Nginx Load Balancer Setup on EC2 (Task 3)

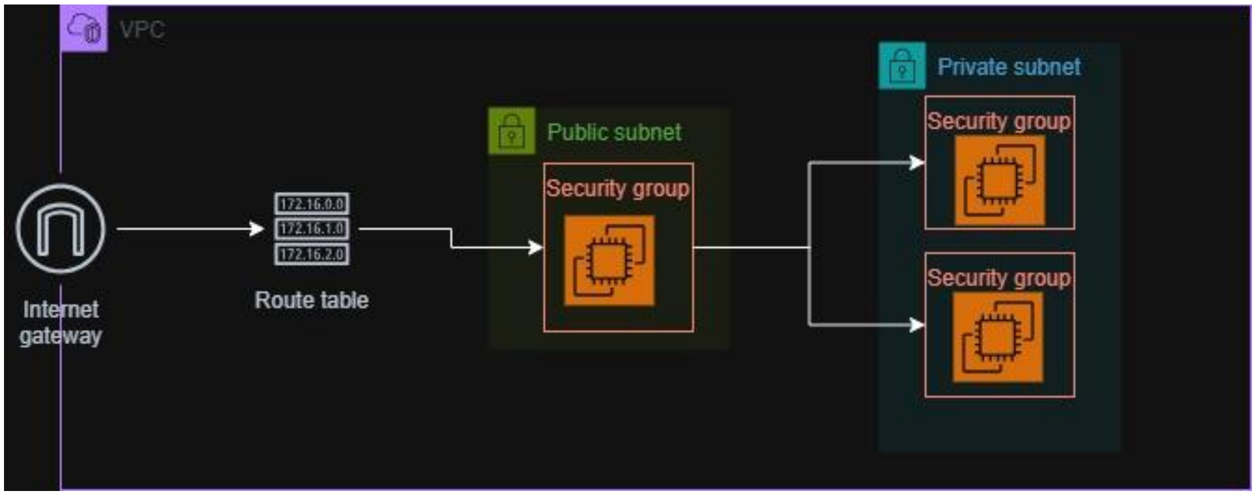
Zaeem Attique Ashar

Cloud Intern

Task Description:

The project involves setting up a NGINX load balancer on an EC2 instance to distribute traffic to two backend EC2 servers running simple web servers. It includes launching and configuring all EC2 instances, installing and configuring NGINX on the load balancer, and specifying backend server addresses for load distribution. After setup, the load balancer is tested to ensure traffic is properly routed between backends.

Architecture Diagram:



Task3.0: Deploying the initial infrastructure

1. VPC Configuration:
   - CIDR: 10.0.0.0/16.
2. Subnets Configuration:

   For public subnet:

   - CIDR Block for subnet: 10.0.1.0/24.
   - Availability Zone: us-west-2a.

   For private subnet:

   - CIDR Block for subnet: 10.0.2.0/24.
   - Availability Zone: us-west-2a.
3. Internet Gateway Configuration:
   - VPC ID.
4. Route Table (Public):
   - VPC ID.
   - Route: Destination 0.0.0.0/0, Target Internet Gateway.
   - Local routing route entry automatically created for private IP.
5. Route Table (Private):
   - VPC ID.
   - Route: Destination 0.0.0.0/0, Target NAT Gateway.
   - Local routing route entry automatically created for private IP.
6. Security Group (Public):
   - Public Instance for Nginx Server:
     - Inbound SSH Traffic: Port 22, source 0.0.0.0/0
     - Inbound HTTP Traffic: Port 80, source 0.0.0.0/0
   - Private Instance for Backend Servers:
     - Inbound HTTP Traffic: Port 80, source PublicInstanceSG
     - Outbound Any Traffic: destination 0.0.0.0/0, destination NGW
7. NAT Gateway for Private Instance:
   - Allocate EIP.
   - Subnet: Public Subnet.

