LoxiLB Enterprise User Guide

(LoxiLB Enterprise Version 1.0)

NETLOX

LoxiLB Enterprise User Guide 구성

Chapter 01. LoxiLB Enterprise 소개

LoxiLB Version 1.0 기반의 패키지를 구성하는 엔터프리이즈용 'loxilb' 와 이의 CLI 명령어도구 'loxicmd'를 소개합니다. 선택적으로 사용가능한 UI를 포함합니다.

Chapter 02. 기본 설치

베어메탈이나 가상머신 등을 사용하는 LoxiLB Enterprise를 설치하는 방법을 알아봅니다.

Chapter 03. CLI 명령어 (loxicmd)

엔터프라이즈용 'loxilb'의 CLI(Command-Line Interface) 명령어 도구 'loxicmd'가 제공하는 기능을 설명합니다. CLI 명령어를 통해 로드밸런싱은 물론 보안 정책 강화와 모니터링 기능 등의 설정과 확인방법을 알아봅니다.

Chapter 04. HA(High-Availability)

LoxiLB Enterprise를 HA 모드로 배포하는 방법을 설명합니다.

Chapter 05. 문제 해결

네트워크 구성과 운영 시 문제 해결을 위한 유현별 조치 방법을 설명합니다

부록 01. 시험 방법

loxicmd 기능을 시험하는 방법을 설명합니다.

부록 02. UI

LoxiLB Enterprise 용으로 제공하는 그래픽 사용자 인터페이스 UI로 LB정책 관련 생성과 실행을 모니터링 합니다.

부록 03. Error 메시지

LoxiLB Enterprise에서 발생하는 유형별 메시지를 확인합니다.

목차

목차

CHAPTER

01

LoxiLB Enterprise 소개

개요

LoxiLB Enterprise는 eBPF를 엔진을 사용하는 로드밸런싱(LB) 솔루션입니다. 클라우드 환경의 고가용성(HA)과 관리를 위한 CLI와 함께 그래픽 기반의 사용자 인터페이스(UI: User Interface)를 제공합니다.

1. LoxiLB Enterprise 소개

LoxiLB Enterprise는 서비스 데이터의 초저지연 포워딩과 프로그램밍이 가능한 API를 사용하여 유연하게 로드밸런싱 서비스를 구현합니다.

LoxiLB Enterprise 솔루션은 클라우드 환경에서 가상머신(VM)이나 베어베탈로 컨테이너 이미지로 배포하여 구성, 프로비저닝, 확장, 업그레이드, 마이그레이션, 라우팅, 모니터링 및 리소스 관리와 연계하여 로드밸런싱 서비스를 제공합니다.

그리고, 프라이비트 클라우드(Private Cloud)는 물론 모든 국내외 CSP(Could Service Provider) 사업자 환경에서도 동일하게 실행 가능 할 뿐만 아니라 온프레미스 및 에지 클라우드 환경을 위해서 확장 서비스를 제공합니다.

클라우드 환경이 빠르게 확대하여 인프라 환경을 위한 엣지 서비스에 로드밸런싱 지원이 필요하게 되면서 LoxiLB Enterprise는 엔터프라이즈의 네트워크 표준화를 지원하는 오픈솔루션이 될 수 있습니다.

그림 1. LoxiLB Enterprise 아키텍처

트랜스포트 계층의 TCP/UDP 프로토콜을 모두 사용하는 서비스들은 클라우드화 하고 있습니다. LoxiLB Enterprise 솔루션은 트랜스포트 계층의 주요 프로토콜인 TCP/UDP를 지원하는 로드밸런싱을 제공합니다.

Linux 커뮤니티에서는 OS 커널에 새로운 기능을 도입할 수 있는 기술인 eBPF를 제공하고 있습니다. eBPF 기반으로 안전하게 독립적인 기능을 제공하는 샌드박스 프로그램으로 OS 커널에 새로운 기능을 도입할 수 있는 유연성과 OS가 제공하는 관측 능력(Observability)의 활용은 LoxiLB 솔루션의 설계 사상을 완벽하게 지원하고 있습니다.

eBPF 기반의 LoxiLB Enterprise는 로드밸런싱의 처리속도를 개선할 뿐만 아니라 컨테이너 기반으로 CPU 코어를 할당하지 않으므로 지속가능성(Sustainability)을 구현하는 에너지 효율적인 엣지 아키텍처 설계에 적합합니다.

기업들의 사용이 증가하고 있는 클라우드 서비스가 증가하면서 확장성이 필요하고 클라우드 서비스 입구의 병목현상을 해결하고 이의 확장성을 제공하는 로드밸런싱을 구성하는 것이 일반화되고 있고, 중요 구간에 이중화가 가능한 엔터프라이즈 클래스 수준의 인프라 서비스 구성이 필요 합니다.

특징	TCP (Transmission Control Protocol)	UDP (User Datagram Protocol)
클라이언트와 서버 간 연결	가능, 1:1	불가능
데이터	세그먼트로 그룹화된 바이트 스트림	데이터그램
신뢰성	가능, 누적 수신 거부, 시간 초과, 재전송	불가능
시퀀싱	가능	불가능
흐름/혼잡 제어	가능, 윈도우 기반	불가능
전이중	가능	가능

표 1. 트랜스포트 계층 프로토콜 TCP/UDP 기능 비교

2. LoxiLB Enterprise 패키지 구성

LoxiLB Enterprise Version 1.0 기반의 패키지는 엔터프라이즈 기능을 제공하는 'loxilb' 와 CLI 명령어도구 'loxicmd'를 포함합니다. 그리고, 그래픽 기반 관리를 위한 UI를 선택하여 적용할 수 있습니다.

그림 2. LoxiLB Enterprise 기능의 구성

3. LoxiLB Enterprise의 동작

LoxiLB Enterprise는 엔터프라이즈용 'loxilb'를 클라우드 서비스 외부(external mode) 환경에서 필요한 다양한 요구에 맞추어 설치할 수 있습니다.

고가용성이 필요한 구성을 위해 적용 시나리오별 로드밸런서 'loxilb'를 배포하는 방법에 필요한 기능들을 제공합니다.

엔터프라이즈 'loxilb'의 동작 모드

대부분의 온프레미스 환경이나 퍼블릭 클라우드에서는 일반적으로 로드밸런서/방화벽을 클라우드 네이티브 워크로드와 분리하여 외부 노드 가상머신(VM) 또는 베어메탈에서 실행합니다. 이런 환경에서 'loxilb'를 외부에서 연동합니다.

클라우드의 경우 VM 또는 인스턴스를 생성하여 'loxilb' 컨테이너를 실행합니다. 온프레미스 환경에서는 예비 노드/VM을 사용자가 따로 준비하여 'loxilb' 컨테이너를 실행 합니다. 'loxilb' 컨테이너는 클라우드 클러스터와 독립적이며 docker, Containerd, Podman 같은 컨테이너 엔진으로 관리할 수 있습니다.

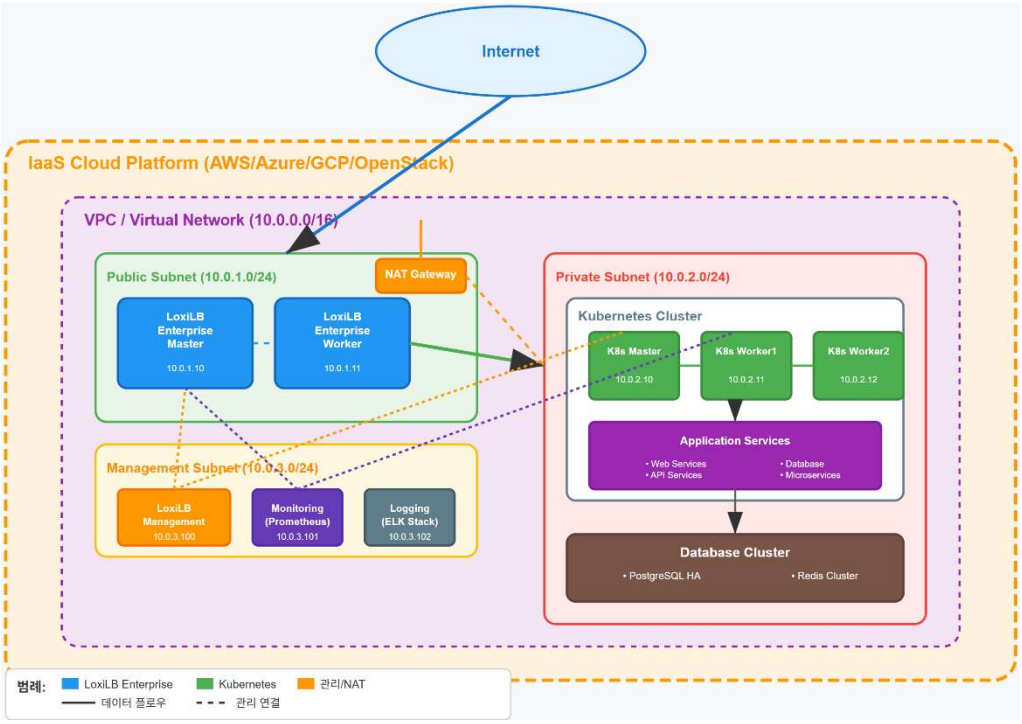

그림 3. 클라우드를 위한 LoxiLB Enterprise 구성 (예)

‘loxilb’의 로드밸런서 알고리즘

- Round-Robin (rr)
- Weighted round-robin (wrr)
- Persistence (persist)
- Flow-hash (hash)
- Least-Connections (lc)

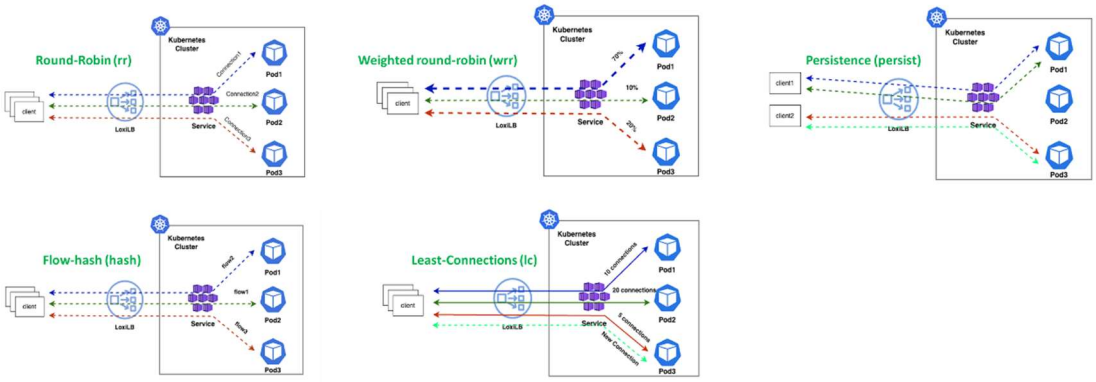

그림 4. Load-balancer algorithms

‘loxilb’ NAT 모드

- Normal NAT
- One-ARM
- Full-NAT
- L2-DSR mode
- L3-DSR mode

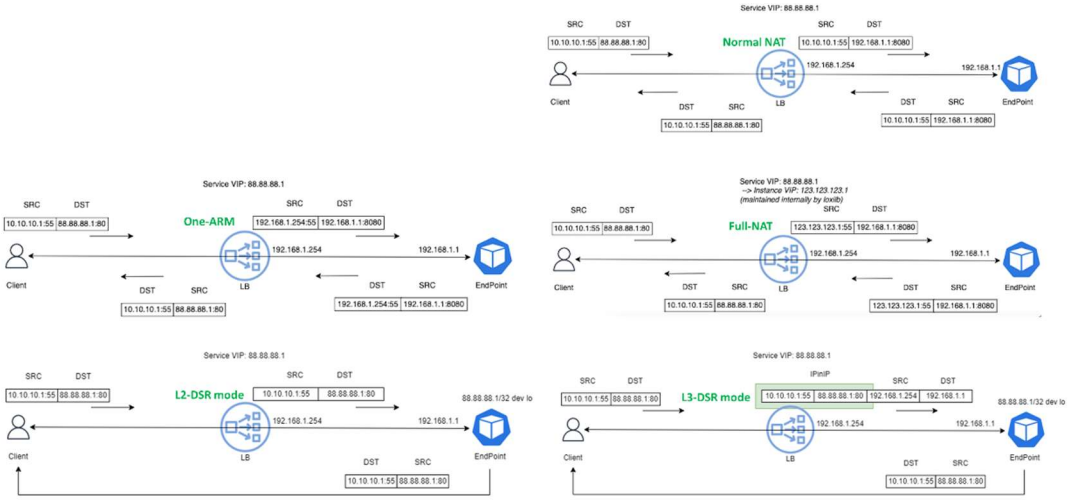

그림 5. NAT modes

SUMMARY

LoxiLB Enterprise는 eBPF를 엔진을 사용하는 클라우드용 로드 밸런싱 솔루션이며, 클라우드 워크로드를 위해 설계된 오픈 소스 하이퍼 스케일 소프트웨어 로드 밸런서입니다.

