AWS Network Load Balancer (NLB) - Port Based Routing Project

==========

This project demonstrates how to deploy two websites on separate EC2 instances with different ports (80 and 8080) and configure an AWS Network Load Balancer (NLB) to perform port-based routing.

Project Architecture

- EC2 Instance 1: Hosts website on Port 80
- EC2 Instance 2: Hosts website on Port 8080
- Security Group: Allows All Traffic, HTTP (80), and Custom TCP (8080)
- Target Groups: Separate target groups for each port
- NLB: Routes incoming traffic based on the port number
- Port 80 → Target Group for EC2 Instance 1
- Port 8080 → Target Group for EC2 Instance 2

Steps to Deploy

---

1. Launch EC2 Instances
- Launch two EC2 instances using Amazon Linux 2.
- Configure Security Group to allow:
* HTTP (80)
* Custom TCP (8080)
* All Traffic (for testing purpose only; restrict in production)

Install a simple web server on each instance:

```bash
# On EC2 Instance 1 (Port 80)
sudo yum install -y httpd
sudo systemctl start httpd
sudo systemctl enable httpd
echo "Welcome to Website on Port 80" | sudo tee /var/www/html/index.html

# On EC2 Instance 2 (Port 8080)
sudo yum install -y httpd
sudo systemctl start httpd
sudo systemctl enable httpd
echo "Welcome to Website on Port 8080" | sudo tee /var/www/html/index.html
```

2. Create Target Groups

- Create two target groups:

* TG-80 for Port 80 (EC2 Instance 1)

* TG-8080 for Port 8080 (EC2 Instance 2)

- Register the respective instances in each target group.

3. Create Network Load Balancer (NLB)

- Go to EC2 → Load Balancers → Create Load Balancer → Network Load Balancer.

- Add Listeners:

* Port 80 → TG-80

* Port 8080 → TG-8080

- Assign Security Groups allowing 80 and 8080.

4. Test the Setup

- Copy the DNS name of the NLB.

- Open in browser:

* http://:80 → Should display "Welcome to Website on Port 80"

* http://:8080 → Should display "Welcome to Website on Port 8080"

Key AWS Services Used

- EC2 – Hosting the websites

- Security Groups – Controlling traffic access

- Target Groups – Grouping EC2 instances per port

- Network Load Balancer – Port-based routing

Conclusion

This project demonstrates a simple implementation of port-based routing using AWS NLB with two backend EC2 instances on different ports.

Bash Script for EC2 Configuration

```
#!/bin/bash

yum update -y
yum install -y httpd

# Enable and start the default Apache (port 80)
systemctl enable httpd
systemctl start httpd

# Create website directories
mkdir -p /var/www/html/app1
mkdir -p /var/www/html/app2

# Add sample content
echo "<h1>This is Website 1 on Port 80</h1>" > /var/www/html/app1/index.html
echo "<h1>This is Website 2 on Port 8080</h1>" > /var/www/html/app2/index.html

# Configure default Apache to serve /app1
cat > /etc/httpd/conf.d/app1.conf <<EOF
Alias /app1 /var/www/html/app1
<Directory /var/www/html/app1>
Require all granted
</Directory>
EOF

systemctl restart httpd

# Prepare second Apache instance for port 8080
cp -a /etc/httpd /etc/httpd-8080
cp /usr/lib/systemd/system/httpd.service /etc/systemd/system/httpd-8080.service

# Create separate log and run directories
mkdir -p /var/log/httpd-8080
mkdir -p /var/run/httpd-8080

# Modify httpd-8080.conf
sed -i 's/^Listen 80/Listen 8080/' /etc/httpd-8080/conf/httpd.conf
sed -i 's|^DocumentRoot ".*"|DocumentRoot "/var/www/html/app2"|'
/etc/httpd-8080/conf/httpd.conf
sed -i 's|<Directory ".*">|<Directory "/var/www/html/app2">|' /etc/httpd-8080/conf/httpd.conf
echo 'PidFile /var/run/httpd-8080/httpd.pid' >> /etc/httpd-8080/conf/httpd.conf
echo 'ErrorLog /var/log/httpd-8080/error_log' >> /etc/httpd-8080/conf/httpd.conf
echo 'CustomLog /var/log/httpd-8080/access_log combined' >> /etc/httpd-8080/conf/httpd.conf

# Update systemd service for 8080
cat > /etc/systemd/system/httpd-8080.service <<EOF
[Unit]
Description=Apache HTTP Server on port 8080
After=network.target

[Service]
```

```
Type=forking
PIDFile=/var/run/httpd-8080/httpd.pid
ExecStart=/usr/sbin/httpd -f /etc/httpd-8080/conf/httpd.conf -k start
ExecReload=/usr/sbin/httpd -f /etc/httpd-8080/conf/httpd.conf -k graceful
ExecStop=/usr/sbin/httpd -f /etc/httpd-8080/conf/httpd.conf -k stop
PrivateTmp=true

[Install]
WantedBy=multi-user.target
EOF

# Reload systemd and start second Apache instance
systemctl daemon-reexec
systemctl daemon-reload
systemctl enable httpd-8080
systemctl start httpd-8080
```