Task3.1: Backend Instances for running the webserver

Backend Server 1 Deployment and Configuration:

AMI: Amazon Linux 2023

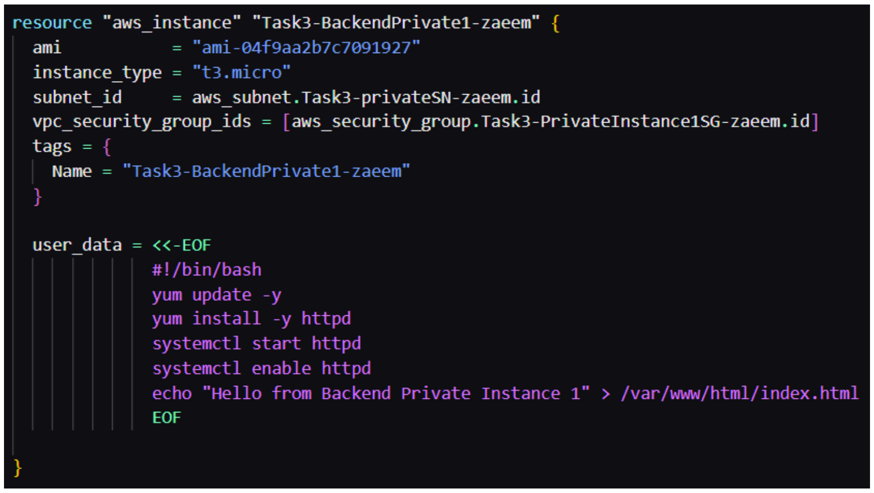
Instance Type: t3.micro

Subnet: Public Subnet ID

Security Group: Private Instance Security Group 1

User Data Script Steps:

- Update the yum repository
- Install httpd server for serving content
- Start the httpd service
- Enable auto-start httpd service upon boot
- Echo Server 1 identification text



```
resource "aws_instance" "Task3-BackendPrivate1-zaeem" {
  ami           = "ami-04f9aa2b7c7091927"
  instance_type = "t3.micro"
  subnet_id     = aws_subnet.Task3-privateSN-zaeem.id
  vpc_security_group_ids = [aws_security_group.Task3-PrivateInstance1SG-zaeem.id]
  tags = {
    Name = "Task3-BackendPrivate1-zaeem"
  }

  user_data = <<-EOF
              #!/bin/bash
              yum update -y
              yum install -y httpd
              systemctl start httpd
              systemctl enable httpd
              echo "Hello from Backend Private Instance 1" > /var/www/html/index.html
              EOF

}
```

Backend Server 2 Deployment and Configuration:

AMI: Amazon Linux 2023

Instance Type: t3.micro

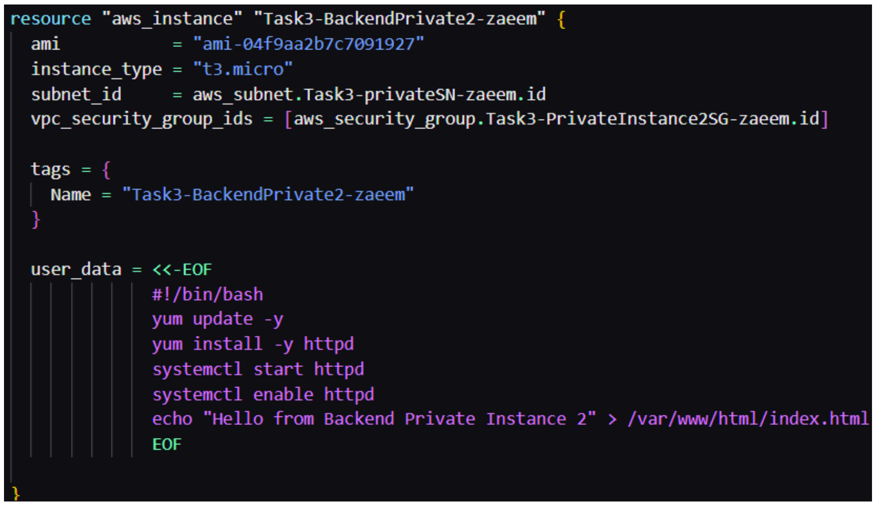
Subnet: Public Subnet ID

Security Group: Private Instance Security Group 1

User Data Script Steps:

- Update the yum repository
- Install httpd server for serving content
- Start the httpd service

- Enable auto-start httpd service upon boot
- Echo Server 1 identification text



```
resource "aws_instance" "Task3-BackendPrivate2-zaeem" {
  ami           = "ami-04f9aa2b7c7091927"
  instance_type = "t3.micro"
  subnet_id     = aws_subnet.Task3-privateSN-zaeem.id
  vpc_security_group_ids = [aws_security_group.Task3-PrivateInstance2SG-zaeem.id]

  tags = {
    Name = "Task3-BackendPrivate2-zaeem"
  }

  user_data = <<-EOF
            #!/bin/bash
            yum update -y
            yum install -y httpd
            systemctl start httpd
            systemctl enable httpd
            echo "Hello from Backend Private Instance 2" > /var/www/html/index.html
            EOF

}
```

Task3.2: Launch NGINX Load Balancer EC2 Instance:

AMI: Amazon Linux 2023

Instance Type: t3.micro

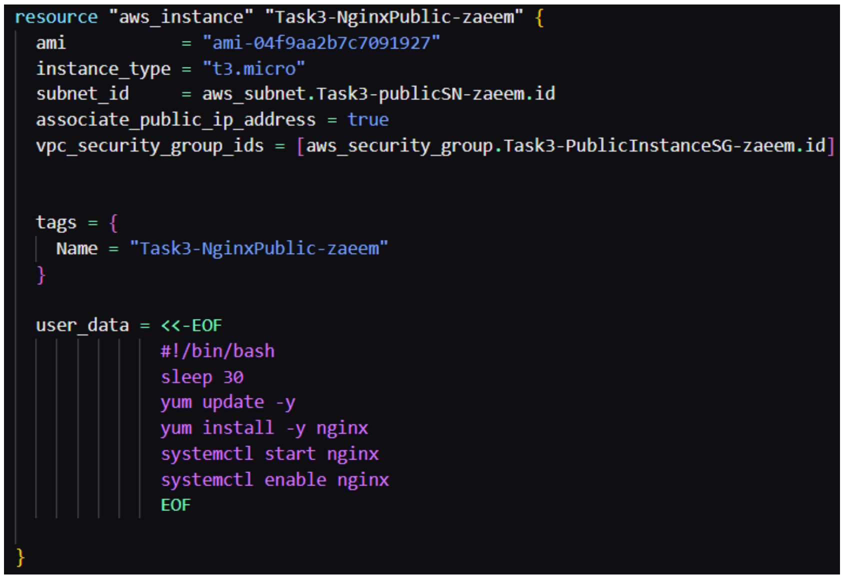
Subnet: Public Subnet ID

Associate Public IP: True

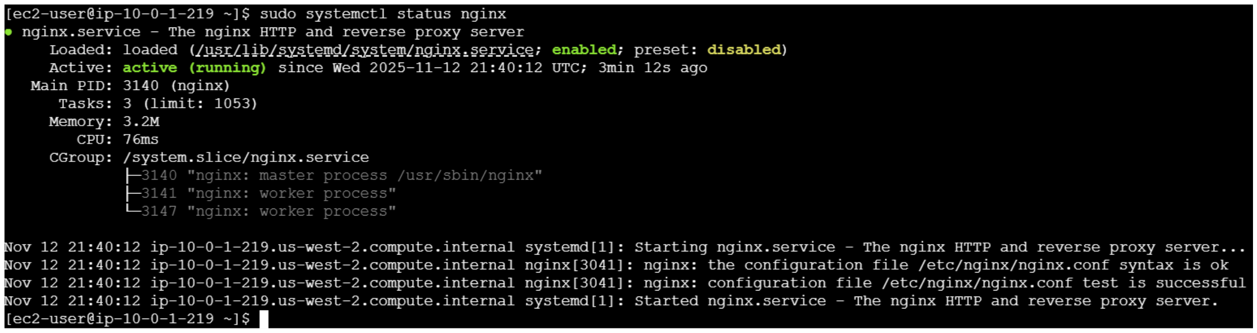
Security Group: Public Instance Security Group ID