온프레미스, 엣지, 퍼블릭 클라우드 애플리케이션을 지원하는 독립형 로드 밸런서로 작동하며 고속의 유연하고 프로그래밍 가능한 로드 밸런싱 서비스를 제공합니다.

배포, 부트스트랩, 구성, 프로비저닝, 확장, 업그레이드, 마이그레이션, 라우팅, 모니터링 및 리소스 관리와 같은 로드 밸런서 관리 작업과 함께 고가용성(HA) 등의 자동화를 제공합니다.

CHAPTER

02

기본 설치

개요

빌드된 도커(Docker) 이미지를 사용하여 컨테이너로 실행하여 설정하며 LoxiLB를 독립적인 단독모드(Standalone mode)로 작동하는 방법에 대해서 설명합니다. 그리고, 인클러스터 모드(In-cluster mode)는 'kube-loxilb'가 자동으로 'loxilb'를 구성하므로 별도 설치가 필요 없지만 익스터널모드(External mode)에서도 단독모드(Standalone mode)와 같이 'loxilb' 설치가 필요 합니다.

---

1. 사전 준비

'LoxiLB Enterprise'는 클라우드 서비스 외부에 엔터프라이즈용 'loxilb'를 설치하여 구성합니다.

'loxilb' 최소 시스템 요구사항

- 'loxilb' 실행용 Linux 설치 노드 (VM 또는 베어메탈)
- Linux 커널 버전 5.15 이상 (eBPF 지원을 확인)
- 지원 확인 운영체제: Ubuntu 20.04 이상 또는 Rocky Linux 9 이상
- CPU: 2 vCPU 이상
- 메모리: 최소 4GB (8GB 이상을 권장)
- 저장 공간: 10GB 이상의 여유 디스크 공간
- 네트워크: 최소 1개의 인터페이스
- Docker 설치

'loxilb' 설치 기기(가상머신 또는 베어메탈) 사전 점검

- 기기에 Docker 가 설치되어 있는가?
- 호스트 네트워크등 LB 를위한 인터페이스가 구성되어 있는가?
- 방화벽 및 보안 정책이 외부 통신 및 API 사용을 허용하는가?
- 테스트용 외부 IP 및 CIDR 블록이 준비되어 있는가?

Docker 설치 (Ubuntu 20.04 이상, Rocky Linux)

아래와 같이 Ubuntu계열이나 Rocky Linux 계열 등에 Docker를 설치합니다.

```
### Docker Installation @ Ubuntu 20.04 or higher (예)

sudo apt update
sudo apt install docker.io
```

```
### Docker Installation @ Rocky Linux 9.5 (예)

sudo dnf install -y dnf-utils
sudo dnf config-manager --add-repo https://download.docker.com/linux/rhel/docker-ce.repo
sudo dnf -y install docker-ce docker-ce-cli containerd.io docker-compose-plugin
sudo systemctl enable docker
sudo systemctl start docker
sudo usermod -a -G docker $(whoami)
```

Docker 서비스 활성화 명령어 (필요시 선택)

```
sudo systemctl enable docker
sudo systemctl start docker
```

2. 'loxilb' 설치 및 배포

- LoxiLB Enterprise 용 별도 제공 'loxilb' Docker 이미지 사용 로드:

```
sudo docker load -i loxilb-enterprise-1.0.0.tar
```

- 오픈소스 'loxilb' Docker 이미지 다운로드 (선택):

```
sudo docker pull ghcr.io/loxilb-io/loxilb:latest
```

- 'loxilb' 컨테이너 실행 (** 'LoxiLB Enterprise'의 'loxilb' 이미지 이름 적용)

```
sudo docker run -u root --cap-add SYS_ADMIN --restart unless-stopped --privileged --entrypoint "/root/loxilb-io/loxilb/loxilb" -dit -v /dev/log:/dev/log --net=host --name loxilb ghcr.io/loxilb-io/loxilb:latest
```

- 'loxicmd' 사용을 위한 'loxilb' 터미널 연결

```
sudo docker exec -it loxilb bash
```

- 'loxicmd' 명령어를 사용하는 수동 구성/관리 가능 (예)

loxicmd 명령어를 통해 LB, Endpoint의 설정 가능 (예):

loxicmd create lb 192.168.1.9 --tcp=80:8080 --endpoints=10.0.0.1:1
loxicmd create lb 192.168.1.9 --tcp=80:30458 --endpoints=192.168.1.31:1,192.168.1.32:1 --monitor

'loxilb' 컨테이너 실행 후 상태와 제조사 요구 시 필요한 로그 확인 명령어 제공:

docker ps | grep loxilb
docker logs loxilb

```
[loxilb@localhost ~]$ docker ps | grep loxilb
392ad81f856f   ghcr.io/loxilb-io/loxilb:latest   "/root/loxilb-io/lox..."   2 weeks ago   Up 4 minutes                loxilb
[loxilb@localhost ~]$ docker logs loxilb
loxilb start
05:46:49 DEBUG loxilb_libdp.c:3506: /opt/loxilb/llb_xdp_main.o: nr 0 psection xdp_packet_hook
05:46:58 DEBUG loxilb_libdp.c:3272: llb_link_prop_add: IF-llb0 added idx 0 type 1
05:46:58 DEBUG loxilb_libdp.c:3525: setting up xdp for llb0|xdp_packet_hook
05:46:58 DEBUG loxilb_libdp.c:3347: llb_psec_add: SEC-tc_packet_hook0 added idx 0
05:46:58 DEBUG loxilb_libdp.c:3506: /opt/loxilb/llb_ebpf_main.o: nr 0 psection tc_packet_hook0
05:46:58 INFO  common_libbpf.c:121: tc: bpf attach start for llb0:0
05:46:58 ERROR common_libbpf.c:141: tc: no obj for pinpath /opt/loxilb/dp/bpf/llb_ebpf.rodata
05:46:58 DEBUG common_libbpf.c:161: tc: autoload sec tc_packet_hook1 prog tc_packet_func
05:46:58 DEBUG common_libbpf.c:161: tc: autoload sec tc_packet_hook2 prog tc_packet_func_slow
05:46:58 DEBUG common_libbpf.c:161: tc: autoload sec tc_packet_hook3 prog tc_packet_func_fw
05:46:58 DEBUG common_libbpf.c:161: tc: autoload sec tc_packet_hook4 prog tc_csum_func1
05:46:58 DEBUG common_libbpf.c:161: tc: autoload sec tc_packet_hook5 prog tc_csum_func2
05:46:58 DEBUG common_libbpf.c:161: tc: autoload sec tc_packet_hook6 prog tc_slow_unp_func
05:46:58 DEBUG common_libbpf.c:161: tc: autoload sec tc_packet_hook7 prog tc_packet_func_masq
05:46:59 INFO  common_libbpf.c:204: tc: bpf attach OK for llb0
05:46:59 DEBUG loxilb_libdp.c:3248: llb_link_prop_add: IF-llb0 ref idx 0:1 type 2
2025-04-22 05:46:59 ebpf unload - lo
libbpf: Kernel error message: Cannot find specified qdisc on specified device
2025-04-22 05:46:59 ebpf unload - docker0
libbpf: Kernel error message: Cannot find specified qdisc on specified device
05:46:59 ERROR common_libbpf.c:94: tc: bpf hook destroy failed for docker0:0
2025/04/22 05:47:00 Serving loxilb rest API at http://[::]:11111
2025-04-22 05:47:02 nlp: Link msgs subscribed
2025-04-22 05:47:02 nlp: Addr msgs subscribed
2025-04-22 05:47:02 nlp: Neigh msgs subscribed
2025-04-22 05:47:02 nlp: Route msgs subscribed
2025-04-22 05:47:02 nlp: NLP Subscription done
2025-04-22 05:47:02 nlp: Getting device info
2025-04-22 05:47:02 nlp: ADD dev docker0 mac([46 234 67 17 110 75]) attrs(&{3 1500 0 docker0 2e:ea:43:11:6e:4b up|broadcast|multica
st 4099 0 0 <nil>  0xc00274cf00 0 0 1 0xc0049c76c8 ether <nil> down 0 -1 1 1 65536 65535 [] 0 <nil>}) - info recvd
2025-04-22 05:47:02 port add - docker0 isPvid false
2025-04-22 05:47:02 port added - docker0:1 OSID 3
2025-04-22 05:47:02 vlan 4090 bd created
2025-04-22 05:47:02 nlp: Bridge docker0, 4090, [46 234 67 17 110 75], false, 1500 ADD [OK]
2025-04-22 05:47:02 nlp: ADD dev ens160 mac([0 12 41 223 240 137]) attrs(&{2 1500 1000 ens160 00:0c:29:df:f0:89 up|broadcast|multic
ast 69699 0 0 <nil>  0xc00274d440 0 0 1 0xc0049c78c0 ether <nil> up 0 -1 2 2 65536 65535 [] 0 <nil>}) - info recvd
2025-04-22 05:47:02 port add - vlan3802 isPvid false
2025-04-22 05:47:02 port added - vlan3802:3 OSID -1
2025-04-22 05:47:02 vlan 3802 bd created
2025-04-22 05:47:02 port added - ens160:2 OSID 2
2025-04-22 05:47:02 nlp: Port ens160, [0 12 41 223 240 137], true, 1500 add [OK]
2025-04-22 05:47:02 nlp: ADD dev docker0 mac([46 234 67 17 110 75]) attrs(&{3 1500 0 docker0 2e:ea:43:11:6e:4b up|broadcast|multica
```

```
st 4099 0 0 <nil>  0xc00274d5c0 0 0 1 0xc0049c78d8 ether <nil> down 0 -1 1 1 65536 65535 [] 0 <nil>}) - info recvd
2025-04-22 05:47:02 nlp: Bridge docker0, 4090, [46 234 67 17 110 75], false, 1500 ADD [OK]
2025-04-22 05:47:02 nlp: ADD dev llb0 mac([0 0 202 254 250 206]) attrs(&{4 1500 1000 llb0 00:00:ca:fe:fa:ce up|broadcast|multicast 696
99 0 0 <nil>  0xc00274d740 0 0 1 0xc0049c78f0 ether <nil> unknown 0 -1 1 1 65536 65535 [] 0 <nil>}) - info recvd
2025-04-22 05:47:02 port add - vlan3804 isPvid false
2025-04-22 05:47:02 port added - vlan3804:5 OSID -1
2025-04-22 05:47:02 vlan 3804 bd created
2025-04-22 05:47:02 port added - llb0:4 OSID 4
2025-04-22 05:47:02 nlp: Port llb0, [0 0 202 254 250 206], true, 1500 add [OK]
2025-04-22 05:47:02 rt added - 127.0.0.0/8:root mark:
2025-04-22 05:47:02 rt added - 127.0.0.1/32:root mark:
2025-04-22 05:47:02 ifa added 127.0.0.1:lo
2025-04-22 05:47:02 nlp: IPv4 Address 127.0.0.1/8 Port lo added
2025-04-22 05:47:02 rt added - ::1/128:root mark:
2025-04-22 05:47:02 ifa added ::1:lo
2025-04-22 05:47:02 nlp: IPv4 Address ::1/128 Port lo added
2025-04-22 05:47:02 rt added - 192.168.191.0/24:root mark:
2025-04-22 05:47:02 rt added - 192.168.191.137/32:root mark:
2025-04-22 05:47:02 ifa added 192.168.191.137:ens160
2025-04-22 05:47:02 nlp: IPv4 Address 192.168.191.137/24 Port ens160 added
2025-04-22 05:47:02 rt added - fe80::/64:root mark:
2025-04-22 05:47:02 rt added - fe80::/32:root mark:
2025-04-22 05:47:02 ifa added fe80::20c:29ff:fedf:f089:ens160
2025-04-22 05:47:02 nlp: IPv4 Address fe80::20c:29ff:fedf:f089/64 Port ens160 added
2025-04-22 05:47:02 neigh rtpair - 192.168.191.1/32->192.168.191.1
2025-04-22 05:47:02 rt added - 192.168.191.1/32:root mark:1
2025-04-22 05:47:02 added fdb ent, {[0 80 86 192 0 8] 3802} : health(true)
```

설치 리눅스 OS별 'loxilb' 설치 환경 확인

'loxilb' 설치에 사용하는 OS 커널 버전을 '그림 5. loxilb 설치 시험 OS커널 확인 (예시)'와 같이 "uname -a'와 'hostnamectl' 명령어로 확인할 수 있습니다.

