**AWS Network Load Balancer (NLB) -Detailed Step-by-Step Guide**

**Table of Contents**

1. Introduction to AWS Load Balancers
2. Prerequisites
3. Step 1: VPC Setup
4. Step 2: Security Group Configuration
5. Step 3: Launch EC2 Instances
6. Step 4: Create Target Group
7. Step 5: Configure Network Load Balancer
8. Step 6: SSL/TLS Setup with ACM
9. Step 7: Route 53 DNS Configuration
10. Step 8: Testing & Validation
11. Step 9: Cross-Zone Load Balancing
12. Cleanup
13. Real-World Use Cases
14. FAQ

**1. Introduction to AWS Load Balancers**

AWS offers **three types** of load balancers:

1. **Application Load Balancer (ALB)** - Layer 7 (HTTP/HTTPS)
2. **Network Load Balancer (NLB)** - Layer 4 (TCP/UDP)
3. **Classic Load Balancer (CLB)** - Legacy (Avoid for new deployments)

**Today’s Focus:**  
✔ **Network Load Balancer (NLB)** - Best for **high-performance, low-latency** applications (Gaming, VoIP, Financial Transactions).  
✔ Works on **TCP/UDP** (Layer 4).  
✔ Supports **static IPs** and **Elastic IPs**.  
✔ **No HTTP/HTTPS processing** (Passes traffic as-is).

**2. Prerequisites**

* **AWS Account** (Free Tier eligible)
* **Domain Name** (Optional, for HTTPS setup)
* **Basic AWS EC2 & VPC Knowledge**

**3. Step 1: VPC Setup**

**A. Create a Custom VPC**

1. Go to **VPC Dashboard → Create VPC**
   * **Name tag:** nlb-vpc
   * **IPv4 CIDR:** 10.0.0.0/16
   * **Enable DNS hostnames:** Yes

**B. Create Subnets**

| **Subnet Name** | **AZ** | **CIDR Block** | **Type** |
| --- | --- | --- | --- |
| nlb-subnet-1a | us-east-1a | 10.0.1.0/24 | Private |
| nlb-subnet-1b | us-east-1b | 10.0.2.0/24 | Private |
| nlb-subnet-1c | us-east-1c | 10.0.3.0/24 | Private |
| nlb-public-1a | us-east-1a | 10.0.4.0/24 | Public (For NAT Gateway) |

**C. Set Up NAT Gateway**

1. **Allocate Elastic IP**
2. **Create NAT Gateway** in nlb-public-1a
3. **Edit Route Tables** for private subnets to route 0.0.0.0/0 → NAT Gateway

**4. Step 2: Security Group Configuration**

Create a security group (nlb-sg) with the following rules:

| **Type** | **Protocol** | **Port Range** | **Source** | **Purpose** |
| --- | --- | --- | --- | --- |
| SSH | TCP | 22 | My IP | Secure Shell Access |
| HTTP | TCP | 80 | 0.0.0.0/0 | Web Traffic |
| HTTPS | TCP | 443 | 0.0.0.0/0 | Secure Web Traffic |
| Custom TCP | TCP | 1024-65535 | nlb-sg | Ephemeral Ports for NLB |

**5. Step 3: Launch EC2 Instances**

**A. Launch 3 Instances (1 per AZ)**

* **AMI:** Amazon Linux 2
* **Instance Type:** t2.micro
* **Subnet:** nlb-subnet-1a, nlb-subnet-1b, nlb-subnet-1c
* **Security Group:** nlb-sg
* **User Data (Bootstrap Script):**

bash

#!/bin/bash

yum update -y

yum install -y nginx

systemctl start nginx

echo "<h1>Server $(hostname) - AZ: $(curl -s http://169.254.169.254/latest/meta-data/placement/availability-zone)</h1>" > /usr/share/nginx/html/index.html

**B. Verify Instances**

bash

curl http://<EC2-Private-IP>

Expected Output:

<h1>Server ip-10-0-1-10 - AZ: us-east-1a</h1>

**6. Step 4: Create Target Group**

1. **EC2 → Target Groups → Create Target Group**
   * **Name:** nlb-tg
   * **Protocol:** TCP
   * **Port:** 80
   * **VPC:** nlb-vpc
   * **Health Check Path:** /
2. **Register Instances:** Select all 3 EC2 instances.

**7. Step 5: Configure Network Load Balancer**

1. **EC2 → Load Balancers → Create NLB**
   * **Name:** nlb-demo
   * **Scheme:** Internet-facing
   * **Listeners:**
     + **TCP:80 → Forward to**nlb-tg
     + **TLS:443 → Forward to**nlb-tg (Requires ACM Certificate)
   * **Availability Zones:** Select all 3 public subnets (nlb-public-1a, nlb-public-1b, nlb-public-1c)

**8. Step 6: SSL/TLS Setup with ACM**

1. **Request ACM Certificate**
   * **Domain Name:** \*.cloudvishwakarma.in
   * **Validation Method:** DNS (Route 53)
2. **Validate via Route 53** (Automated if using AWS DNS)

**9. Step 7: Route 53 DNS Configuration**

1. **Create Hosted Zone** (cloudvishwakarma.in)
2. **Add A Record:**
   * **Name:** www
   * **Alias:** Yes → Point to NLB DNS

**10. Step 8: Testing & Validation**

**A. Check Load Balancing**

bash

for i in {1..10}; do curl -s http://www.cloudvishwakarma.in | grep "Server"; done

Expected Output (Different AZs respond):

Copy

Download

Server ip-10-0-1-10 - AZ: us-east-1a

Server ip-10-0-2-20 - AZ: us-east-1b

Server ip-10-0-3-30 - AZ: us-east-1c

**B. Verify HTTPS**

Open https://www.cloudvishwakarma.in → Check SSL Certificate.

**11. Step 9: Cross-Zone Load Balancing**

1. **Edit NLB Attributes** → Enable **Cross-Zone Load Balancing**
2. **Verify Traffic Distribution:**

bash

watch -n 1 'curl -s http://www.cloudvishwakarma.in | grep "Server"'

**12. Cleanup**

1. **Delete NLB**
2. **Terminate EC2 Instances**
3. **Release Elastic IPs**
4. **Delete VPC**

**13. Real-World Use Cases**

| **Scenario** | **Why NLB?** |
| --- | --- |
| **Online Gaming** | Low-latency UDP traffic |
| **VoIP (Zoom, Teams)** | Real-time audio/video |
| **Financial Transactions** | High-throughput TCP |

**14. FAQ**

**Q: Can NLB do HTTPS termination?**  
A: No, use **ALB** or **CloudFront** for HTTPS offloading.

**Q: Does NLB support WebSockets?**  
A: Yes, via TCP.

**Q: How to assign static IP to NLB?**  
A: Use **Elastic IPs** per AZ.

**Why We Select Public Subnets for NLB (Even When Instances Are Private)**

When setting up a **Network Load Balancer (NLB)**, AWS requires you to **select public subnets** for the NLB nodes, even if your backend instances are in **private subnets**. Here’s why:

**NLB Needs Public IPs to Accept Internet Traffic**

* **NLB operates at Layer 4 (TCP/UDP)** and **does not perform NAT** (unlike ALB).
* The **NLB nodes themselves must have public IPs** to accept traffic from the internet.
* The **backend instances (EC2)** can remain private (no public IPs).

**How Traffic Flows:**

Copy

Download

Internet User → NLB (Public Subnet) → Private EC2 Instances

* The **NLB nodes** act as the entry point.
* They **forward traffic** to private instances via **private IPs**.