User Data Script Steps:

- Sleep 30 to wait for network infrastructure to be deployed
- Update the yum repository
- Install nginx server for load balancing
- Start the nginx service
- Enable auto-start nginx service upon boot


```
resource "aws_instance" "Task3-NginxPublic-zaeem" {
  ami           = "ami-04f9aa2b7c7091927"
  instance_type = "t3.micro"
  subnet_id     = aws_subnet.Task3-publicSN-zaeem.id
  associate_public_ip_address = true
  vpc_security_group_ids = [aws_security_group.Task3-PublicInstanceSG-zaeem.id]


  tags = {
    Name = "Task3-NginxPublic-zaeem"
  }

  user_data = <<-EOF
              #!/bin/bash
              sleep 30
              yum update -y
              yum install -y nginx
              systemctl start nginx
              systemctl enable nginx
              EOF

}
```

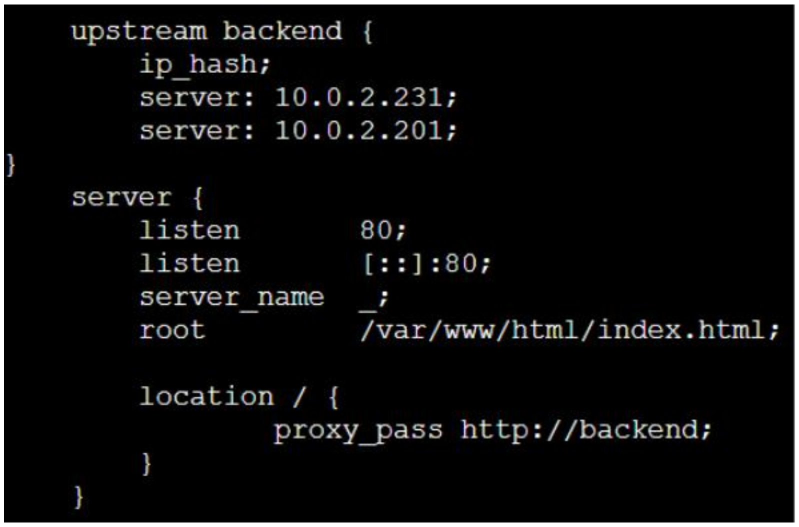
- Make sure the Nginx Server is running by using the command:
  Sudo systemctl status nginx


```
[ec2-user@ip-10-0-1-219 ~]$ sudo systemctl status nginx
● nginx.service - The nginx HTTP and reverse proxy server
     Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; preset: disabled)
     Active: active (running) since Wed 2025-11-12 21:40:12 UTC; 3min 12s ago
   Main PID: 3140 (nginx)
      Tasks: 3 (limit: 1053)
     Memory: 3.2M
        CPU: 76ms
     CGroup: /system.slice/nginx.service
             ├─3140 "nginx: master process /usr/sbin/nginx"
             ├─3141 "nginx: worker process"
             └─3147 "nginx: worker process"

Nov 12 21:40:12 ip-10-0-1-219.us-west-2.compute.internal systemd[1]: Starting nginx.service - The nginx HTTP and reverse proxy server...
Nov 12 21:40:12 ip-10-0-1-219.us-west-2.compute.internal nginx[3041]: nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
Nov 12 21:40:12 ip-10-0-1-219.us-west-2.compute.internal nginx[3041]: nginx: configuration file /etc/nginx/nginx.conf test is successful
Nov 12 21:40:12 ip-10-0-1-219.us-west-2.compute.internal systemd[1]: Started nginx.service - The nginx HTTP and reverse proxy server.
[ec2-user@ip-10-0-1-219 ~]$
```