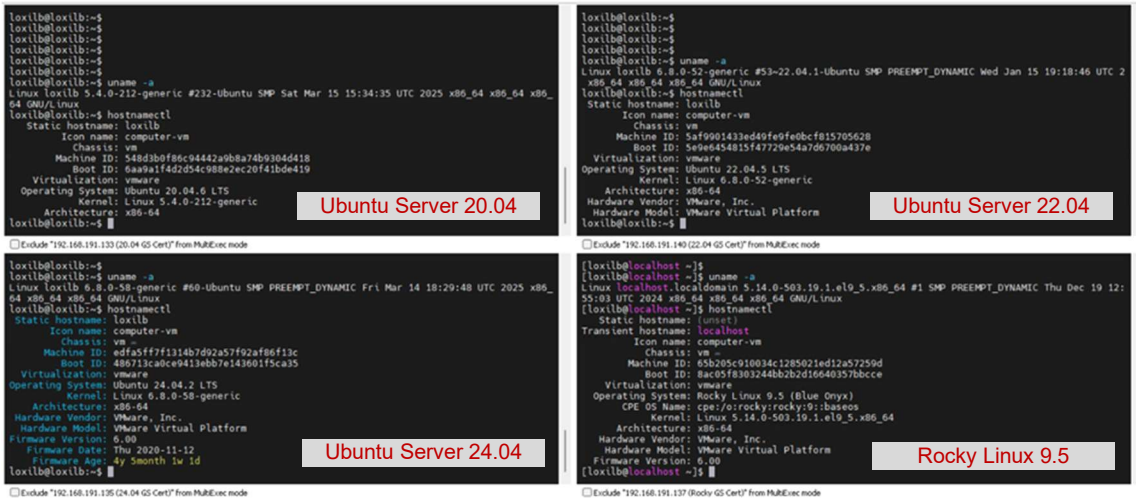

그림 5. Loxilb 설치 시험 OS 커널 확인 (예시)

설치한 'loxilb' 버전을 아래 예시와 같이 확인할 수 있습니다.

- sudo docker ps
- sudo docker exec -it loxilb bash
- loxicmd version
- loxicmd get lbversion

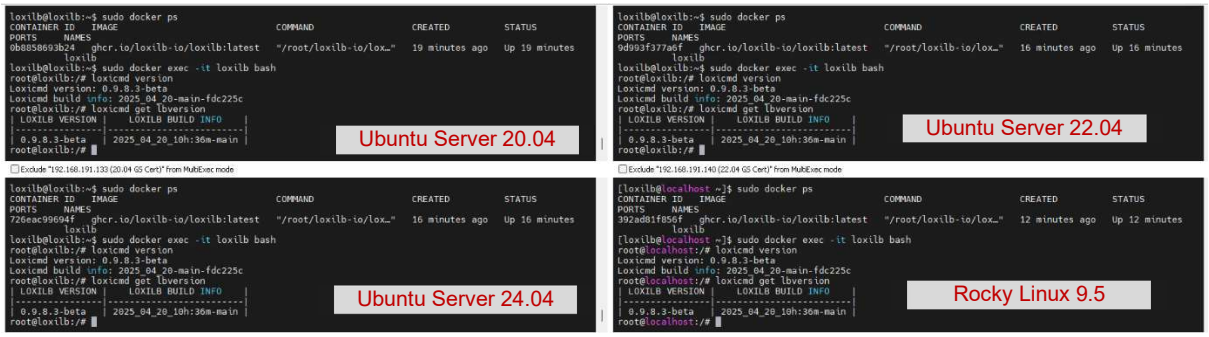

그림 6. 시험 OS 상의 'loxilb' 설치 버전 확인

사용 OS 종류	Linux Kernel Version
Ubuntu Server 20.04	Linux 5.4.0-212-generic
Ubuntu Server 22.04	Linux 6.8.0-52-generic
Ubuntu Server 24.04	Linux 6.8.0-58-generic
Rocky Linux 9.5	Linux 5.14.0-503.19.1.el9_5.x86_64

'loxilb' 기능 종료 및 재실행

아래와 같이 도커(Docker)컨테이너 실행 종료(stop)와 실행 명령을 사용합니다.

```
sudo docker stop loxilb
sudo docker start loxilb
```

SUMMARY

빌드된 도커(Docker) 이미지를 사용하여 컨테이너로 실행하여 설정하며 LoxiLB Enterprise를 위한 'loxilb'로 작동합니다.

CHAPTER

03

CLI 명령어 (loxicmd)

개요

LoxiLB Enterprise를 위한 설정은 'loxilb'의 명령어 도구(Command Tool) 'loxicmd'를 사용합니다. 'loxicmd'는 LoxiLB Enterprise의 LB 정책 설정, 디버깅, 모니터링 등을 위해 사용합니다.

---

1. 'loxilb' 명령어 개요

loxicmd는 eBPF 기반 클라우드 환경을 위한 로드밸런서인 Enterprise LoxiLB를 관리하기 위한 명령어 도구입니다.

주요 명령어 카테고리

- **apply**: 구성 파일 적용
- **create**: 리소스 생성
- **delete**: 리소스 삭제
- **get**: 리소스 조회
- **set**: 설정 변경
- **save**: 현재 구성 저장

지원 기능

- **로드 밸런서:** 생성, 삭제, 조회
- **엔드포인트 관리:** 헬스 체크 및 모니터링
- **네트워크 구성:** VLAN, VXLAN, 라우팅
- **방화벽:** 규칙 생성 및 관리
- **정책:** 트래픽 제어 정책

2. loxicmd 명령어

loxicmd는 loxilb용 CLI(Command Line Interface) 도구입니다.

아래와 같이 'loxicmd --help'로 loxicmd 명령어를 확인할 수 있습니다.

- apply Apply configuration
- completion Generate completion script
- create Create a Load balance features in the LoxiLB.
- delete Delete a Load balance features in the LoxiLB.
- get Get a Load balance features from LoxiLB.
- help Help about any command
- save saves current configuration
- set Set configurations
- version Get a version

```
root@loxilb:/# loxicmd --help
loxicmd is the command-line tool for loxilb. It is equivalent of "kubectl" for loxilb. loxicmd provides the foll
owing (currently) :
        - Create/Delete/Get - Service type external load-balancer, Vlan, Vxlan, Qos Policies, Endpoint client,FD
B, IPaddress, Neighbor, Route,Firewall, Mirror, Session, UICI
        - Get Port(interface) dump used by loxilb or its docker
        - Get Connection track (TCP/UDP/ICMP/SCTP) information
loxicmd aim to provide all of the configuation for the loxilb.

Usage:
  loxicmd [command]

Available Commands:
  apply       Apply configuration
  completion  Generate completion script
  create      Create a Load balance features in the LoxiLB.
  delete      Delete a Load balance features in the LoxiLB.
  get         Get a Load balance features from LoxiLB.
  help        Help about any command
  save        saves current configuration
  set         Set configurations
  version     Get a version

Flags:
  -s, --apiserver string   Set API server IP address (default "127.0.0.1")
  -h, --help               help for loxicmd
  -o, --output string      Set output layer (ex.) wide, json)
  -p, --port int16         Set API server port number (default 11111)
      --protocol string    Set API server http/https (default "http")
  -t, --timeout int16      Set timeout (default 10)
      --token string       Set Token for the API server

Use "loxicmd [command] --help" for more information about a command.
```

아래의 'loxicmd save' 명령어로 'loxilb' 설정을 저장할 수 잇습니다.

```
loxicmd save --all
```

apply 명령어

apply 명령어는 텍스트 파일에서 구성을 읽고 적용합니다.

```
root@loxilb:/# loxicmd apply --help
Reads and apply configuration from the text file

Usage:
  loxicmd apply [flags]

Flags:
      --bfd string           BFD Config file to apply
  -c, --config-path string   Configuration path only for applying per interface config (default "/et
c/loxilb/ipconfig/")
  -f, --file string          Config file to apply as like K8s
      --firewall string      Firewall config file to apply
  -h, --help                 help for apply
  -i, --ip string            IP config file to apply
  -r, --ipv4route            Apply route configuration only for specific interface
  -l, --lb string            Load Balancer config file to apply
      --per-intf string      Apply configuration only for specific interface
      --session string       Session config file to apply
      --ulcl string          Ulcl config file to apply

Global Flags:
  -s, --apiserver string   Set API server IP address (default "127.0.0.1")
  -o, --output string      Set output layer (ex.) wide, json)
  -p, --port int16         Set API server port number (default 11111)
      --protocol string    Set API server http/https (default "http")
  -t, --timeout int16      Set timeout (default 10)
      --token string       Set Token for the API server
root@loxilb:/#
```

Usage Examples

```
# Apply load balancer configuration
loxicmd apply --lb /path/to/lb-config.yaml

# Apply firewall rules
loxicmd apply --firewall /path/to/firewall-rules.yaml

# Apply configuration to specific interface
loxicmd apply --per-intf eth0 --ip /path/to/ip-config.yaml

# Use custom API server
loxicmd apply -s 192.168.1.100 -p 8080 --lb /path/to/config.yaml
```

completion 명령어

completion 명령어는 쉘 자동완성 스크립트를 생성하는 명령어입니다. 이 명령어는 텍스트 파일을 사용하지 않고, 자동완성 스크립트를 생성하여 출력합니다.

아래와 같이 'loxicmd completion --help'로 completion 설정 값을 확인할 수 있습니다.

```
root@loxilb:/# loxicmd completion --help
To load completions

Usage:
  loxicmd completion [bash|zsh|fish|powershell]

Flags:
  -h, --help   help for completion

Global Flags:
  -s, --apiserver string   Set API server IP address (default "127.0.0.1")
  -o, --output string      Set output layer (ex.) wide, json)
  -p, --port int16         Set API server port number (default 11111)
      --protocol string    Set API server http/https (default "http")
  -t, --timeout int16      Set timeout (default 10)
      --token string       Set Token for the API server
```

create 명령어

create 명령어는 아래와 같이 LB 정책 등을 생성할 수 있습니다.

- bfd Create a BFD session
- endpoint Create a LB EndPoint for monitoring
- firewall Create a Firewall
- ip Create a IPv4Address
- lb Create a LoadBalancer
- mirror Create a Mirror
- neighbor Create a Neighbors
- policy Create a Policy
- route Create a Route
- vlan Create a vlan
- vlanmember Create a vlanmember
- vxlan Create a vxlan
- vxlanpeer Create a vxlan

아래와 같이 'loxicmd create --help' 명령어는 생성에 관련한 명령어들을 볼 수 있습니다.

```
root@loxilb:/# loxicmd create —help
Create a Load balance features in the LoxiLB.
Create - Service type external load-balancer, Vlan, Vxlan, Qos Policies,
         Endpoint client,FDB, IPaddress, Neighbor, Route,Firewall, Mirror, Session, UICI

Usage:
  loxicmd create [flags]
  loxicmd create [command]

Available Commands:
  bfd         Create a BFD session
  endpoint    Create a LB EndPoint for monitoring
  firewall    Create a Firewall
  ip          Create a IPv4Address
  lb          Create a LoadBalancer
  mirror      Create a Mirror
  policy      Create a Policy
  route       Create a Route
  vlan        Create a vlan
  vlanmember  Create a vlanmember
  vxlan       Create a vxlan
  vxlanpeer   Create a vxlan

Flags:
  -h, —help   help for create

Global Flags:
  -s, —apiserver string   Set API server IP address (default "127.0.0.1")
```

```
  -o, --output string      Set output layer (ex.) wide, json)
  -p, --port int16         Set API server port number (default 11111)
      --protocol string    Set API server http/https (default "http")
  -t, --timeout int16      Set timeout (default 10)
      --token string       Set Token for the API server

Use "loxicmd create [command] --help" for more information about a command.
```

create bfd 명령어

loxicmd create bfd 명령어는 HA 구성 loxilb간 서로의 상태를 직접 확인하여 빠른 회복력은 제공합니다. 아래와 같이 'loxicmd create bfd --help' 명령어는 bfd에 관련한 설정 값들을 볼 수 있습니다.

```
root@loxilb:/# loxicmd create bfd --help
Create a BFD session for HA failover

ex) loxicmd create bfd 32.32.32.2 --instance=default --sourceIP=32.32.32.1 --interval=200000 --retryCount=3

Usage:
  loxicmd create bfd remoteIP [--instance=<instance>] [--sourceIP=<source-IP>] [--interval=<interval>] [--retryC
ount=<count>] [flags]

Aliases:
  bfd, bfd-session

Flags:
  -h, --help               help for bfd
      --instance string    Specify the cluster instance name (default "default")
      --interval uint      Specify the BFD packet tx interval (in microseconds) (default 200000)
      --retryCount uint8   Specify the number of reties (default 3)
      --sourceIP string    Specify the source IP for the session

Global Flags:
  -s, --apiserver string   Set API server IP address (default "127.0.0.1")
  -o, --output string      Set output layer (ex.) wide, json)
  -p, --port int16         Set API server port number (default 11111)
      --protocol string    Set API server http/https (default "http")
  -t, --timeout int16      Set timeout (default 10)
      --token string       Set Token for the API server
```

