[과제3] ALB 대신 NLB를 생성하여 ASG에 연결하는 테라폼 코드 작성 및 실습

4팀

목차

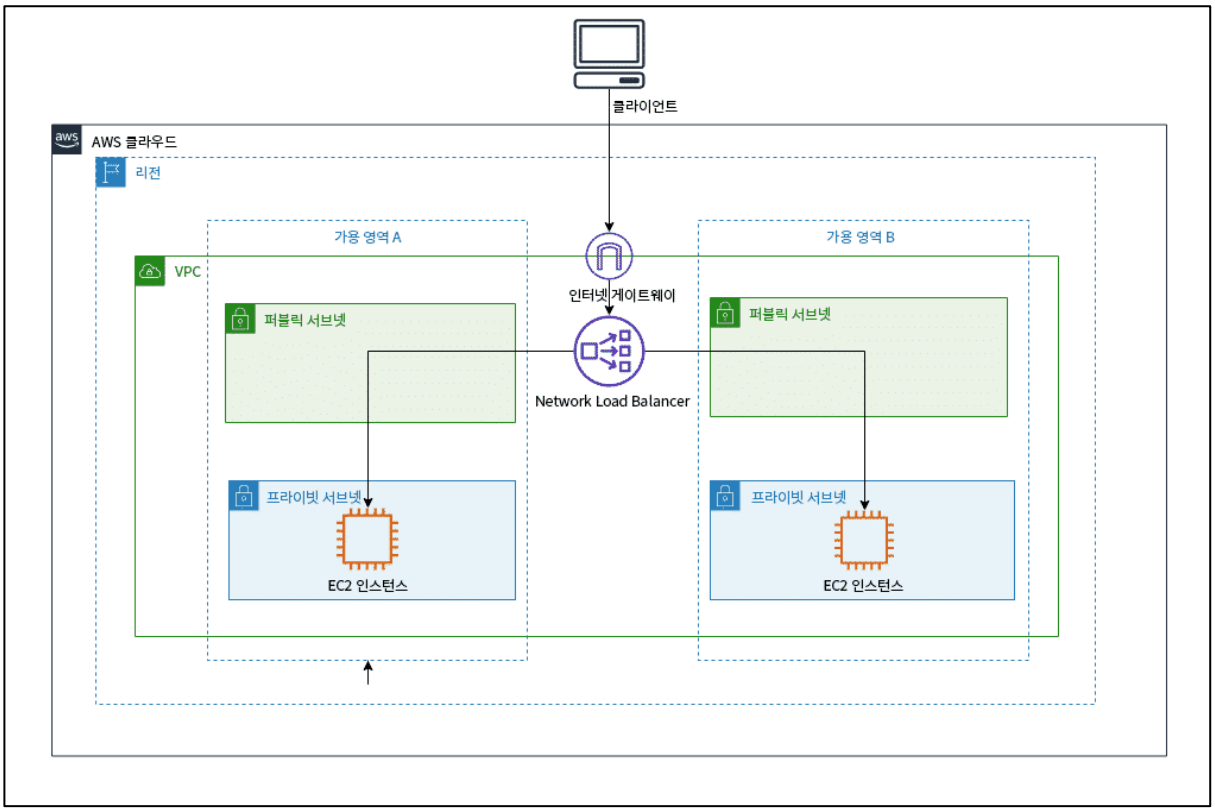
1. NLB

1) 개념 및 특징

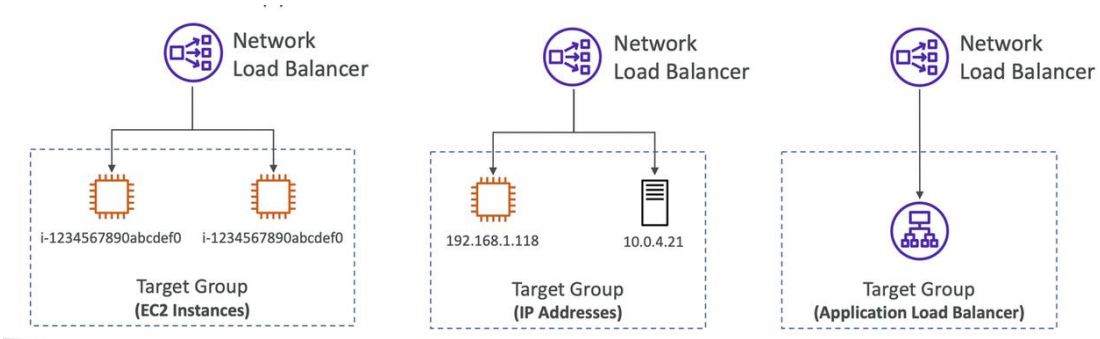
- Network Load Balancer

- **4계층**에서 동작하는 것으로, 이는 **TCP 및 UDP 로드 밸런싱**에 적합함

- 수신 트래픽을 **여러 대상 그룹으로 분산하여** 갑작스럽고 변동이 심한 **트래픽 패턴에 대응**하도록 최적화됨



- NLB의 대상 그룹으로 포함될 수 있는 것 : EC2 인스턴스, 사설 IP 주소, ALB



- 참고) NLB는 플로우 해싱 알고리즘을 사용하여 대상을 선택함. 이 알고리즘은 네트워크 트래픽을 여러 서버에 공정하고 효율적으로 분산시키는 방법임.

  ⇒ 플로우 해싱 알고리즘 작동 방식은 먼저 NLB가 각 TCP 연결 요청에 대해 소스 IP, 소스 포트, 목적지 IP, 목적지 포트, TCP 시퀀스 번호와 같은 특정 데이터를 수집함

  ⇒ 수집된 데이터를 해시 함수에 입력한 후, 이는 고정된 길이의 해시값을 생성함. 이 해시값은 대상 서버를 선택하는데 사용됨

  ⇒ 생성된 해시값을 사용해 연결이 어떤 서버에 할당될지 결정함. 이 과정은 일관성을 유지하며 같은 소스에서 오는 요청들이 동일한 서버로 라우팅 되도록 함

- 고객이 거의 아무런 조치를 하지 않아도 **높은 처리량**과 **매우 짧은 지연 시간을 유지**하면서 **초당 수천만 건의 요청을 처리**하도록 설계됨

- IP 주소 + 포트번호 보고 스위칭함

2) 이점

① 짧은 지연 시간

- 지연 시간이 중요한 애플리케이션에 매우 짧은 지연 시간을 제공

② 고정 IP 주소를 제공

- 각 가용 영역(서브넷)에 하나의 고정 IP를 자동으로 제공함

③ 장기적 TCP 연결 지원

- 장기간(~수년) 유지되는 연결을 처리할 수 있는 장기적 tcp 연결 지원함

④ DNS 장애 조치

- NLB는 여러 가용 영역(AZ)에 걸쳐 구성할 수 있으며, 각 가용 영역에는 로드 밸런서의 노드가 있음

  ⇒ 로드밸런서 노드 : 트래픽을 처리하거나 분산시키는 실제 장비 또는 소프트웨어 인스턴스

- Amazon Route 53를 사용하면, **하나의 가용 영역에 있는 NLB 노드에 문제가 발생**했

을 때, **자동으로 다른 가용 영역으로 트래픽 전환**

⑤ TLS 오프로드

- SSL/TLS 오프로딩(Offloading)은 서버 애플리케이션 외에서 SSL/TLS(이하 TLS) 처리를 대신하는 것
- 클라이언트와 서버 간에 안전하게 TLS로 암호화된 데이터를 전송하기 위해, 서버는 해당 데이터를 암호화하고 복호화 하는 작업을 해야함.

