais Nginx Server2! --- TOTAL VISITS:  667
Server Name:  Owais Nginx Server3! --- TOTAL VISITS:  333
Server Name:  Owais Nginx Server4! --- TOTAL VISITS:  667
```

Backup & Restore an AMI:

1. Select the EC2 instance to backup, under the Actions Menu, Go to Image and templates, and Create Image, fill in the desired field.



2. Now to restore instance from an AMI, under AMI sections of EC2 instance, select the AMI you want to restore from and click on Launch Instance from AMI. Provide Network & Security information i.e. Key Pair and Security Group and Click on Launch instance. All instance related information will be made available from the AMI backup



3. Verify that all the files are already present. Connect to the newly launched instance, and we will cross check if the last configuration is already present for the nginx load balancer.

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

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

events {
    worker_connections 768;
}

http {
    upstream myapp {
        #ip_hash
        server ec2-18-222-211-108.us-east-2.compute.amazonaws.com weight=1;
        server ec2-18-118-47-0.us-east-2.compute.amazonaws.com weight=2;
        server ec2-3-140-192-51.us-east-2.compute.amazonaws.com weight=1;
        server ec2-18-116-61-124.us-east-2.compute.amazonaws.com weight=2;
    }
    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;
    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.
```

i-030f78e6fa8aa8484

PublicIPs: 18.118.213.13 PrivateIPs: 172.31.32.94

All configuration was backed up, it means the restore and backup was successful.