아래(예)와 같이 'loxilb' 실행 시 Master와 Backup 설정이 가능 합니다.

```
# BFD 세션 생성 (Master 'loxilb')
sudo docker exec -it loxilb-master bash
loxicmd create bfd 192.168.1.81 --instance=llb-inst0 --sourceIP=192.168.1.91 --interval=200000 --retryCount=3


# BFD 세션 생성 (Backup 'loxilb)
sudo docker exec -it loxilb-backup bash
loxicmd create bfd 192.168.1.91 --instance=llb-inst0   --sourceIP=192.168.1.81 --interval=200000 --retryCount=3
```

create endpoint 명령어

loxicmd create endpoint 명령어는 loxilb의 LB를 위해 엔드포인트를 지정합니다. 아래와 같이 'loxicmd create endpoint --help' 명령어는 endpoint에 관련한 설정 값들을 볼 수 있습니다.

```
root@loxilb:/# loxicmd create endpoint --help
Create a LB EndPoint for monitoring using LoxiLB

ex) loxicmd create endpoint 32.32.32.1 --name=32.32.32.1_http_8080 --probetype=http --probeport=8080 --period=60 --retries=2

Usage:
  loxicmd create endpoint IP [--name=<id>] [--probetype=<probetype>] [--probereq=<probereq>] [--proberesp=<probe
resp>] [--probeport=<port>] [--period=<period>] [--retries=<retries>] [flags]

Aliases:
  endpoint, Endpoint, ep, endpoints

Flags:
  -h, --help                help for endpoint
      --name string         Endpoint Identifier
      --period int          Period of probing (default 60)
      --probeport int       If probe is http,https,tcp,udp,sctp one can specify custom l4port to use
      --probereq string     If probe is http/https, one can specify additional uri path
      --proberesp string    If probe is http/https, one can specify custom response string
      --probetype string    Probe-type:ping,http,https,udp,tcp,sctp,none (default "ping")
      --retries int         Number of retries before marking endPoint inactive (default 2)

Global Flags:
  -s, --apiserver string    Set API server IP address (default "127.0.0.1")
  -o, --output string       Set output layer (ex.) wide, json)
  -p, --port int16          Set API server port number (default 11111)
      --protocol string     Set API server http/https (default "http")
  -t, --timeout int16       Set timeout (default 10)
      --token string        Set Token for the API server
```

아래(예)와 같이 endpoint로 헬스체크 설정이 가능 합니다.

```
root@loxilb:/# loxicmd create endpoint 192.168.1.31 --name=web-server-1 --probetype=http --probeport=8080 --period=60 --
retries=2
Debug: response.StatusCode: 200
Success
root@loxilb:/# loxicmd create endpoint 192.168.1.32 --name=web-server-2 --probetype=ping --period=30 --retries=3
Debug: response.StatusCode: 200
Success
root@loxilb:/# loxicmd get endpoint
|     HOST     |          NAME          | PTYPE | PORT  | DURATION | RETRIES | MINDELAY  | AVGDELAY  | MAXDELAY  | STATE |
|--------------|------------------------|-------|-------|----------|---------|-----------|-----------|-----------|-------|
| 10.212.0.1   | 10.212.0.1_tcp_80      | none: |    80 |        0 |       0 |           |           |           | ok    |
| 10.212.0.2   | 10.212.0.2_tcp_80      | none: |    80 |        0 |       0 |           |           |           | ok    |
| 10.212.0.3   | 10.212.0.3_tcp_80      | none: |    80 |        0 |       0 |           |           |           | ok    |
| 192.168.1.31 | web-server-1           | http: |  8080 |       60 |       2 |           |           |           | nok   |
| 192.168.1.31 | 192.168.1.31_tcp_8080  | tcp:  |  8080 |       60 |       2 |           |           |           | nok   |
| 192.168.1.31 | 192.168.1.31_tcp_55555 | none: | 55555 |        0 |       0 |           |           |           | ok    |
| 192.168.1.32 | web-server-2           | ping: |     0 |       30 |       3 | 743.777µs | 824.179µs | 891.967µs | ok    |
| 192.168.1.32 | 192.168.1.32_tcp_8080  | tcp:  |  8080 |       60 |       2 |           |           |           | nok   |
| 192.168.1.32 | 192.168.1.32_tcp_55555 | none: | 55555 |        0 |       0 |           |           |           | ok    |
| 31.31.31.1   | 31.31.31.1_tcp_8080    | none: |  8080 |        0 |       0 |           |           |           | ok    |
| 32.32.32.1   | 32.32.32.1_tcp_8080    | none: |  8080 |        0 |       0 |           |           |           | ok    |
| 33.33.33.1   | 33.33.33.1_tcp_8080    | none: |  8080 |        0 |       0 |           |           |           | ok    |
```

create firewall 명령어

loxicmd create firewall 명령어는 loxilb에 방화벽 정책을 설정할 수 있습니다. 아래와 같이 'loxicmd create firewall --help' 명령어는 firewall에 관련한 설정 값들을 볼 수 있습니다.

```
root@loxilb:/# loxicmd create firewall --help
Create a Firewall using LoxiLB

--<ruleKey>s of firewallRule
sourceIP(string) - Source IP in CIDR notation
destinationIP(string) - Destination IP in CIDR notation
minSourcePort(int) - Minimum source port range
maxSourcePort(int) - Maximum source port range
minDestinationPort(int) - Minimum destination port range
maxDestinationPort(int) - Maximum source port range
protocol(int) - the protocol
portName(string) - the incoming port
preference(int) - User preference for ordering


ex) loxicmd create firewall --firewallRule="sourceIP:1.2.3.2/32,destinationIP:2.3.1.2/32,preference:200" --allow
    loxicmd create firewall --firewallRule="sourceIP:1.2.3.2/32,destinationIP:2.3.1.2/32,preference:200" --allow --record
    loxicmd create firewall --firewallRule="sourceIP:1.2.3.2/32,destinationIP:2.3.1.2/32,preference:200" --allow --setmark=10
    loxicmd create firewall --firewallRule="sourceIP:1.2.3.2/32,destinationIP:2.3.1.2/32,preference:200" --drop
    loxicmd create firewall --firewallRule="sourceIP:3ffe::1/128" --drop
    loxicmd create firewall --firewallRule="sourceIP:1.2.3.2/32,destinationIP:2.3.1.2/32,preference:200" --trap
    loxicmd create firewall --firewallRule="sourceIP:1.2.3.2/32,destinationIP:2.3.1.2/32,preference:200" --redirect=eth1
    loxicmd create firewall --firewallRule="sourceIP:1.2.3.2/32,destinationIP:2.3.1.2/32,preference:200" --snat=10.10.10.1,3030
    loxicmd create firewall --firewallRule="sourceIP:1.2.3.2/32,destinationIP:2.3.1.2/32,preference:200" --snat=10.10.10.1 (Do not change sourceport)
    loxicmd create firewall --firewallRule="sourceIP:1.2.3.2/32,destinationIP:2.3.1.2/32,preference:200" --snat=10.10.10.1,3030 --egress (Egress rules
match for non-k8s traffic)

Usage:
  loxicmd create firewall --firewallRule=<ruleKey>:<ruleValue>, [--allow] [--drop] [--trap] [--record] [--egres
s] [--redirect=<PortName>] [--setmark=<FwMark>] [flags]

Aliases:
  firewall, Firewall, fw, firewalls

Flags:
      --allow                 Allow any matching rule
      --drop                  Drop any matching rule
      --egress                Specify that this an egress rule (to be used with snat)
      --firewallRule strings  Information related to firewall rule
  -h, --help                  help for firewall
      --record                Record/Dump any matching rule
      --redirect strings      Redirect any matching rule
      --setmark uint32        Add a fw mark
      --snat strings          SNAT any matching rule
      --trap                  Trap anything matching rule

Global Flags:
  -s, --apiserver string   Set API server IP address (default "127.0.0.1")
  -o, --output string      Set output layer (ex.) wide, json)
  -p, --port int16         Set API server port number (default 11111)
      --protocol string    Set API server http/https (default "http")
  -t, --timeout int16      Set timeout (default 10)
      --token string       Set Token for the API server
```

아래(예)와 같이 firewall 명령어로 방화벽 설정과 관리가 가능 합니다.

```
root@loxilb:/# loxicmd create firewall --firewallRule="sourceIP:192.168.1.0/24,destinationIP:192.168.1.31/32,preference:100" --allow
Debug: response.StatusCode: 200
Success
root@loxilb:/# loxicmd create firewall --firewallRule="sourceIP:10.0.0.0/8,preference:200" --drop
Debug: response.StatusCode: 200
Success
root@loxilb:/# loxicmd create firewall --firewallRule="sourceIP:192.168.1.31/32,preference:300" --snat=10.10.10.1,3030
Debug: response.StatusCode: 200
Success
root@loxilb:/# loxicmd get firewall -o wide
|    SOURCE IP    | DESTINATION IP  | MIN SPORT | MAX SPORT | MIN DPORT | MAX DPORT | PROTOCOL | PORT NAME | PREFERENCE |        OPTION          | COUNTERS |
|-----------------|-----------------|-----------|-----------|-----------|-----------|----------|-----------|------------|------------------------|----------|
| 1.1.1.1/32      | 2.3.1.2/32      |         0 |         0 |         0 |         0 |        0 |           |        200 | Allow,FwMark(10)       | 0:0      |
| 1.2.3.2/32      | 2.3.1.2/32      |         0 |         0 |         0 |         0 |        0 |           |        200 | Snat(10.10.10.1:3030)  | 0:0      |
| 10.0.0.0/8      | 0.0.0.0/0       |         0 |         0 |         0 |         0 |        0 |           |        200 | Drop                   | 0:0      |
| 192.168.1.0/24  | 192.168.1.31/32 |         0 |         0 |         0 |         0 |        0 |           |        100 | Allow                  | 0:0      |
| 192.168.1.31/32 | 0.0.0.0/0       |         0 |         0 |         0 |         0 |        0 |           |        300 | Snat(10.10.10.1:3030)  | 21:3759  |
root@loxilb:/#
```

```
root@loxilb:/# loxicmd create firewall --firewallRule="sourceIP:1.1.1.1/32,destinationIP:2.3.1.2/32,preference:200" --allow --setmark=10
Debug: response.StatusCode: 200
Success
root@loxilb:/# loxicmd get firewall -o wide
| SOURCE IP  | DESTINATION IP | MIN SPORT | MAX SPORT | MIN DPORT | MAX DPORT | PROTOCOL | PORT NAME | PREFERENCE |       OPTION          | COUNTERS |
|------------|----------------|-----------|-----------|-----------|-----------|----------|-----------|------------|-----------------------|----------|
| 1.1.1.1/32 | 2.3.1.2/32     |         0 |         0 |         0 |         0 |        0 |           |        200 | Allow,FwMark(10)      | 0:0      |
```

create ip 명령어

loxicmd create ip 명령어는 loxilb에서 사용하는 IP 주소를 설정할 수 있습니다. 아래와 같이 'loxicmd create ip --help' 명령어는 ip에 관련한 설정 값들을 볼 수 있습니다.

```
root@loxilb:/# loxicmd create ip --help
Create a IPv4Address using LoxiLB. It is working as "ip addr add <DeviceIPNet> dev <device>"
ex) loxicmd create ip 192.168.0.1/24 eno7

Usage:
  loxicmd create ip <DeviceIPNet> <device> [flags]

Aliases:
  ip, ipv4address, ipv4, ipaddress

Flags:
  -h, --help   help for ip

Global Flags:
  -s, --apiserver string   Set API server IP address (default "127.0.0.1")
  -o, --output string      Set output layer (ex.) wide, json)
  -p, --port int16         Set API server port number (default 11111)
      --protocol string    Set API server http/https (default "http")
  -t, --timeout int16      Set timeout (default 10)
      --token string       Set Token for the API server
```