  ⇒ 하지만 TLS 오프로드 기능 사용 시 **로드 밸런서가 암/복호화 과정을 서버 대신 수행해 서버의 부하를 경감시킴**
- 보통 로드밸런서가 클라이언트의 요청을 받아 백엔드 서버로 전달할 때 클라이언트의 실제 IP가 로드밸런서의 IP 주소로 대체될 수 있음.

  ⇒ 하지만 TLS 오프로드를 지원하는 로드밸런서는 원래 클라이언트의 IP 주소를 유지할 수 있는 기능을 제공해, **백엔드 서버가 실제 클라이언트의 IP 주소를 알 수 있게 해 줌 (소스 IP 주소 유지)**

2. ALB

1) 개념

- Application Load Balancer
- **7계층**에서 동작하는 것으로, HTTP와 HTTPS, WebSocket을 활용함

  ⇒ 따라서 HTTP의 Header, 요청 Method 등을 이용해 **사용자의 요청을 적절한 대상 그룹으로 라우팅(부하분산)** 할 수 있으며 규칙에 우선순위를 두고 차례대로 적용할 수 있음
- 7계층 로드밸런서라고 해서 4계층(TCP, UDP)을 무시하는 것이 아님. 4계층의 규약(프로토콜)을 충분히 이행한 후 HTTP를 이용해 라우팅 실시함

2) 특징

- **HTTP를 활용한 라우팅(부하분산) :** ALB의 가장 큰 특징은 HTTP의 특성을 활용하는 것임.

  ⇒ **HTTP 헤더**를 라우팅 규칙에 활용함. 요청 헤더에 사용자의 상태 정보나 사용자가

사용하는 디바이스의 정보를 담음. 일반 헤더에 속하는 대표적인 헤더는 X-Forwarded-For로 사용자의 IP를 헤더에 담아 서버에게 전달

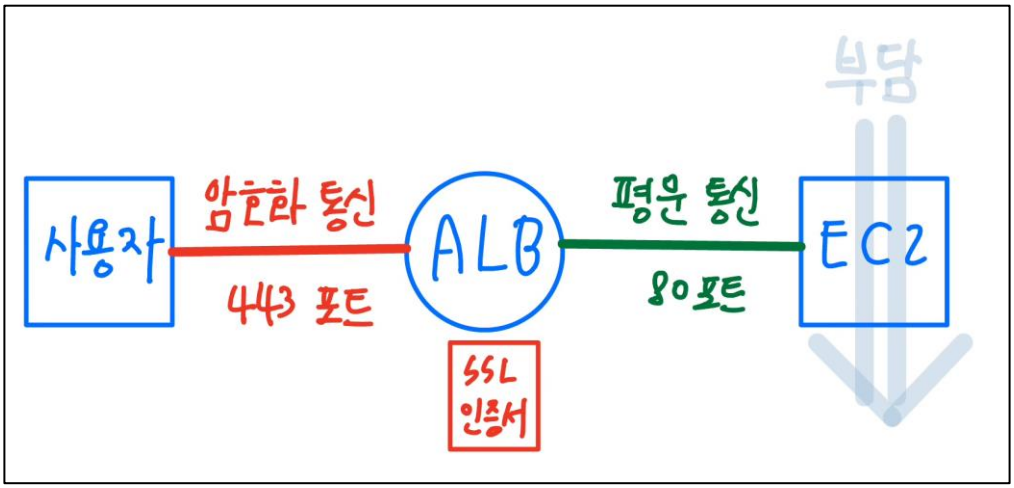
⇒ ALB는 **요청 메서드**(GET, HEAD, POST, PUT 등)를 기준으로 규칙을 생성해 각 규칙에 맞는 적절한 대상그룹으로 라우팅을 실시할 수 있음

- 로드밸런싱 부하 방식 : 기본은 Round Robin으로, 요청이 오면 EC2에 순서대로 할당되어 요청이 고르게 분산됨. 하지만 요청된 작업을 끝내지 못했는데 지속적으로 요청이 유입될 시 균형 깨질 수 있음

⇒ Round Robin의 아래 항목인 Least Outstanding Requests 활용함. 처리되지 않은 요청을 가장 적게 있는 ec2에게 할당하는 것임

- IP 주소 + 포트번호 + 패킷 내용을 보고 스위칭

- SSL 인증서 탑재 가능 : 사용자와 EC2 인스턴스가 암호화 통신을 해야 할 경우, ALB가 EC2 인스턴스의 부담을 줄이기 위해 EC2 대신 암호화 통신 실시함.



⇒ 따라서 위의 그림처럼 EC2와 ALB는 암호화 통신(HTTPS)을 하여 443 포트로 접근

⇒ EC2 인스턴스는 평문 통신만 하면 됨. 따라서 대상그룹 EC2의 포트는 80번임

- Sticky Session 지원 : 로드밸런서에 생성된 쿠키를 사용해 동일한 클라이언트의 요청을 동일한 대상으로 라우팅하는 sticky session을 지원함

- 프록시 서버로서의 역할 : ALB가 사용자와 EC2 중간에서 프록시 서버로서 양쪽과 통신함

- AWS WAF 연동 : ALB는 HTTP를 위한 로드밸런서이기 때문에 웹 관련 공격에서 자유로울 수 없음.

⇒ AWS의 웹 애플리케이션 방화벽 서비스인 AWS WAF를 연결해 ALB로 유입되는 웹 공격으로부터 EC2 인스턴스 보호할 수 있음

3. NLB, ALB 차이점

1) OSI 계층

- ALB : 애플리케이션 계층은 7계층의 HTTP/HTTPS를 확인해서 웹 애플리케이션 로드 밸런싱에 효과적

  - 특히 헤더를 확인해서 목적에 맞는 대상 그룹에게 라우팅할 수 있음

- NLB : 4계층을 전담하는 로드 밸런서로 7계층처럼 많은 내용을 볼 필요가 없음

  - 때문에 대용량의 데이터를 빠르게 처리하는 데 효과적. 아래의 실습 과정에서 UDP를 사용하는 이유도 이러한 네트워크 로드 밸런서의 기능을 최대화하기 위함

2) 통신 방법

- ALB : 내부에서 나가는 트래픽 또한 ALB를 사용할 수 있음

  - 이유 : 아키텍팅에 따라 다른데 내부에서 로깅, 애플리케이션의 성능 모니터링을 목적으로 ALB를 사용해서 라우팅할 수 있기 때문

- NLB : EC2를 로드밸런싱하는 경우 외부에서 내부로 이동하는 트래픽은 NLB를 거치지만 내부에서 외부로 이동하는 트래픽은 NLB를 거치지 않음

  - 이유 : 이는 NLB가 높은 성능으로 대용량 네트워크 트래픽을 처리하는 데 최적화되어 있기 때문

  - 내부에서 나가는 트래픽을 NLB를 거치지 않도록 해 트래픽을 분산하기 위함

3) IP

- ALB : IP가 유동적이다

- NLB : IP가 고정이다.

4. NLB, ALB 사용 용도

1) ALB

- 7계층 로드밸런서 장비로 HTTP/HTTPS 프로토콜 해더를 확인한 후 패킷을 전송한다.