- Open the nginx.conf file located in the /etc/nginx/ directory using the command:

  Sudo nano /etc/nginx/nginx.conf

- First define the backend servers that will be receiving the proxy requests inside the upstream block. Use the server directive to define each server.
- Use the private IP addresses of the servers as shown in the example below.
- Now use the proxy pass directive under the location block to define where the client requests will be passed. Use the name assigned to the upstream block, e.g. 'backend' used in the example shown.
- Finally, add the name of the distribution algorithm such as ip_hash, least_conn. If we do not mention any specific algorithm, then by default the round robin algorithm is used.



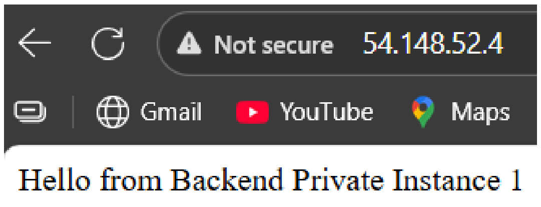
```
    upstream backend {
        ip_hash;
        server: 10.0.2.231;
        server: 10.0.2.201;
}

    server {
        listen       80;
        listen       [::]:80;
        server_name  _;
        root         /var/www/html/index.html;

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

Here are a few different distribution algorithms that can be used:

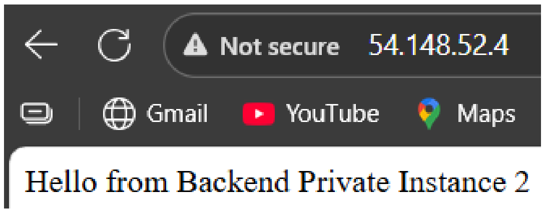
- **Round Robin:** Distribute traffic one after the other. Loop back to the first server after the last one is done. Useful for when the servers have identical resources. It doesn't need to be configured as it is the default load balancing algo.
- **Least Conn:** Traffic is forwarded to the server that has the least active connections. This helps distribute the load evenly when the connection TTL varies.
- **IP Hash:** Requests from the same client's IP are always sent to the same backend server. This is how sticky sessions work to preserve user sessions.
- **Weighted Round Robin:** Traffic is distributed in a sequence, one after the other but with weight. Meaning that the server assigned more weight will get more requests.
- **Backup Servers:** Traffic is only distributed to backup servers when no other server responds (is down).

Task3.3: Testing the Load Balancer.

We can test that the load balancer is working by entering the IP address of the nginx server in the browser and refreshing multiple times.



Hello from Backend Private Instance 1



Hello from Backend Private Instance 2

This means that Nginx Load Balancer is working perfectly by routing the traffic to both of the instances.

Task3.4: Challenges Faced

1. Every time we make any changes inside the main nginx.conf file or any other custom configuration file that alters the website or its behavior, we must always reload the nginx service to make the changes reflect. Use the following command to do so:
   Sudo systemctl reload nginx.service

2. It is always a good practice to check if the syntax of the configuration files to not cause any downtime because if we apply a file with syntax errors then the client will have to face downtime. Use the following command:
   Sudo nginx –t

3. If we use the dnf service in our userdata script to install a few packages we need to let the instance sleep for 30 seconds before the script is run because if the script runs before the network infrastructure is provisioned then it will fail silently and there will be no output neither will the packages be installed.

4. If we install Nginx server from the apt repository in a Debian distribution, the default server block configuration will be stored inside the default file located at /etc/nginx/sites-available/default. But when we install Nginx Server from the Amazon Linux repository, the server block will not be stored separately. It will be in the main configuration file at the location /etc/nginx/nginx.conf.