아래(예)와 같이 ip 명령어로 IP주소 설정과 관리가 가능 합니다.

```
root@loxilb:/# loxicmd get ip
| DEVICE NAME |          IP ADDRESS          |
|-------------|------------------------------|
| docker0     | 172.17.0.1/16                |
| ens34       | 192.168.1.9/24               |
|             | fe80::20c:29ff:fe3e:2bb2/64  |
| llb0        | fe80::40d6:10ff:fe99:2b0e/64 |
| lo          | 127.0.0.1/8                  |
|             | ::1/128                      |
root@loxilb:/# loxicmd create ip 192.168.1.123/24 ens34
Success
root@loxilb:/# loxicmd get ip
| DEVICE NAME |          IP ADDRESS          |
|-------------|------------------------------|
| docker0     | 172.17.0.1/16                |
| ens34       | 192.168.1.9/24               |
|             | fe80::20c:29ff:fe3e:2bb2/64  |
|             | 192.168.1.123/24             |
| llb0        | fe80::40d6:10ff:fe99:2b0e/64 |
| lo          | 127.0.0.1/8                  |
|             | ::1/128                      |
```

create lb 명령어

loxicmd create lb 명령어는 loxilb에서 사용하는 LB 정책을 설정할 수 있습니다. 아래와 같이 'loxicmd create lb --help' 명령어는 LB 설정 값들을 볼 수 있습니다.

```
root@loxilb:/# loxicmd create lb --help
Create a LoadBalancer

--select value options
        rr - select the lb end-points based on round-robin
        hash - select the lb end-points based on hashing
        priority - select the lb based on weighted round-robin
        persist - select the lb end-point based on sender
        lc - select the lb end-point base on least connection

--mode value options
        onearm - LB put LB-IP as srcIP
        fullnat - LB put Service IP as scrIP
        dsr - LB in DSR mode allows return traffic to bypass the load balancer (only available with hash select)
        fullproxy - LB operating as a L7 proxy
        hostonearm - LB operating in host one-arm

ex)
         loxicmd create lb 192.168.0.200 --tcp=80:32015 --endpoints=10.212.0.1:1,10.212.0.2:1,10.212.0.3:1
         loxicmd create lb 192.168.0.200 --tcp=8080-8081:32015 --endpoints=10.212.0.1:1,10.212.0.2:1,10.212.0.3:1
         loxicmd create lb 192.168.0.200 --tcp=5000:5201-5300 --endpoints=10.212.0.1:1,10.212.0.2:1,10.212.0.3:1
         loxicmd create lb 192.168.0.200 --tcp=80:32015 --endpoints=10.212.0.1:1,10.212.0.2:1,10.212.0.3:1 --security=https
         loxicmd create lb 192.168.0.200 --tcp=80:32015 --endpoints=10.212.0.1:1,10.212.0.2:1,10.212.0.3:1 --host=loxilb.io
         loxicmd create lb 192.168.0.200 --tcp=80:32015 --name="http-service" --endpoints=10.212.0.1:1,10.212.0.2:1,10.212.0.3:1
         loxicmd create lb 192.168.0.200 --udp=80:32015 --endpoints=10.212.0.1:1,10.212.0.2:1,10.212.0.3:1 --mark=10
         loxicmd create lb 192.168.0.200 --tcp=80:32015 --udp=80:32015 --endpoints=10.212.0.1:1,10.212.0.2:1,10.212.0.3:1
         loxicmd create lb 192.168.0.200 --select=hash --tcp=80:32015 --endpoints=10.212.0.1:1,10.212.0.2:1,10.212.0.3:1
         loxicmd create lb 192.168.0.200 --tcp=80:80 --endpoints=10.212.0.1:1,10.212.0.2:1,10.212.0.3:1 --mode=dsr --select=hash
         loxicmd create lb 192.168.0.200 --sctp=37412:38412 --secips=192.168.0.201,192.168.0.202 --endpoints=10.212.0.1:1,10.212.0.2:1,10.212.0.3:1
         loxicmd create lb 192.168.0.200 --tcp=80:32015 --endpoints=10.212.0.1:1,10.212.0.2:1,10.212.0.3:1 --sources=10.10.10.1/32

         loxicmd create lb  2001::1 --tcp=2020:8080 --endpoints=4ffe::1:1,5ffe::1:1,6ffe::1:1
         loxicmd create lb  2001::1 --tcp=2020:8080 --endpoints=31.31.31.1:1,32.32.32.1:1,33.33.33.1:1
         loxicmd create lb 10.10.10.254 --sctp=2020:8080 --endpoints=33.33.33.1:1 --attachEP
         loxicmd create lb 100.100.100.1 --tcp=8080:80 --endpoints=10.10.10.1:1 --ppv2en

Usage:
  loxicmd create lb IP [--select=<rr|hash|priority|persist>] [--tcp=<ports>:<targetPorts>] [--udp=<ports>:<targetPorts>] [--sctp=<ports>:<targetPort
s>] [--icmp] [--mark=<val>] [--secips=<ip>,] [--sources=<ip>,] [--endpoints=<ip>:<weight>,] [--mode=<onearm|fullnat>] [--bgp] [--monitor] [--inatimeou
t=<to>] [--name=<service-name>] [--attachEP] [--detachEP] [--security=<https|e2ehttps|none>] [--host=<url>] [--ppv2en] [--egress] [flags]

Flags:
      --attachEP            Attach endpoints to the load balancer rule
      --bgp                 Enable BGP in the load balancer
      --detachEP            Detach endpoints from the load balancer rule
      --egress              Specify egress rule
      --endpoints strings   Endpoints is pairs that can be specified as '<endpointIP>:<Weight>'
  -h, --help                help for lb
      --host string         Ingress Host URL Path
      --icmp                ICMP Ping packet Load balancer
      --inatimeout uint32   Specify the timeout (in seconds) after which a LB session will be reset for inactivity
      --mark uint32         Specify the mark num to segregate a load-balancer VIP service
      --mode string         NAT mode for load balancer rule
      --monitor             Enable monitoring end-points of this rule
      --name string         Name for load balancer rule
      --ppv2en              Enable proxy procotol v2
      --sctp strings        Port pairs can be specified as '<port>:<targetPort>'
      --secips strings      Secondary IPs for SCTP multihoming rule specified as '<secondaryIP>'
      --security string     Security mode for load balancer rule
      --select string       Select the hash algorithm for the load balance.(ex) rr, hash, priority, persist, lc (default "rr")
      --sources strings     Allowed sources for this rule as '<allowedSources>'
      --tcp strings         Port pairs can be specified as '<port>:<targetPort>'
      --udp strings         Port pairs can be specified as '<port>:<targetPort>'

Global Flags:
  -s, --apiserver string   Set API server IP address (default "127.0.0.1")
  -o, --output string      Set output layer (ex.) wide, json)
  -p, --port int16         Set API server port number (default 11111)
      --protocol string    Set API server http/https (default "http")
  -t, --timeout int16      Set timeout (default 10)
      --token string       Set Token for the API server
```

아래(예)와 같이 lb 명령어로 로드밸런싱 설정과 관리가 가능 합니다.

```
root@loxilb:/# loxicmd get lb
root@loxilb:/# loxicmd create lb 192.168.1.9 --tcp=80:55555 --endpoints=192.168.1.31:1,192.168.1.32:1
ProtoPortpair: map[tcp:[80:55555]]
Debug: response.StatusCode: 200
Success
root@loxilb:/# loxicmd create lb 192.168.1.10 --select=priority --tcp=8080:55555 --endpoints=192.168.1.31:40,192.168.1.32:60
ProtoPortpair: map[tcp:[8080:55555]]
Debug: response.StatusCode: 200
Success
root@loxilb:/# loxicmd create lb 192.168.0.200 --tcp=80:80 --endpoints=10.212.0.1:1,10.212.0.2:1,10.212.0.3:1 --mode=dsr --select=hash
ProtoPortpair: map[tcp:[80:80]]
Debug: response.StatusCode: 200
Success
root@loxilb:/# loxicmd create lb 192.168.0.201 --tcp=80:80 --endpoints=10.212.0.1:1,10.212.0.2:1,10.212.0.3:1 --select=persist
ProtoPortpair: map[tcp:[80:80]]
Debug: response.StatusCode: 200
Success
root@loxilb:/# loxicmd create lb 192.168.0.202 --tcp=80:80 --endpoints=10.212.0.1:1,10.212.0.2:1,10.212.0.3:1 --select=lc
ProtoPortpair: map[tcp:[80:80]]
Debug: response.StatusCode: 200
Success
root@loxilb:/# loxicmd create lb 192.168.1.12 --tcp=2020:8080 --endpoints=192.168.1.31:1,192.168.1.32:1 --mode=onearm
ProtoPortpair: map[tcp:[2020:8080]]
Debug: response.StatusCode: 200
Success
root@loxilb:/# loxicmd create lb 192.168.1.13 --tcp=3030:8080 --endpoints=192.168.1.31:1,192.168.1.32:1 --mode=fullnat
ProtoPortpair: map[tcp:[3030:8080]]
Debug: response.StatusCode: 200
Success
root@loxilb:/# loxicmd create lb 192.168.1.14 --tcp=4040:8080 --endpoints=192.168.1.31:1,192.168.1.32:1 --mode=dsr --select=hash
ProtoPortpair: map[tcp:[4040:8080]]
Error: No port-translation in dsr mode
root@loxilb:/# loxicmd create lb 192.168.1.15 --tcp=4040:8080 --endpoints=192.168.1.41:1,192.168.1.42:1 --mode=dsr
ProtoPortpair: map[tcp:[4040:8080]]
Error: No port-translation in dsr mode
root@loxilb:/# loxicmd create lb 192.168.1.15 --tcp=4040:80 --endpoints=192.168.1.41:1,192.168.1.42:1 --mode=dsr
ProtoPortpair: map[tcp:[4040:80]]
Error: No port-translation in dsr mode
root@loxilb:/# loxicmd create lb 192.168.1.15 --tcp=80:80 --endpoints=192.168.1.41:1,192.168.1.42:1 --mode=dsr
ProtoPortpair: map[tcp:[80:80]]
Debug: response.StatusCode: 200
Error: Only Hash Selection criteria allowed for DSR mode
root@loxilb:/# loxicmd create lb 192.168.1.15 --tcp=80:80 --endpoints=192.168.1.41:1,192.168.1.42:1 --mode=dsr --select=hash
ProtoPortpair: map[tcp:[80:80]]
Debug: response.StatusCode: 200
Success
root@loxilb:/# loxicmd get lb -o wide
|    EXT IP     | SEC IPS | SOURCES | HOST | PORT | PROTO | NAME | MARK |    SEL    |   MODE    |    ENDPOINT    | EPORT | WEIGHT | STATE | COUNTERS |
|---------------|---------|---------|------|------|-------|------|------|-----------|-----------|----------------|-------|--------|-------|----------|
| 192.168.0.200 |         |         |      |   80 | tcp   |      |    0 | hash      | dsr       | 10.212.0.1     |    80 |      1 | -     | 0:0      |
|               |         |         |      |      |       |      |      |           |           | 10.212.0.2     |    80 |      1 | -     | 0:0      |
|               |         |         |      |      |       |      |      |           |           | 10.212.0.3     |    80 |      1 | -     | 0:0      |
| 192.168.0.201 |         |         |      |   80 | tcp   |      |    0 | persist   | default   | 10.212.0.1     |    80 |      1 | -     | 0:0      |
|               |         |         |      |      |       |      |      |           |           | 10.212.0.2     |    80 |      1 | -     | 0:0      |
|               |         |         |      |      |       |      |      |           |           | 10.212.0.3     |    80 |      1 | -     | 0:0      |
| 192.168.0.202 |         |         |      |   80 | tcp   |      |    0 | lc        | default   | 10.212.0.1     |    80 |      1 | -     | 0:0      |
|               |         |         |      |      |       |      |      |           |           | 10.212.0.2     |    80 |      1 | -     | 0:0      |
|               |         |         |      |      |       |      |      |           |           | 10.212.0.3     |    80 |      1 | -     | 0:0      |
| 192.168.1.10  |         |         |      | 8080 | tcp   |      |    0 | priority  | default   | 192.168.1.31   | 55555 |     40 | -     | 0:0      |
|               |         |         |      |      |       |      |      |           |           | 192.168.1.32   | 55555 |     60 | -     | 0:0      |
| 192.168.1.12  |         |         |      | 2020 | tcp   |      |    0 | rr        | onearm    | 192.168.1.31   |  8080 |      1 | -     | 0:0      |
|               |         |         |      |      |       |      |      |           |           | 192.168.1.32   |  8080 |      1 | -     | 0:0      |
| 192.168.1.13  |         |         |      | 3030 | tcp   |      |    0 | rr        | fullnat   | 192.168.1.31   |  8080 |      1 | -     | 0:0      |
|               |         |         |      |      |       |      |      |           |           | 192.168.1.32   |  8080 |      1 | -     | 0:0      |
| 192.168.1.15  |         |         |      |   80 | tcp   |      |    0 | hash      | dsr       | 192.168.1.41   |    80 |      1 | -     | 0:0      |
|               |         |         |      |      |       |      |      |           |           | 192.168.1.42   |    80 |      1 | -     | 0:0      |
| 192.168.1.9   |         |         |      |   80 | tcp   |      |    0 | rr        | default   | 192.168.1.31   | 55555 |      1 | -     | 0:0      |
|               |         |         |      |      |       |      |      |           |           | 192.168.1.32   | 55555 |      1 | -     | 0:0      |
root@loxilb:/#
```