- 때문에 웹 애플리케이션에 적합하지만, 7계층까지 확인하기 때문에 NLB보다 느리다는 단점이 존재함

- 라우팅 규칙이 필요한 웹 애플리케이션

2) NLB

- 4계층 로드밸런서 장비로 ALB보다 신속하게 대규모의 트래픽을 처리한다는 특징이 있음

- 고주파 거래 플랫폼 : 컴퓨터 알고리즘을 사용해서 빠르게 많은 주식을 거래하는 전략으로 대용량의 신속한 거래가 필요함

- 대규모 IoT 데이터 수집 : 연속적인 모니터링이 필요한 IoT의 경우 대규모 데이터를 보내서 이를 처리함

5. SNMP를 사용한 실습

1) SNMP란?

- 네트워크를 장치를 관리 및 모니터링하고 성능을 실시간 추적하는 모니터링 도구

- UDP 방식을 사용해서 통신

- SNMP Manager(162) : 장비의 정보를 수집하는 하드웨어/소프트웨어

- SNMP Agent(161) : 정보를 제공하는 주체인 네트워크 장비

2) SNMP를 사용하는 이유

- NLB는 4계층 로드밸런서로 ALB보다 빠르게 패킷을 처리할 수 있음

- 때문에 대용량의 빠른 처리가 필요한 아키텍처(대규모 IoT 데이터 수집, 실시간 멀티플레이어 게임 서버)에서 사용됨

- 그렇기에 UDP를 사용하는 SNMP로 NLB의 장점을 살리면서 서버의 상태를 파악할 수 있음.

6. SNMP 실습 가이드

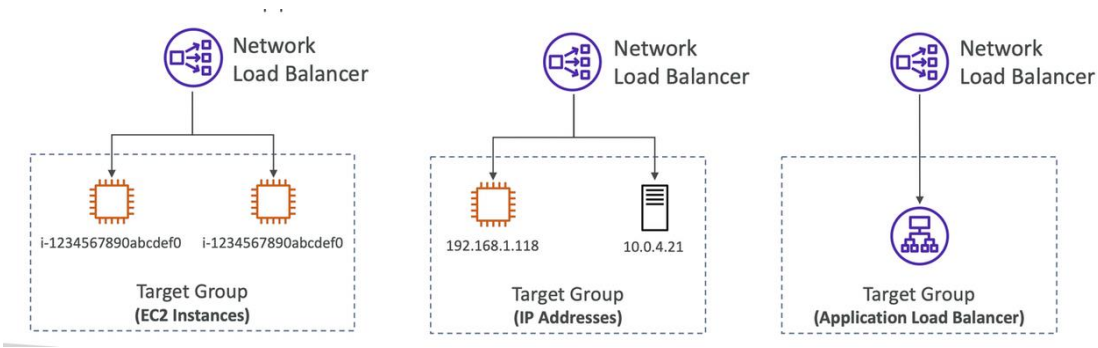
1) 개념 및 특징

- Network Load Balancer

- **4계층**에서 동작하는 것으로, 이는 **TCP 및 UDP 로드 밸런싱**에 적합함

- 수신 트래픽을 **여러 대상 그룹으로 분산하여** 갑작스럽고 변동이 심한 **트래픽 패턴에 대응**하도록 최적화됨

  - NLB의 대상 그룹으로 포함될 수 있는 것 : EC2 인스턴스, 사설 IP 주소, ALB

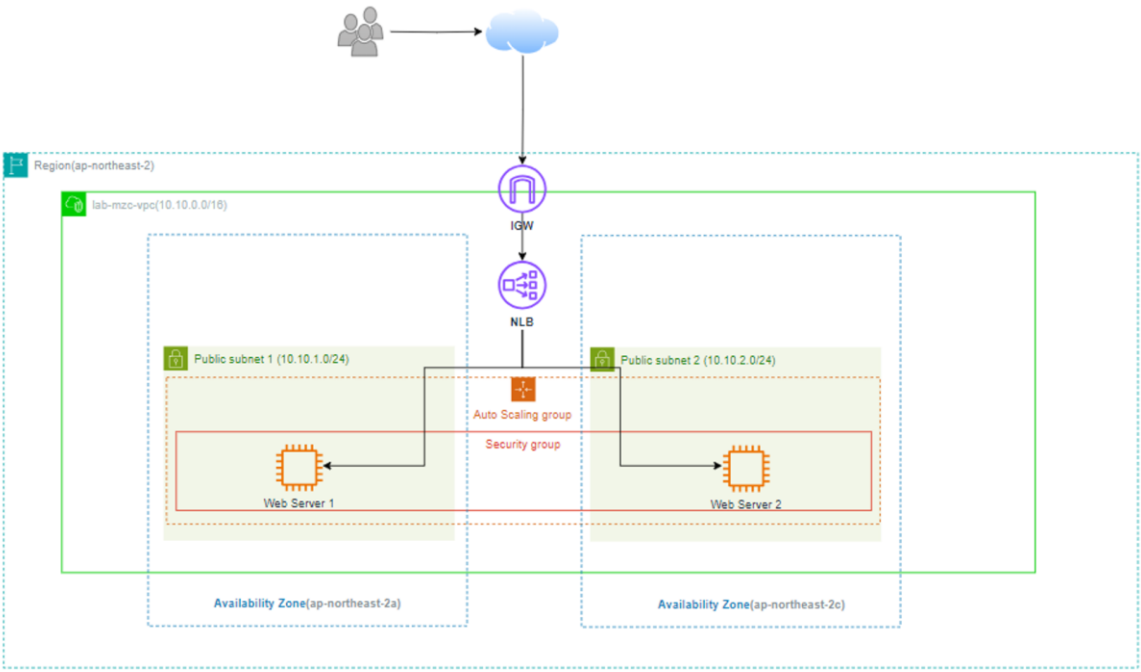


2) 시나리오

1. NLB 와 웹 서버 인스턴스 간의 통신을 허용하도록 보안 그룹을 구성합니다. SNMP 트래픽(UDP 포트 161)과 HTTP 트래픽(TCP 포트 80)을 인바운드 정책으로 허용 합니다.
2. EC2 Web Server 에 SNMP 를 설치합니다. EC2 인스턴스의 **user_data**를 통해 인스턴스가 시작될 때 자동으로 SNMP 서비스가 설치합니다.
3. 기존의 ALB 를 NLB 로 교체합니다. Terraform 를 사용하여 NLB 를 생성하고 TCP:80, UDP:161 리스너와 타겟 그룹을 설정합니다.

4. 외부에서 EC2 Web Server 에 설치된 SNMP 에이전트에 snmpget 명령어를 사용하여 SNMP 를 통한 로드밸런싱을 확인합니다.