create mirror 명령어

loxicmd create mirror 명령어는 loxilb에서 미러링 정책을 설정할 수 있습니다. 아래와 같이 'loxicmd create mirror --help' 명령어는 미러링에 관련한 설정 값들을 볼 수 있습니다.

```
root@loxilb:/# loxicmd create mirror --help
Create a Mirror using LoxiLB
--<infoOption>s of mirrorInfo
type(int) : Mirroring type as like 0 == SPAN, 1 == RSPAN, 2 == ERSPAN
port(string) : The port where mirrored traffic needs to be sent
vlan(int) : for RSPAN we may need to send tagged mirror traffic
remoteIP(string) : For ERSPAN we may need to send tunnelled mirror traffic
sourceIP(string): For ERSPAN we may need to send tunnelled mirror traffic
tunnelID(int): For ERSPAN we may need to send tunnelled mirror traffic

ex) loxicmd create mirror mirr-1 --mirrorInfo="type:0,port:hs0" --targetObject="attachement:1,mirrObjName:hs1"

Usage:
  loxicmd       create       mirror       <mirrorIdent>       --mirrorInfo=<InfoOption>:<InfoValue>,...       --
targetObject=attachement:<port1,rule2>,mirrObjName:<ObjectName> [flags]

Aliases:
  mirror, mirror, mirr, mirrors

Flags:
  -h, --help                  help for mirror
      --mirrorInfo strings    Information about the mirror
      --targetObject strings  Information about object to which mirror needs to be attached

Global Flags:
  -s, --apiserver string   Set API server IP address (default "127.0.0.1")
  -o, --output string      Set output layer (ex.) wide, json)
  -p, --port int16         Set API server port number (default 11111)
      --protocol string    Set API server http/https (default "http")
  -t, --timeout int16      Set timeout (default 10)
      --token string       Set Token for the API server
```

아래(예)와 같이 mirror 명령어로 미러링 설정과 관리가 가능 합니다.

```
root@loxilb:/# loxicmd get mirror
root@loxilb:/# loxicmd create mirror mirr-1 --mirrorInfo="type:0,port:hs0" --targetObject="attachement:1,mirrObjName:hs1"
Debug: response.StatusCode: 200
Success
root@loxilb:/# loxicmd get mirror
| MIRROR NAME | MIRROR INFO |   TARGET   | TARGET |
|             |             | ATTACHMENT |  NAME  |
|-------------|-------------|------------|--------|
| mirr-1      | Type : SPAN | Port       | hs1    |
|             | Port : hs0  |            |        |
root@loxilb:/# loxicmd delete mirror mirr-1
Debug: response.StatusCode: 200
Success
root@loxilb:/# loxicmd get mirror
root@loxilb:/# loxicmd create mirror mirr-1 --mirrorInfo="type:0,port:hs0" --targetObject="attachement:1,mirrObjName:hs1"
Debug: response.StatusCode: 200
Success
root@loxilb:/# loxicmd get mirror -o wide
| MIRROR NAME | MIRROR INFO |   TARGET   | TARGET | SYNC |
|             |             | ATTACHMENT |  NAME  |      |
|-------------|-------------|------------|--------|------|
| mirr-1      | Type : SPAN |          1 | hs1    |    1 |
|             | Port : hs0  |            |        |      |
```

create neighbor 명령어

loxicmd create neighbor 명령어는 loxilb에서 네이버링 정책을 설정할 수 있습니다. 아래와 같이 'loxicmd create neighbor --help' 명령어는 네이버링에 관련한 설정 값들을 볼 수 있습니다.

```
root@loxilb:/# loxicmd create neighbor --help
Create a Neighbors using LoxiLB. It is working as "ip neigh add <DeviceIP> dev <device> lladdr <--macAddress>"

ex) loxicmd create neighbor 192.168.0.1 eno7 --macAddress=aa:aa:aa:aa:aa:aa

Usage:
  loxicmd create neighbor <DeviceIP> <DeviceName> [--macAddress=aa:aa:aa:aa:aa:aa] [flags]

Aliases:
  neighbor, nei, neigh

Flags:
  -h, --help                help for neighbor
      --macAddress string   Hardware MAC address

Global Flags:
  -s, --apiserver string   Set API server IP address (default "127.0.0.1")
  -o, --output string      Set output layer (ex.) wide, json)
  -p, --port int16         Set API server port number (default 11111)
      --protocol string    Set API server http/https (default "http")
  -t, --timeout int16      Set timeout (default 10)
      --token string       Set Token for the API server
```

아래(예)와 같이 neighbor 명령어로 네트워크 이웃 설정과 관리가 가능 합니다.

- DeviceIP: 이웃 장치의 IP 주소
- DeviceName: 네트워크 인터페이스 이름 (예: eno7, eth0 등)
- --macAddress: 해당 IP에 대응하는 하드웨어 MAC 주소

```
root@loxilb:/# loxicmd create neighbor 192.168.1.200 ens34 --macAddress=aa:aa:aa:aa:aa:aa
Success
root@loxilb:/# arp
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.1.32             ether   00:0c:29:b0:1b:49   C                     ens34
192.168.1.20             ether   00:0c:29:62:b8:b6   C                     ens34
_gateway                 ether   00:0c:29:31:04:81   C                     ens34
192.168.1.31             ether   00:0c:29:5d:18:dc   C                     ens34
192.168.1.99             ether   00:0c:29:d4:ec:41   C                     ens34
192.168.1.33             ether   00:0c:29:61:4e:a5   C                     ens34
192.168.1.200            ether   aa:aa:aa:aa:aa:aa   CM                    ens34
root@loxilb:/# arp -a
? (192.168.1.32) at 00:0c:29:b0:1b:49 [ether] on ens34
? (192.168.1.20) at 00:0c:29:62:b8:b6 [ether] on ens34
_gateway (192.168.1.1) at 00:0c:29:31:04:81 [ether] on ens34
? (192.168.1.31) at 00:0c:29:5d:18:dc [ether] on ens34
? (192.168.1.99) at 00:0c:29:d4:ec:41 [ether] on ens34
? (192.168.1.33) at 00:0c:29:61:4e:a5 [ether] on ens34
? (192.168.1.200) at aa:aa:aa:aa:aa:aa [ether] PERM on ens34
root@loxilb:/#
```

create policy 명령어

loxicmd create policy 명령어로 loxilb에서 정책을 설정할 수 있습니다. 아래와 같이 'loxicmd create policy --help' 명령어는 정책에 관련한 설정 값들을 볼 수 있습니다.

```
root@loxilb:/# loxicmd create policy --help
Create a Policy
Ex) loxicmd create policy pol-hs0 --rate=100:100 --target=hs0:1
    loxicmd create policy pol-hs1 --rate=100:100 --target=hs0:1 --block-size=12000:6000
    loxicmd create policy pol-hs1 --rate=100:100 --target=hs0:1 --color
    loxicmd create policy pol-hs1 --rate=100:100 --target=hs0:1 --color --pol-type 0

rate unit : Mbps
block-size unit : bps
Policy type(pol-type) 0 : TrTCM,  1 : SrTCM

Usage:
  loxicmd   create   policy   IDENT   --rate=<Peak>:<Commited>   --target=<ObjectName>:<Attachment>   [--block-
size=<Excess>:<Committed>] [--color] [--pol-type=<policy type>] [flags]

Aliases:
  policy, pol, policys, pols, polices

Flags:
      --block-size string   Block Size pairs can be specified as '<Excess>:<Committed>'
      --color               Policy color enbale or not
  -h, --help                help for policy
      --pol-type int        Target Interface pairs can be specified as '<ObjectName>:<Attachment>'
      --rate string         Rate pairs can be specified as '<Peak>:<Commited>'
      --target string       Target Interface pairs can be specified as '<ObjectName>:<Attachment>'

Global Flags:
  -s, --apiserver string   Set API server IP address (default "127.0.0.1")
  -o, --output string      Set output layer (ex.) wide, json)
  -p, --port int16         Set API server port number (default 11111)
      --protocol string    Set API server http/https (default "http")
  -t, --timeout int16      Set timeout (default 10)
      --token string       Set Token for the API server
```

아래(예)와 같이 policy 명령어로 정책 설정과 관리가 가능 합니다.

- --rate: 트래픽 속도 제한 설정 (단위: Mbps)형식: <Peak>:<Committed> (최대속도:보장속도)
- --target: 정책을 적용할 대상 인터페이스형식: <ObjectName>:<Attachment> (객체명:연결점)선택 매개변수
- --block-size: 버킷 크기 설정 (단위: bps)형식: <Excess>:<Committed> (초과:보장)
- --color: 컬러 마킹 활성화 여부

- --pol-type: 정책 유형 선택 0: TrTCM (Two Rate Three Color Marker) - 기본값1: SrTCM (Single Rate Three Color Marker)

```
root@loxilb:/# loxicmd get policy
root@loxilb:/# loxicmd create policy pol-hs0 --rate=100:100 --target=hs0:1
Debug: response.StatusCode: 200
root@loxilb:/# loxicmd get policy
|  IDENT  | PEAKINFORATE | COMMITTEDINFORATE |
|---------|--------------|-------------------|
| pol-hs0 |          100 |               100 |
root@loxilb:/# loxicmd create policy pol-hs1 —rate=100:100 —target=hs0:1 —block-size=12000:6000
Debug: response.StatusCode: 200
root@loxilb:/# loxicmd get policy
|  IDENT  | PEAKINFORATE | COMMITTEDINFORATE |
|---------|--------------|-------------------|
| pol-hs0 |          100 |               100 |
| pol-hs1 |          100 |               100 |
root@loxilb:/# loxicmd create policy pol-hs1 --rate=100:100 --target=hs0:1 --color
Debug: response.StatusCode: 200
root@loxilb:/# loxicmd get policy
|  IDENT  | PEAKINFORATE | COMMITTEDINFORATE |
|---------|--------------|-------------------|
| pol-hs0 |          100 |               100 |
| pol-hs1 |          100 |               100 |
root@loxilb:/# loxicmd create policy pol-hs1 --rate=100:100 --target=hs0:1 --color --pol-type 0
Debug: response.StatusCode: 200
root@loxilb:/# loxicmd get policy -o wide
|  IDENT  | PEAKINFORATE | COMMITTEDINFORATE | EXCESSBLKSIZE | COMMITTEDBLKSIZE | POLICYTYPE | COLORAWARE | POLOBJNAME | ATTACHMENT |
|---------|--------------|-------------------|---------------|------------------|------------|------------|------------|------------|
| pol-hs0 |          100 |               100 |      60000000 |         30000000 |          0 | false      | hs0        |          1 |
| pol-hs1 |          100 |               100 |      60000000 |         30000000 |          0 | true       | hs0        |          1 |
```

create route 명령어

loxicmd create route 명령어로 loxilb에서 라우팅을 설정할 수 있습니다. 아래와 같이 'loxicmd create route --help' 명령어는 라우팅에 관련한 설정 값들을 볼 수 있습니다.

```
root@loxilb:/# loxicmd create route --help
Create a Route using LoxiLB. It is working as "ip route add <DestinationIPNet> via <gateway> proto <protocol>"

ex) loxicmd create route 192.168.212.0/24 172.17.0.254 --proto=static
    loxicmd create route 192.168.212.0/24 172.17.0.254

Usage:
  loxicmd create route <DestinationIPNet> <gateway> --proto=<protocol> [flags]

Flags:
  -h, --help           help for route
      --proto string   Proto static mode

Global Flags:
  -s, --apiserver string   Set API server IP address (default "127.0.0.1")
  -o, --output string      Set output layer (ex.) wide, json)
  -p, --port int16         Set API server port number (default 11111)
      --protocol string    Set API server http/https (default "http")
  -t, --timeout int16      Set timeout (default 10)
      --token string       Set Token for the API server
```

아래(예)와 같이 route 명령어로 라우팅 설정과 관리가 가능 합니다.