3) 실습환경



4) 실습과정

4-1.1 vpc.tf → VPC 생성

```
provider "aws" {
  region  = "ap-northeast-2"
}

resource "aws_vpc" "myvpc" {
  cidr_block       = "10.10.0.0/16"
  enable_dns_support   = true
  enable_dns_hostnames = true

  tags = {
    Name = "mzcloud"
  }
}
```

4-1.2 vpc.tf → 서브넷 추가

```
resource "aws_subnet" "mysubnet1" {
  vpc_id     = aws_vpc.myvpc.id
  cidr_block = "10.10.1.0/24"

  availability_zone = "ap-northeast-2a"

  tags = {
    Name = "mzc-subnet1"
  }
}

resource "aws_subnet" "mysubnet2" {
  vpc_id     = aws_vpc.myvpc.id
  cidr_block = "10.10.2.0/24"

  availability_zone = "ap-northeast-2c"

  tags = {
    Name = "mzc-subnet2"
  }
}
```

4-1.3 vpc.tf → 인터넷 게이트웨이 추가

```
resource "aws_internet_gateway" "myigw" {
  vpc_id = aws_vpc.myvpc.id

  tags = {
    Name = "mzc-igw"
  }
}
```

4-1.4 vpc.tf → 라우팅 테이블 추가

```
resource "aws_route_table" "myrt" {
  vpc_id = aws_vpc.myvpc.id

  tags = {
    Name = "mzc-rt"
  }
}

resource "aws_route_table_association" "myrtassociation1" {
  subnet_id      = aws_subnet.mysubnet1.id
  route_table_id = aws_route_table.myrt.id
}

resource "aws_route_table_association" "myrtassociation2" {
  subnet_id      = aws_subnet.mysubnet2.id
  route_table_id = aws_route_table.myrt.id
}

resource "aws_route" "mydefaultroute" {
  route_table_id         = aws_route_table.myrt.id
  destination_cidr_block = "0.0.0.0/0"
  gateway_id             = aws_internet_gateway.myigw.id
}
```

4-2.1 sg.tf → 보안그룹 TCP:80 허용 인바운드 정책 생성

```
resource "aws_security_group" "mysg" {
  vpc_id      = aws_vpc.myvpc.id
  name        = "MZC SG"
  description = "MZC-101 SG"
}

resource "aws_security_group_rule" "mysginbound" {
  type              = "ingress"
  from_port         = 0
  to_port           = 80
  protocol          = "tcp"
  cidr_blocks       = ["0.0.0.0/0"]
  security_group_id = aws_security_group.mysg.id
}
```

4-2.2 sg.tf → 보안그룹 UDP:161 허용 인바운드 정책 추가

```
resource "aws_security_group_rule" "snmpsginbound" {
  type              = "ingress"
  from_port         = 161
  to_port           = 161
  protocol          = "udp"
  cidr_blocks       = ["0.0.0.0/0"]
  security_group_id = aws_security_group.mysg.id
}
```

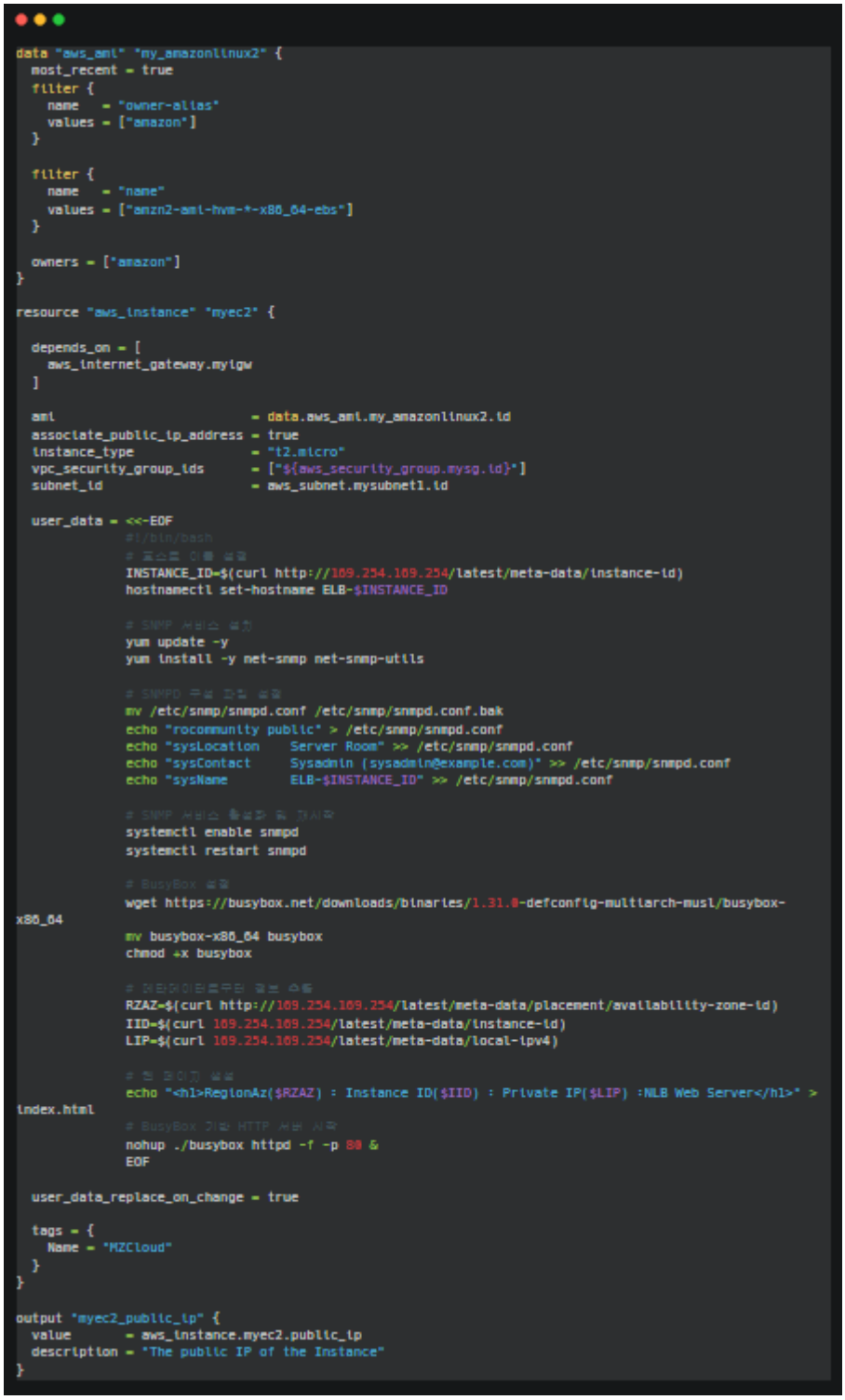
4-2.3 sg.tf → 모든트래픽 허용 아웃바운드 정책 추가

```
resource "aws_security_group_rule" "mysgoutbound" {
  type              = "egress"
  from_port         = 0
  to_port           = 0
  protocol          = "-1"
  cidr_blocks       = ["0.0.0.0/0"]
  security_group_id = aws_security_group.mysg.id
}
```

4-3.1 ec2.tf → 웹서버 생성

```hcl
data "aws_ami" "my_amazonlinux2" {
  most_recent = true
  filter {
    name   = "owner-alias"
    values = ["amazon"]
  }

  filter {
    name   = "name"
    values = ["amzn2-ami-hvm-*-x86_64-ebs"]
  }

  owners = ["amazon"]
}

resource "aws_instance" "myec2" {

  depends_on = [
    aws_internet_gateway.myigw
  ]

  ami                         = data.aws_ami.my_amazonlinux2.id
  associate_public_ip_address = true
  instance_type               = "t2.micro"
  vpc_security_group_ids      = ["${aws_security_group.mysg.id}"]
  subnet_id                   = aws_subnet.mysubnet1.id

  user_data = <<-EOF
              #!/bin/bash
              # BusyBox 설정
              wget https://busybox.net/downloads/binaries/1.31.0-defconfig-multiarch-musl/busybox-
x86_64
              mv busybox-x86_64 busybox
              chmod +x busybox

              # 메타데이터로부터 정보 추출
              RZAZ=$(curl http://169.254.169.254/latest/meta-data/placement/availability-zone-id)
              IID=$(curl 169.254.169.254/latest/meta-data/instance-id)
              LIP=$(curl 169.254.169.254/latest/meta-data/local-ipv4)

              # 웹 페이지 생성
              echo "<h1>RegionAz($RZAZ) : Instance ID($IID) : Private IP($LIP) :NLB Web Server</h1>" >
index.html
              # BusyBox 기반 HTTP 서버 시작
              nohup ./busybox httpd -f -p 80 &
              EOF

  user_data_replace_on_change = true

  tags = {
    Name = "MZCloud"
  }
}

output "myec2_public_ip" {
  value       = aws_instance.myec2.public_ip
  description = "The public IP of the Instance"
}
```

4-3.2 ec2.tf → SNMP 서비스 추가



```
data "aws_ami" "my_amazonlinux2" {
  most_recent = true
  filter {
    name   = "owner-alias"
    values = ["amazon"]
  }

  filter {
    name   = "name"
    values = ["amzn2-ami-hvm-*-x86_64-ebs"]
  }

  owners = ["amazon"]
}

resource "aws_instance" "myec2" {

  depends_on = [
    aws_internet_gateway.myigw
  ]

  ami                         = data.aws_ami.my_amazonlinux2.id
  associate_public_ip_address = true
  instance_type               = "t2.micro"
  vpc_security_group_ids      = ["${aws_security_group.mysg.id}"]
  subnet_id                   = aws_subnet.mysubnet1.id

  user_data = <<-EOF
              #!/bin/bash
              # 호스트 이름 설정
              INSTANCE_ID=$(curl http://169.254.169.254/latest/meta-data/instance-id)
              hostnamectl set-hostname ELB-$INSTANCE_ID

              # SNMP 서비스 설치
              yum update -y
              yum install -y net-snmp net-snmp-utils

              # SNMPD 구성 파일 설정
              mv /etc/snmp/snmpd.conf /etc/snmp/snmpd.conf.bak
              echo "rocommunity public" > /etc/snmp/snmpd.conf
              echo "sysLocation    Server Room" >> /etc/snmp/snmpd.conf
              echo "sysContact     Sysadmin (sysadmin@example.com)" >> /etc/snmp/snmpd.conf
              echo "sysName        ELB-$INSTANCE_ID" >> /etc/snmp/snmpd.conf

              # SNMP 서비스 활성화 및 시작
              systemctl enable snmpd
              systemctl restart snmpd

              # BusyBox 설치
              wget https://busybox.net/downloads/binaries/1.31.0-defconfig-multiarch-musl/busybox-
x86_64
              mv busybox-x86_64 busybox
              chmod +x busybox

              # 메타데이터를 가져와 변수에 저장
              RZAZ=$(curl http://169.254.169.254/latest/meta-data/placement/availability-zone-id)
              IID=$(curl 169.254.169.254/latest/meta-data/instance-id)
              LIP=$(curl 169.254.169.254/latest/meta-data/local-ipv4)

              # 웹 페이지 생성
              echo "<h1>RegionAz($RZAZ) : Instance ID($IID) : Private IP($LIP) :NLB Web Server</h1>" >
index.html
              # BusyBox 기반 HTTP 서버 시작
              nohup ./busybox httpd -f -p 80 &
              EOF

  user_data_replace_on_change = true

  tags = {
    Name = "MZCloud"
  }
}

output "myec2_public_ip" {
  value       = aws_instance.myec2.public_ip
  description = "The public IP of the Instance"
}
```

1. EC2 인스턴스의 호스트 이름 설정:
    - **INSTANCE_ID**를 사용하여 인스턴스의 메타데이터에서 인스턴스 ID 를 가져옵니다.
    - hostnamectl set-hostname 명령어를 사용하여 호스트 이름을 설정합니다.
2. SNMP 서비스 설치 및 구성:

- **yum update -y**를 사용하여 패키지를 업데이트합니다.
- **yum install -y net-snmp net-snmp-utils**를 사용하여 SNMP 관련 패키지를 설치합니다.
- /etc/snmp/snmpd.conf 파일을 업데이트하여 SNMP 구성을 설정합니다. 여기서는 커뮤니티 문자열을 public 으로 설정하고, 시스템 위치 및 연락처를 설정합니다.
- **systemctl enable snmpd**와 **systemctl restart snmpd**를 사용하여 SNMP 서비스를 활성화하고 재시작합니다.

3. BusyBox 설정:
   - BusyBox 를 다운로드하고 실행 가능한 파일로 변경합니다.
4. 웹 페이지 생성 및 HTTP 서버 시작:
   - 인스턴스의 메타데이터에서 지역 IP 주소 및 기타 정보를 가져와서 HTML 파일을 생성합니다.
   - 생성된 HTML 파일을 사용하여 간단한 웹 페이지를 생성합니다.
   - BusyBox 를 사용하여 HTTP 서버를 실행하고 포트 80 에서 수신 대기합니다.

❓ 구성 파일 설정

1. `echo "rocommunity public" > /etc/snmp/snmpd.conf` : 이 명령어는 SNMP 에이전트가 읽기 전용 커뮤니티 문자열을 설정하는 데 사용됩니다. 여기서 "public"은 일반적으로 사용되는 커뮤니티 문자열로, SNMP 요청을 보내는 장치가 이 문자열을 사용하여 SNMP 에이전트에 읽기 액세스할 수 있습니다. 이 설정은 SNMP 요청을 수신하는 데 사용되며, SNMP 에이전트는 이 커뮤니티 문자열을 사용하여 요청을 인증합니다.
2. `echo "sysLocation Server Room" >> /etc/snmp/snmpd.conf` : 이 명령어는 SNMP 에이전트의 위치를 설정하는 데 사용됩니다. 이 위치 정보는 SNMP 관리자가 네트워크 장비의 물리적인 위치를 파악할 수 있도록 돕습니다. 예를 들어, "Server Room"은 이 SNMP 에이전트가 서버 룸에 위치해 있음을 나타냅니다.
3. `echo "sysContact Sysadmin (sysadmin@example.com)" >> /etc/snmp/snmpd.conf` : 이 명령어는 SNMP 에이전트의 관리자 연락처를 설정하는 데 사용됩니다. 이 연락처 정보는 SNMP 관리자가 문제가 발생했을 때 적절한 담당자에게 연락할 수 있도록 돕습니다. 예를 들어, "sysadmin@example.com"은 시스템 관리자의 이메일 주소를 나타냅니다.
4. `echo "sysName ELB-$INSTANCE_ID" >> /etc/snmp/snmpd.conf` : 이 명령어는 SNMP 에이전트의 이름을 설정하는 데 사용됩니다. 이 이름은 SNMP 관리자가 각 에이전트를 식별하는 데 사용됩니다. 여기서 "$INSTANCE_ID"는 인스턴스의 고유 식별자를 나타내며, 보통 인스턴스의 고유한 이름을 설정하는 데 사용됩니다.