- DestinationIPNet: 목적지 IP 네트워크 (CIDR 표기법)
- gateway: 게이트웨이 IP 주소

```
root@loxilb:/# loxicmd create route 192.168.212.0/24 172.17.0.254 --proto=static
Success
root@loxilb:/# loxicmd get route -o wide
|      DESTINATIONIPNET      |        GATEWAY        |   FLAG   | HARDWAREMARK | PACKETS | BYTES |
|----------------------------|-----------------------|----------|--------------|---------|-------|
| 0.0.0.0/0                  | 192.168.1.1           | Ind      |           21 |       0 |     0 |
| 127.0.0.0/8                |                       | Self     |            1 |       0 |     0 |
| 127.0.0.1/32               |                       | Self     |            2 |       0 |     0 |
| 172.17.0.0/16              |                       | Self     |           18 |       0 |     0 |
| 172.17.0.1/32              |                       | Self     |           19 |       0 |     0 |
| 172.17.0.254/32            | 172.17.0.254          | Ind Host |           30 |       0 |     0 |
| 192.168.1.1/32             | 192.168.1.1           | Ind Host |            8 |       0 |     0 |
| 192.168.1.20/32            | 192.168.1.20          | Ind Host |           23 |       0 |     0 |
| 192.168.1.200/32           | 192.168.1.200         | Ind Host |           28 |       0 |     0 |
| 192.168.1.31/32            | 192.168.1.31          | Ind Host |           10 |       0 |     0 |
| 192.168.1.32/32            | 192.168.1.32          | Ind Host |           12 |       0 |     0 |
| 192.168.1.33/32            | 192.168.1.33          | Ind Host |            9 |       0 |     0 |
| 192.168.1.9/32             |                       | Self     |            5 |       0 |     0 |
| 192.168.1.99/32            | 192.168.1.99          | Ind Host |           11 |       0 |     0 |
| 192.168.212.0/24           | 172.17.0.254          | Ind      |           29 |       0 |     0 |
| 224.0.0.22/32              | 224.0.0.22            | Ind Host |           20 |       0 |     0 |
| ::/0                       | fe80::20c:29ff:fe31:481 | Ind      |           22 |       0 |     0 |
| ::1/128                    |                       | Self     |            3 |       0 |     0 |
| fe80::/32                  |                       | Self     |            7 |       0 |     0 |
| fe80::/64                  |                       | Self     |            6 |       0 |     0 |
| fe80::20c:29ff:fe31:481/128 | fe80::20c:29ff:fe31:481 | Ind Host |           16 |       0 |     0 |
| ff02::16/128               | ff02::16              | Ind Host |           17 |       0 |     0 |
| ff02::1:2/128              | ff02::1:2             | Ind Host |           14 |       0 |     0 |
| ff02::1:ff3e:2bb2/128      | ff02::1:ff3e:2bb2     | Ind Host |           13 |       0 |     0 |
| ff02::2/128                | ff02::2               | Ind Host |           15 |       0 |     0 |
```

create vlan 명령어

loxicmd create vlan 명령어로 loxilb에서 VLAN을 설정할 수 있습니다. 아래와 같이 'loxicmd create vlan --help' 명령어는 VLAN에 관련한 설정 값들을 볼 수 있습니다.

```
root@loxilb:/# loxicmd create vlan --help
Create a vlan using LoxiLB. It is working as "brctl addbr vlan$<Vid>"

ex) loxicmd create vlan 100

Usage:
  loxicmd create vlan <Vid> [flags]

Flags:
  -h, --help   help for vlan

Global Flags:
  -s, --apiserver string   Set API server IP address (default "127.0.0.1")
  -o, --output string      Set output layer (ex.) wide, json)
  -p, --port int16         Set API server port number (default 11111)
      --protocol string    Set API server http/https (default "http")
  -t, --timeout int16      Set timeout (default 10)
      --token string       Set Token for the API server
```

create vlanmember 명령어

loxicmd create vlanmember 명령어로 loxilb에서 VLAN 멤버를 설정할 수 있습니다. 아래와 같이 'loxicmd create vlanmember --help' 명령어는 VLAN 멤버에 관련한 설정 값들을 볼 수 있습니다.

```
root@loxilb:/# loxicmd create vlanmember --help
Create a vlanmember using LoxiLB. It is working as "brctl addif vlan<Vid> <DeviceName>.<tagged>"

ex) loxicmd create vlanmember 100 eno7 --tagged=true
    loxicmd create vlanmember 100 eno7

Usage:
  loxicmd create vlanmember <Vid> <DeviceName> --tagged=<Tagged> [flags]

Aliases:
  vlanmember, vlanMember, vlan-member, vlan_member

Flags:
  -h, --help     help for vlanmember
      --tagged   Tagged mode Vlan

Global Flags:
  -s, --apiserver string   Set API server IP address (default "127.0.0.1")
  -o, --output string      Set output layer (ex.) wide, json)
  -p, --port int16         Set API server port number (default 11111)
      --protocol string    Set API server http/https (default "http")
  -t, --timeout int16      Set timeout (default 10)
      --token string       Set Token for the API server
```

아래(예)와 같이 vlan과 vlanmember 명령어로 가상랜 설정과 관리가 가능 합니다.

```
root@loxilb:/# loxicmd create vlanmember 100 --tagged=true
root@loxilb:/# loxicmd get vlan -o wide
| DEVICE NAME | VLAN ID |      MEMBER       |      STATISTICS      |
|-------------|---------|-------------------|----------------------|
| vlan100     |     100 | Device: ens34.100 | In/Out byte : 0/0    |
|             |         | tagged: true      | In/Out packets : 0/0 |
| vlan3802    |    3802 |                   | In/Out byte : 0/0    |
|             |         |                   | In/Out packets : 0/0 |
| vlan3804    |    3804 |                   | In/Out byte : 0/0    |
|             |         |                   | In/Out packets : 0/0 |
| vlan3808    |    3808 |                   | In/Out byte : 0/0    |
|             |         |                   | In/Out packets : 0/0 |
| docker0     |    4090 |                   | In/Out byte : 0/0    |
|             |         |                   | In/Out packets : 0/0 |
root@loxilb:/# loxicmd create vlan 200
Success
root@loxilb:/# loxicmd get vlan -o wide
| DEVICE NAME | VLAN ID |      MEMBER       |      STATISTICS      |
|-------------|---------|-------------------|----------------------|
| vlan100     |     100 | Device: ens34.100 | In/Out byte : 0/0    |
|             |         | tagged: true      | In/Out packets : 0/0 |
| vlan200     |     200 |                   | In/Out byte : 0/0    |
|             |         |                   | In/Out packets : 0/0 |
| vlan3802    |    3802 |                   | In/Out byte : 0/0    |
|             |         |                   | In/Out packets : 0/0 |
| vlan3804    |    3804 |                   | In/Out byte : 0/0    |
|             |         |                   | In/Out packets : 0/0 |
| vlan3808    |    3808 |                   | In/Out byte : 0/0    |
|             |         |                   | In/Out packets : 0/0 |
| docker0     |    4090 |                   | In/Out byte : 0/0    |
|             |         |                   | In/Out packets : 0/0 |
```

create vxlan 명령어

loxicmd create vxlan 명령어로 loxilb에서 VxLAN을 설정할 수 있습니다. 아래와 같이 'loxicmd create vxlan --help' 명령어는 VxLAN에 관련한 설정 값들을 볼 수 있습니다.

```
root@loxilb:/# loxicmd create vxlan --help
Create a vxlan using LoxiLB.

ex) loxicmd create vxlan 100 eno7

Usage:
  loxicmd create vxlan <VxlanID> <EndpointDeviceName> [flags]

Flags:
  -h, --help   help for vxlan

Global Flags:
  -s, --apiserver string   Set API server IP address (default "127.0.0.1")
  -o, --output string      Set output layer (ex.) wide, json)
  -p, --port int16         Set API server port number (default 11111)
      --protocol string    Set API server http/https (default "http")
  -t, --timeout int16      Set timeout (default 10)
      --token string       Set Token for the API server
```

create vxlanpeer 명령어

loxicmd create vxlanpeer 명령어로 loxilb에서 VxLAN 피어를 설정할 수 있습니다. 아래와 같이 'loxicmd create vxlanpeer --help' 명령어는 VxLAN 피어에 관련한 설정 값들을 볼 수 있습니다.

```
root@loxilb:/# loxicmd create vxlanpeer --help
Create a vxlan using LoxiLB.

ex) loxicmd create vxlan-peer 100 30.1.3.1

Usage:
  loxicmd create vxlanpeer <Vnid> <PeerIP> [flags]

Aliases:
  vxlanpeer, vxlanPeer, vxlan-peer, vxlan_peer

Flags:
  -h, --help   help for vxlanpeer

Global Flags:
  -s, --apiserver string   Set API server IP address (default "127.0.0.1")
  -o, --output string      Set output layer (ex.) wide, json)
  -p, --port int16         Set API server port number (default 11111)
      --protocol string    Set API server http/https (default "http")
  -t, --timeout int16      Set timeout (default 10)
      --token string       Set Token for the API server
```

아래(예)와 같이 vxlan과 vxlanpeer 명령어로 VxLAN 설정과 관리가 가능 합니다.

```
root@loxilb:/# loxicmd create vxlan 100 ens34
Success
root@loxilb:/# loxicmd get vxlan -o wide
| DEVICE NAME | VXLAN ID | ENDPOINT INTERFACE | PEER IP |
|-------------|----------|--------------------|---------|
| vxlan100    |      100 | ens34              |         |
root@loxilb:/# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: ens34: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:3e:2b:b2 brd ff:ff:ff:ff:ff:ff
    altname enp2s2
    inet 192.168.1.9/24 brd 192.168.1.255 scope global ens34
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fe3e:2bb2/64 scope link
       valid_lft forever preferred_lft forever
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 02:42:cf:da:1b:8e brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
       valid_lft forever preferred_lft forever
4: llb0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 xdpgeneric/id:38 qdisc fq_codel state UNKNOWN group default
qlen 1000
    link/ether 00:00:ca:fe:fa:ce brd ff:ff:ff:ff:ff:ff
    inet6 fe80::58e5:2ff:fe38:17ec/64 scope link
       valid_lft forever preferred_lft forever
5: vxlan100: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc noqueue state UNKNOWN group default
    link/ether ce:e2:3e:c7:0e:02 brd ff:ff:ff:ff:ff:ff
    inet6 fe80::cce2:3eff:fec7:e02/64 scope link
       valid_lft forever preferred_lft forever

root@loxilb:/# loxicmd create vxlan-peer 100 30.1.3.1
Success
root@loxilb:/# loxicmd get vxlan -o wide
| DEVICE NAME | VXLAN ID | ENDPOINT INTERFACE | PEER IP  |
|-------------|----------|--------------------|----------|
| vxlan100    |      100 | ens34              | 30.1.3.1 |
root@loxilb:/# ip -d link show type vxlan
5: vxlan100: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc noqueue state UNKNOWN mode DEFAULT group default
    link/ether ce:e2:3e:c7:0e:02 brd ff:ff:ff:ff:ff:ff promiscuity 0 minmtu 68 maxmtu 65535
    vxlan id 100 local 192.168.1.9 dev ens34 srcport 0 0 dstport 8472 ttl auto ageing 300 udpcsum noudp6zerocsumtx
noudp6zerocsumrx addrgenmode eui64 numtxqueues 1 numrxqueues 1 gso_max_size 65536 gso_max_segs 65535
```

delete 명령어

delete 명령어는 생성한 정책 등을 삭제할 수 있습니다. 아래와 같이 'loxicmd delete --help' 명령어는 삭제에 관련한 명령어들을 볼 수 있습니다.

```
root@loxilb:/# loxicmd delete --help
Delete a Load balance features in the LoxiLB.
Delete - Service type external load-balancer, Vlan, Vxlan, Qos Policies,
         Endpoint client,FDB, IPaddress, Neighbor, Route,Firewall, Mirror, Session, UICI

Usage:
  loxicmd delete [flags]
  loxicmd delete [command]

Available Commands:
  bfd         Delete a BFD session
  bgpneighbor Delete a BGP Neighbor peer information
  endpoint    Delete a LB EndPoint from monitoring
  fdb         Delete a FDB
  firewall    Delete a Firewall
  ip          Delete a IPv4Address
  lb          Delete a LoadBalancer
  mirror      Delete a Mirror
  neighbor    Delete a Neighbors
  policy      Delete a Policy
  route       Delete a Route
  session     Delete a Session
  sessionulcl Delete a Ulcl configuration in the LoxiLB.
  vlan        Delete a VlanBridge
  vlanmember  Delete a VlanMember
  vxlan       Delete a vxlanBridge
  vxlanpeer   Delete a vxlanPeer

Flags:
  -f, --file string   Config file to apply as like K8s
  -h, --help          help for delete

Global Flags:
  -s, --apiserver string   Set API server IP address (default "127.0.0.1")
  -o, --output string      Set output layer (ex.) wide, json)
  -p, --port int16         Set API server port number (default 11111)
      --protocol string    Set API server http/https (default "http")
  -t, --timeout int16      Set timeout (default 10)
      --token string       Set Token for the API server

Use "loxicmd delete [command] --help" for more information about a command.
```