4-4.1 nlb.tf → NLB 및 타켓 그룹 생성

```hcl
resource "aws_lb" "mynlb" {
  name               = "mzc-nlb"
  load_balancer_type = "network"
  subnets            = [aws_subnet.mysubnet1.id, aws_subnet.mysubnet2.id]
  security_groups    = [aws_security_group.mysg.id]

  tags = {
    Name = "mzc-nlb"
  }
}

resource "aws_lb_target_group" "my_nlb_tg_udp" {
  name     = "mzc-nlb-tg-udp"
  port     = 161
  protocol = "UDP"
  vpc_id   = aws_vpc.myvpc.id

  health_check {
    protocol            = "TCP"
    port                = "80"
    healthy_threshold   = 3
    unhealthy_threshold = 3
    interval            = 30
    timeout             = 10
  }
}

resource "aws_lb_target_group" "my_nlb_tg_tcp" {
  name     = "mzc-nlb-tg-tcp"
  port     = 80
  protocol = "TCP"
  vpc_id   = aws_vpc.myvpc.id

  health_check {
    protocol            = "TCP"
    port                = "80"
    healthy_threshold   = 3
    unhealthy_threshold = 3
    interval            = 30
    timeout             = 10
  }
}
```

1. **aws_lb**: 이 리소스는 AWS 에서 Network Load Balancer 를 생성합니다.
    - name: NLB 의 이름을 설정합니다.
    - load_balancer_type: 로드 밸런서 유형을 지정합니다. 여기서는 "network"로 설정되어 있습니다.
    - subnets: 로드 밸런서를 배치할 서브넷의 ID 목록을 설정합니다.
    - security_groups: 로드 밸런서에 연결할 보안 그룹의 ID 를 설정합니다.
2. **aws_lb_target_group**: 이 리소스는 NLB 에 대한 대상 그룹을 생성합니다. 각 대상 그룹은 NLB 로 전송되는 트래픽을 관리합니다.

- name: 대상 그룹의 이름을 설정합니다.
- port: 대상 그룹의 포트 번호를 설정합니다.
- protocol: 대상 그룹이 사용하는 프로토콜을 설정합니다.
- vpc_id: 대상 그룹이 속한 VPC 의 ID 를 설정합니다.
- health_check: 대상 그룹의 상태를 확인하기 위한 건강 검사 구성을 설정합니다. 여기서는 TCP 프로토콜을 사용하여 80 번 포트를 확인합니다. 이는 대상 그룹 내의 대상 인스턴스가 정상적으로 작동하는지 확인하는 데 사용됩니다.
  - **healthy_threshold** 는 건강 검사에서 인스턴스를 '정상'으로 간주하기 위해 필요한 연속적인 성공 응답 수입니다.
  - **unhealthy_threshold** 는 인스턴스를 '비정상'으로 표시하기 위해 필요한 연속적인 실패 응답 수입니다.
  - **nterval** 은 각 건강 검사 간의 시간 간격을 설정합니다.
  - **timeout** 은 건강 검사 요청에 대한 응답을 기다리는 시간 제한을 설정

> ? **헬스 체크 시 TCP:80을 사용하는 이유**
>
> 네트워크 로드 밸런서(NLB)는 OSI 모델의 4계층인 전송 계층에서 동작하므로, HTTP 요청과 같은 프로토콜 레벨의 상태를 확인할 수 없습니다. **NLB는 패킷 레벨에서만 동작**하며, **TCP 및 UDP 패킷의 라우팅 및 분산을 담당**합니다.
>
> 따라서 NLB의 헬스 체크에서 포트 80을 사용하는 것은 실제 HTTP 서버에 요청을 보내는 것이 아니라, **해당 포트에 대한 TCP 연결만을 확인**하는 것입니다. 이를 통해 서버의 네트워크 레벨에서의 가용성을 확인할 수 있습니다.

4-4.2 nlb.tf → NLB 리스너 추가

```
resource "aws_lb_listener" "mynlb_listener_udp" {
  load_balancer_arn = aws_lb.mynlb.arn
  port              = 161
  protocol          = "UDP"

  default_action {
    type             = "forward"
    target_group_arn = aws_lb_target_group.my_nlb_tg_udp.arn
  }
}

resource "aws_lb_listener" "mynlb_listeneDr_tcp" {
  load_balancer_arn = aws_lb.mynlb.arn
  port              = 80
  protocol          = "TCP"

  default_action {
    type             = "forward"
    target_group_arn = aws_lb_target_group.my_nlb_tg_tcp.arn
  }
}

output "mynlb_dns" {
  value       = aws_lb.mynlb.dns_name
  description = "The DNS Address of the NLB"
}
```

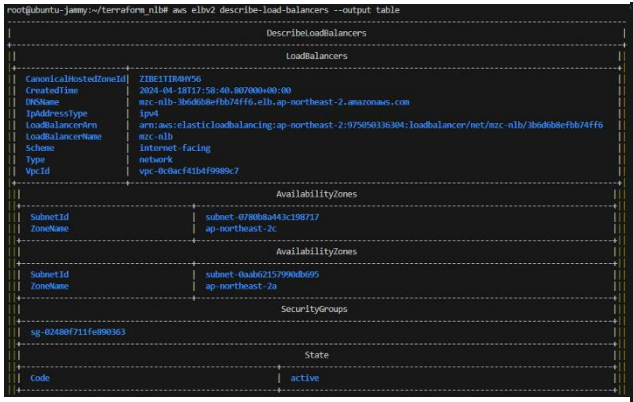
1. **aws_lb_listener.mynlb_listener_udp**: 이 리소스는 NLB 에 UDP 프로토콜을 사용하는 리스너를 추가합니다.
    - load_balancer_arn: 리스너가 연결될 NLB 의 Amazon 리소스 이름(ARN)을 지정합니다.
    - port: 리스너가 수신 대기할 포트를 지정합니다. 여기서는 161 번 포트를 사용합니다.
    - protocol: 리스너가 사용할 프로토콜을 지정합니다. UDP 프로토콜을 사용합니다.
    - default_action: 리스너에 대한 기본 작업을 정의합니다. 여기서는 "forward"로 설정되어 있어서, 수신된 요청을 대상 그룹으로 전달합니다. 이 때, **target_group_arn**은 UDP 프로토콜을 사용하는 대상 그룹의 ARN 을 지정합니다.
2. **aws_lb_listener.mynlb_listener_tcp**: 이 리소스는 NLB 에 TCP 프로토콜을 사용하는 리스너를 추가합니다.
    - load_balancer_arn: 리스너가 연결될 NLB 의 Amazon 리소스 이름(ARN)을 지정합니다.

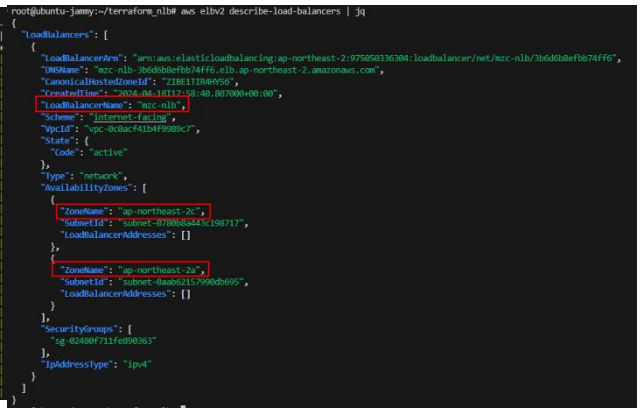
- port: 리스너가 수신 대기할 포트를 지정합니다. 여기서는 80 번 포트를 사용합니다.
- protocol: 리스너가 사용할 프로토콜을 지정합니다. TCP 프로토콜을 사용합니다.
- default_action: 리스너에 대한 기본 작업을 정의합니다. 여기서도 "forward"로 설정되어 있어서, 수신된 요청을 대상 그룹으로 전달합니다. 이 때, **target_group_arn**은 TCP 프로토콜을 사용하는 대상 그룹의 ARN 을 지정합니다.

3. **output "mynlb_dns"**: 이 출력은 NLB 의 DNS 주소를 반환합니다. 이 주소를 통해 NLB 로 전달되는 요청을 처리할 수 있습니다.

4-4.3 NLB 정보 확인

```
# ALB 정보 확인
aws elbv2 describe-load-balancers --output table
aws elbv2 describe-load-balancers | jq
```





4-5.1 asg.tf → 시작템플릿 생성

위의 EC2 의 user data 와 마찬가지로 수정

```
resource "aws_launch_configuration" "mylauchconfig" {
  name_prefix     = "mzc-cloud-"
  image_id        = data.aws_ami.my_amazonlinux2.id
  instance_type   = "t2.micro"
  security_groups = [aws_security_group.mysg.id]
  associate_public_ip_address = true

  user_data = <<-EOF
              #!/bin/bash
              # 호스트 이름 설정
              INSTANCE_ID=$(curl http://169.254.169.254/latest/meta-data/instance-id)
              hostnamectl set-hostname ELB-$INSTANCE_ID

              # SNMP 서비스 설치
              yum update -y
              yum install -y net-snmp net-snmp-utils

              # SNMPD 구성 파일 설정
              mv /etc/snmp/snmpd.conf /etc/snmp/snmpd.conf.bak
              echo "rocommunity public" > /etc/snmp/snmpd.conf
              echo "sysLocation    Server Room" >> /etc/snmp/snmpd.conf
              echo "sysContact     Sysadmin (sysadmin@example.com)" >> /etc/snmp/snmpd.conf
              echo "sysName        ELB-$INSTANCE_ID" >> /etc/snmp/snmpd.conf

              # SNMP 서비스 활성화 및 재시작
              systemctl enable snmpd
              systemctl restart snmpd

              # BusyBox 설정
              wget https://busybox.net/downloads/binaries/1.31.0-defconfig-multiarch-musl/busybox-
x86_64
              mv busybox-x86_64 busybox
              chmod +x busybox

              # 메타데이터로부터 정보 추출
              RZAZ=$(curl http://169.254.169.254/latest/meta-data/placement/availability-zone-id)
              IID=$(curl 169.254.169.254/latest/meta-data/instance-id)
              LIP=$(curl 169.254.169.254/latest/meta-data/local-ipv4)

              # 웹 페이지 생성
              echo "<h1>RegionAz($RZAZ) : Instance ID($IID) : Private IP($LIP) :NLB Web Server</h1>" >
index.html
              # BusyBox 기반 HTTP 서버 시작
              nohup ./busybox httpd -f -p 80 &
              EOF

  # Required when using a launch configuration with an auto scaling group.
  lifecycle {
    create_before_destroy = true
  }
}
```

4-5.2 asg.tf → 오토스케일링 그룹 추가

```
resource "aws_autoscaling_group" "myasg" {
  name                 = "myasg"
  launch_configuration = aws_launch_configuration.mylauchconfig.name
  vpc_zone_identifier  = [aws_subnet.mysubnet1.id, aws_subnet.mysubnet2.id]
  min_size = 2
  max_size = 4
  health_check_type = "ELB"
  target_group_arns = [aws_lb_target_group.my_nlb_tg_udp.arn, aws_lb_target_group.my_nlb_tg_tcp.arn]

  tag {
    key                 = "Name"
    value               = "test-asg"
    propagate_at_launch = true
  }
}
```

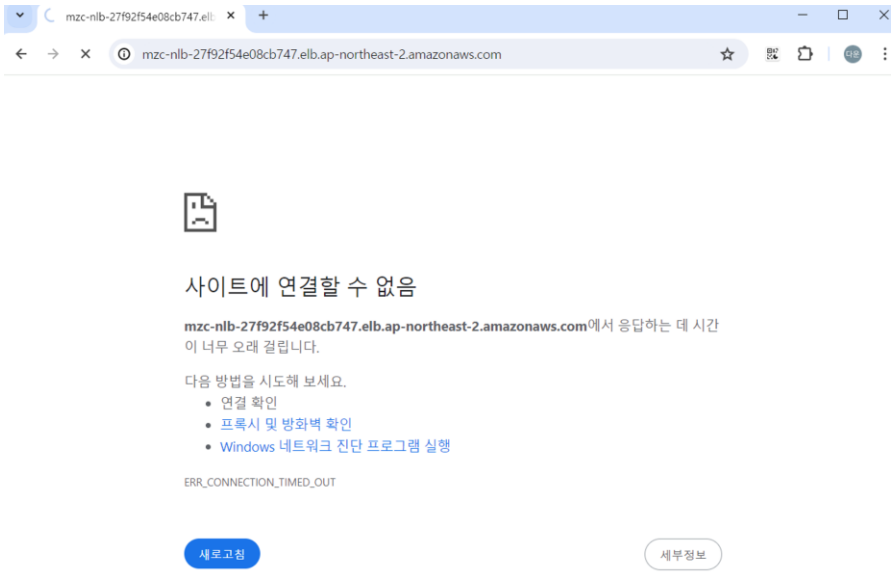
(수정) NLB 리스너 설정

```
resource "aws_lb_listener" "mynlb_listener" {
  load_balancer_arn = aws_lb.mynlb.arn
  port              = 161
  protocol          = "UDP"

  default_action {
    type             = "forward"
    target_group_arn = aws_lb_target_group.mynlbtg.arn
  }
}
```

SNMP 기반으로 로드밸런싱 확인

```
root@ubuntu-jammy:~# snmpget -v2c -c public mzc-nlb-27f92f54e08cb747.elb.ap-northeast-2.amazonaws.com 1.3.6.1.2.1.1.5.0
iso.3.6.1.2.1.1.5.0 = STRING: "ELB-i-08b5fdf4118c92a85"
root@ubuntu-jammy:~# snmpget -v2c -c public mzc-nlb-27f92f54e08cb747.elb.ap-northeast-2.amazonaws.com 1.3.6.1.2.1.1.5.0
iso.3.6.1.2.1.1.5.0 = STRING: "ELB-i-041bbe0520b8ce19c"
```

NLB 리스너 설정에 SNMP 만 설정되어 있어 웹페이지는 접근이 안됩니다.



(수정) TCP 기반 리스너 추가

```
resource "aws_lb_listener" "mynlb_listener_tcp" {
  load_balancer_arn = aws_lb.mynlb.arn
  port              = 80
  protocol          = "TCP"

  default_action {
    type             = "forward"
    target_group_arn = aws_lb_target_group.mynlbtg.arn
  }
}
```

오류 메시지: 생성하려는 TCP 리스너와 지정한 타겟 그룹 간에 프로토콜이 호환되지 않음

해결방법: TCP 리스너와 호환되는 새로운 타겟 그룹을 생성

리스너의 타겟그룹 수정

```
resource "aws_lb_listener" "mynlb_listener_udp" {
  load_balancer_arn = aws_lb.mynlb.arn
  port              = 161
  protocol          = "UDP"

  default_action {
    type             = "forward"
    target_group_arn = aws_lb_target_group.my_nlb_tg_udp.arn
  }
}

resource "aws_lb_listener" "mynlb_listeneDr_tcp" {
  load_balancer_arn = aws_lb.mynlb.arn
  port              = 80
  protocol          = "TCP"

  default_action {
    type             = "forward"
    target_group_arn = aws_lb_target_group.my_nlb_tg_tcp.arn
  }
}
```

(수정) asg.tf 수정

target_group_arns 배열에 UDP 와 TCP 타겟 그룹의 ARN 이 모두 포함되어 있어, ASG 가 이 두 타겟 그룹에 인스턴스를 등록합니다.

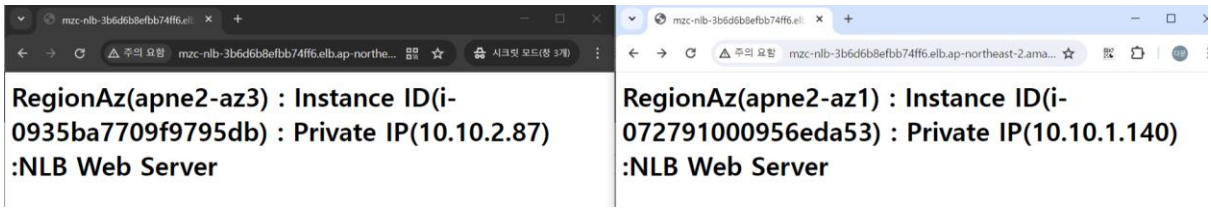
```
resource "aws_autoscaling_group" "myasg" {
  name                 = "myasg"
  launch_configuration = aws_launch_configuration.mylauchconfig.name
  vpc_zone_identifier  = [aws_subnet.mysubnet1.id, aws_subnet.mysubnet2.id]
  min_size = 2
  max_size = 4
  health_check_type = "ELB"
  target_group_arns = [aws_lb_target_group.my_nlb_tg_udp.arn, aws_lb_target_group.my_nlb_tg_tcp.arn]

  tag {
    key                 = "Name"
    value               = "test-asg"
    propagate_at_launch = true
  }
}
```

NLB dns 네임

mzc-nlb-3b6d6b8efbb74ff6.elb.ap-northeast-2.amazonaws.com

```
# NLB 모니터링
NLBDNS=각각DNS주소
while true; do curl --connect-timeout 1  http://$NLBDNS/ ; echo; echo "------------------------------"; date; sleep 1; done
```

```
root@ubuntu-jammy:~/terraform_nlb# NLBDNS=mzc-nlb-3b6d6b8efbb74ff6.elb.ap-northeast-2.amazonaws.com
root@ubuntu-jammy:~/terraform_nlb# while true; do curl --connect-timeout 1  http://$NLBDNS/ ; echo; echo "------------------------------"; date; sleep 1; done
<h1>RegionAz(apne2-az1) : Instance ID(i-072791000956eda53) : Private IP(10.10.1.140) :NLB Web Server</h1>

------------------------------
Thu Apr 18 18:44:06 UTC 2024
<h1>RegionAz(apne2-az1) : Instance ID(i-072791000956eda53) : Private IP(10.10.1.140) :NLB Web Server</h1>

------------------------------
Thu Apr 18 18:44:07 UTC 2024
<h1>RegionAz(apne2-az3) : Instance ID(i-0935ba7709f9795db) : Private IP(10.10.2.87) :NLB Web Server</h1>

------------------------------
Thu Apr 18 18:44:08 UTC 2024
<h1>RegionAz(apne2-az3) : Instance ID(i-0935ba7709f9795db) : Private IP(10.10.2.87) :NLB Web Server</h1>

------------------------------
Thu Apr 18 18:44:09 UTC 2024
```




```
root@ubuntu-jammy:~# snmpget -v2c -c public mzc-nlb-3b6d6b8efbb74ff6.elb.ap-northeast-2.amazonaws.com 1.3.6.1.2.1.1.5.0
iso.3.6.1.2.1.1.5.0 = STRING: "ELB-i-0935ba7709f9795db"
root@ubuntu-jammy:~# snmpget -v2c -c public mzc-nlb-3b6d6b8efbb74ff6.elb.ap-northeast-2.amazonaws.com 1.3.6.1.2.1.1.5.0
iso.3.6.1.2.1.1.5.0 = STRING: "ELB-i-072791000956eda53"
```

네트워크 인터페이스의 PrivateIpAddress 및 연결된 PublicIp(있을 경우)를 검색합니다.

```
aws ec2 describe-network-interfaces \
    --filters Name=interface-type,Values=network_load_balancer \
    --query 'NetworkInterfaces[*].[PrivateIpAddress, Association.PublicIp]' \
    --output text
```

모든 'network_load_balancer' 타입의 네트워크 인터페이스를 필터링하고 각 인터페이스의 **PrivateIpAddress**와 **PublicIp**를 반환합니다.

```
root@ubuntu-jammy:~/terraform_nlb# aws ec2 describe-network-interfaces \
    --filters Name=interface-type,Values=network_load_balancer \
    --query 'NetworkInterfaces[*].[PrivateIpAddress, Association.PublicIp]' \
    --output text
10.10.1.235     43.203.94.167
10.10.2.128     3.37.64.251
```

네트워크 응답 시간 및 대역폭: **iperf**를 사용하여 NLB 를 통한 UDP 전송의 대역폭과 응답 시간을 측정할 수 있습니다. 이를 통해 네트워크의 성능을 평가하고 최적화할 수 있습니다.

이 명령을 실행하면 NLB 를 통해 100 바이트의 패킷을 전송하고, 이에 대한 응답 시간과 대역폭을 측정할 수 있습니다.

```
iperf -u -c mzc-nlb-0c7a2c58ad6ca23a.elb.ap-northeast-2.amazonaws.com -b 1M -l 100
```

```
root@ubuntu-jammy:~/test# iperf -u -c mzc-nlb-0c7a2c58ad6ca23a.elb.ap-northeast-2.amazonaws.com -b 1M -l 100
------------------------------------------------------------
Client connecting to mzc-nlb-0c7a2c58ad6ca23a.elb.ap-northeast-2.amazonaws.com, UDP port 5001
Sending 100 byte datagrams, IPG target: 762.94 us (kalman adjust)
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  1] local 10.0.2.15 port 39989 connected with 13.124.42.46 port 5001
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-10.0002 sec  1.25 MBytes  1.05 Mbits/sec
[  1] Sent 13110 datagrams
[  3] WARNING: did not receive ack of last datagram after 10 tries.
```

1. Client connecting to mzc-nlb-0c7a2c58ad6ca23a.elb.ap-northeast-2.amazonaws.com, UDP port 5001: 클라이언트가 NLB 의 주소로 UDP 포트 5001 을 통해 연결을 시도하고 있음을 나타냅니다.
2. Sending 100 byte datagrams, IPG target: 762.94 us (kalman adjust): 100 바이트 크기의 데이터그램을 보내고 있으며, 패킷 간격의 대상이 762.94 마이크로초임을 나타냅니다.

3. UDP buffer size: 208 KByte (default): UDP 버퍼의 크기는 208 킬로바이트로 설정되어 있습니다. 이는 기본값으로 설정되어 있습니다.
4. [1] local 10.0.2.15 port 39989 connected with 13.124.42.46 port 5001: 클라이언트의 로컬 주소와 NLB 의 주소 간에 연결이 성공했음을 나타냅니다. 클라이언트의 로컬 포트는 39989 이고, NLB 의 포트는 5001 입니다.
5. [1] 0.0000-10.0002 sec 1.25 MBytes 1.05 Mbits/sec: 0 부터 10 초까지의 시간 동안 전송된 데이터양과 대역폭을 보여줍니다. 여기서는 1.25 MBytes 의 데이터가 전송되었고, 대역폭은 1.05 Mbits/sec 입니다.
6. [1] Sent 13110 datagrams: 전송된 데이터그램의 총 개수입니다.
7. [3] WARNING: did not receive ack of last datagram after 10 tries.: 마지막 데이터그램에 대한 확인 응답(acknowledgment)을 10 번 시도했지만 응답을 받지 못했다는 경고 메시지입니다.