get 명령어

get 명령어는 생성한 정책 등을 생성 후에 상태를 모니터링 할 수 있습니다. 아래와 같이 'loxicmd get --help' 명령어는 모니터링에 관련한 명령어들을 볼 수 있습니다.

```
root@loxilb:/# loxicmd get --help
Get a Load balance features from LoxiLB.
        Get - Service type external load-balancer, Vlan, Vxlan, Qos Policies,
         Endpoint client,FDB, IPaddress, Neighbor, Route,Firewall, Mirror, Session, UICI
        Get Port(interface) dump used by loxilb or its docker
        Get Connection track (TCP/UDP/ICMP/SCTP) information

Usage:
  loxicmd get [flags]
  loxicmd get [command]

Available Commands:
  bfd          Get all BFD sessions
  conntrack    Get a Conntrack
  endpoint     Get endpoints
  fdb          Get a fdb
  firewall     Get a firewall
  hastate      Get a HA state
  ip           Get a IP Address
  lbversion    Get a loxilb version
  loadbalancer Get a LoadBalancer
  log-level    Get a log level status
  mirror       Get a Mirror
  neighbor     Get a neighbors
  policy       Get a Policy
  port         Get a Port dump
  process      Get a process status
  route        Get a route
  session      Get a session
  vlan         Get a Vlan
  vxlan        Get a vxlan

Flags:
  -h, --help   help for get

Global Flags:
  -s, --apiserver string   Set API server IP address (default "127.0.0.1")
  -o, --output string      Set output layer (ex.) wide, json)
  -p, --port int16         Set API server port number (default 11111)
      --protocol string    Set API server http/https (default "http")
  -t, --timeout int16      Set timeout (default 10)
      --token string       Set Token for the API server

Use "loxicmd get [command] --help" for more information about a command.
```

help 명령어

help 명령어는 아래와 같이 'loxicmd help --help' 명령어로 관련한 설명을 볼 수 있습니다.

```
root@loxilb:/# loxicmd help --help
Help provides help for any command in the application.
Simply type loxicmd help [path to command] for full details.

Usage:
  loxicmd help [command] [flags]

Flags:
  -h, --help   help for help

Global Flags:
  -s, --apiserver string   Set API server IP address (default "127.0.0.1")
  -o, --output string      Set output layer (ex.) wide, json)
  -p, --port int16         Set API server port number (default 11111)
      --protocol string    Set API server http/https (default "http")
  -t, --timeout int16      Set timeout (default 10)
```

save 명령어

save 명령어로 'loxilb' 설정을 저장할 수 있으며 아래와 같이 'loxicmd save --help' 명령어로 관련한 설명을 볼 수 있습니다.

```
root root@loxilb:/# loxicmd save --help
saves current configuration in text file

Usage:
  loxicmd save [flags]

Flags:
  -a, --all                 Saves all loxilb configuration
      --bfd                 Saves BFD configuration
  -c, --config-path string  config file patch setting
      --endpoint            Saves endpoint configuration
      --firewall            Saves firewall configuration
  -h, --help                help for save
  -i, --ip                  Saves IP configuration
  -l, --lb                  Saves Load Balancer rules configuration
      --session             Saves session configuration
      --ulcl                Saves ulcl configuration

Global Flags:
  -s, --apiserver string   Set API server IP address (default "127.0.0.1")
  -o, --output string      Set output layer (ex.) wide, json)
  -p, --port int16         Set API server port number (default 11111)
      --protocol string    Set API server http/https (default "http")
  -t, --timeout int16      Set timeout (default 10)
      --token string       Set Token for the API server
```

set 명령어

set 명령어는 아래와 같이 'loxicmd set --help' 명령어로 관련한 설명을 볼 수 있습니다.

```
root@loxilb:/# loxicmd set --help
Set the configuration like log-level or bfd session

Usage:
  loxicmd set [flags]
  loxicmd set [command]

Available Commands:
  bfd         bfd session configuration
  log-level   log-level configuration
  login       login and set token
  logout      logout and remove token
  refresh     refresh token

Flags:
  -h, --help   help for set

Global Flags:
  -s, --apiserver string   Set API server IP address (default "127.0.0.1")
  -o, --output string      Set output layer (ex.) wide, json)
  -p, --port int16         Set API server port number (default 11111)
      --protocol string    Set API server http/https (default "http")
  -t, --timeout int16      Set timeout (default 10)
      --token string       Set Token for the API server

Use "loxicmd set [command] --help" for more information about a command.
```

version 명령어

version 명령어로 엔터프라이즈용 loxilb 버전을 확인할 수 있으며 아래와 같이 'loxicmd version --help' 명령어로 관련한 설명을 볼 수 있습니다.

```
root@loxilb:/# loxicmd version --help
It shows Loxicmd version.

Usage:
  loxicmd version [flags]

Flags:
  -h, --help   help for version

Global Flags:
  -s, --apiserver string   Set API server IP address (default "127.0.0.1")
  -o, --output string      Set output layer (ex.) wide, json)
  -p, --port int16         Set API server port number (default 11111)
      --protocol string    Set API server http/https (default "http")
  -t, --timeout int16      Set timeout (default 10)
      --token string       Set Token for the API server
```

SUMMARY

loxilb는 가상화나 K8s등의 external 모드로 클러스터 외부 노드에서 독립적으로 실행합니다.

CHAPTER

04

HA(High-Availability)

HA(High-Availability)

LoxiLB Enterprise를 고가용성(HA) 모드로 배포하는 방법을 설명합니다. 사용자는 NAT 모드에 대해 기본적인 지식을 기반으로 loxilb에서 지원하는 HA 기능을 설정할 수 있습니다.

1. 'loxilb' 설치

K8s 클러스터 외부에 2대의 'loxilb'를 아래의 방법으로 설치를 합니다.

'loxilb' 컨테이너 설치

- 'loxilb' Docker 이미지 다운로드 (**LoxiLB Enterprise 솔루션을 위한 Docker 이미지 별도 제공 예정):

```
sudo docker load -i loxilb-enterprise-1.0.0.tar
```

- 도커 이미지 이름 변경후 'loxilb'컨테이너 실행하며 BFD 는 --ka option 적용 (1 초 이내 복구를 위한 BFD 적용 참고)

```
sudo docker run -u root --cap-add SYS_ADMIN --restart unless-stopped --privileged --entrypoint "/root/loxilb-io/loxilb/loxilb" -dit -v /dev/log:/dev/log --net=host --name loxilb ghcr.io/loxilb-io/loxilb:latest
```

- 'loxicmd' 사용을 위한 'loxilb' 터미널 연결

```
sudo docker exec -it loxilb bash
```

- 'loxilb'가 제공하는 'loxicmd' 명령어를 통한 수동 구성과 관리

```
### loxicmd 명령어를 통해 LB, Endpoint, Policy 등의 설정(예):

loxicmd create lb 192.168.1.9 --tcp=80:8080 --endpoints=10.0.0.1:1
loxicmd create lb 192.168.1.9 --tcp=80:30458 --endpoints=192.168.1.31:1,192.168.1.32:1 --monitor
```

'loxilb' 컨테이너 실행 후 상태와 제조사 요구 시 필요한 로그 확인 명령어 제공:

```
docker ps | grep loxilb
docker logs loxilb
```

```
[loxilb@localhost ~]$ docker ps | grep loxilb
392ad81f856f   ghcr.io/loxilb-io/loxilb:latest   "/root/loxilb-io/lox..."   2 weeks ago   Up 4 minutes                 loxilb
[loxilb@localhost ~]$ docker logs loxilb
loxilb start
05:46:49 DEBUG loxilb_libdp.c:3506: /opt/loxilb/llb_xdp_main.o: nr 0 psection xdp_packet_hook
05:46:58 DEBUG loxilb_libdp.c:3272: llb_link_prop_add: IF-llb0 added idx 0 type 1
05:46:58 DEBUG loxilb_libdp.c:3525: setting up xdp for llb0|xdp_packet_hook
05:46:58 DEBUG loxilb_libdp.c:3347: llb_psec_add: SEC-tc_packet_hook0 added idx 0
05:46:58 DEBUG loxilb_libdp.c:3506: /opt/loxilb/llb_ebpf_main.o: nr 0 psection tc_packet_hook0
05:46:58 INFO  common_libbpf.c:121: tc: bpf attach start for llb0:0
05:46:58 ERROR common_libbpf.c:141: tc: no obj for pinpath /opt/loxilb/dp/bpf/llb_ebpf.rodata
05:46:58 DEBUG common_libbpf.c:161: tc: autoload sec tc_packet_hook1 prog tc_packet_func
05:46:58 DEBUG common_libbpf.c:161: tc: autoload sec tc_packet_hook2 prog tc_packet_func_slow
05:46:58 DEBUG common_libbpf.c:161: tc: autoload sec tc_packet_hook3 prog tc_packet_func_fw
05:46:58 DEBUG common_libbpf.c:161: tc: autoload sec tc_packet_hook4 prog tc_csum_func1
05:46:58 DEBUG common_libbpf.c:161: tc: autoload sec tc_packet_hook5 prog tc_csum_func2
05:46:58 DEBUG common_libbpf.c:161: tc: autoload sec tc_packet_hook6 prog tc_slow_unp_func
05:46:58 DEBUG common_libbpf.c:161: tc: autoload sec tc_packet_hook7 prog tc_packet_func_masq
05:46:59 INFO  common_libbpf.c:204: tc: bpf attach OK for llb0
05:46:59 DEBUG loxilb_libdp.c:3248: llb_link_prop_add: IF-llb0 ref idx 0:1 type 2
2025-04-22 05:46:59 ebpf unload - lo
libbpf: Kernel error message: Cannot find specified qdisc on specified device
2025-04-22 05:46:59 ebpf unload - docker0
libbpf: Kernel error message: Cannot find specified qdisc on specified device
05:46:59 ERROR common_libbpf.c:94: tc: bpf hook destroy failed for docker0:0
2025/04/22 05:47:00 Serving loxilb rest API at http://[::]:11111
2025-04-22 05:47:02 nlp: Link msgs subscribed
2025-04-22 05:47:02 nlp: Addr msgs subscribed
2025-04-22 05:47:02 nlp: Neigh msgs subscribed
2025-04-22 05:47:02 nlp: Route msgs subscribed
2025-04-22 05:47:02 nlp: NLP Subscription done
2025-04-22 05:47:02 nlp: Getting device info
2025-04-22 05:47:02 nlp: ADD dev docker0 mac([46 234 67 17 110 75]) attrs(&{3 1500 0 docker0 2e:ea:43:11:6e:4b up|broadcast|multica
st 4099 0 0 <nil>  0xc00274cf00 0 0 1 0xc0049c76c8 ether <nil> down 0 -1 1 1 65536 65535 [] 0 <nil>}) - info recvd
2025-04-22 05:47:02 port add - docker0 isPvid false
2025-04-22 05:47:02 port added - docker0:1 OSID 3
2025-04-22 05:47:02 vlan 4090 bd created
2025-04-22 05:47:02 nlp: Bridge docker0, 4090, [46 234 67 17 110 75], false, 1500 ADD [OK]
2025-04-22 05:47:02 nlp: ADD dev ens160 mac([0 12 41 223 240 137]) attrs(&{2 1500 1000 ens160 00:0c:29:df:f0:89 up|broadcast|multic
ast 69699 0 0 <nil>  0xc00274d440 0 0 1 0xc0049c78c0 ether <nil> up 0 -1 2 2 65536 65535 [] 0 <nil>}) - info recvd
2025-04-22 05:47:02 port add - vlan3802 isPvid false
2025-04-22 05:47:02 port added - vlan3802:3 OSID -1
2025-04-22 05:47:02 vlan 3802 bd created
2025-04-22 05:47:02 port added - ens160:2 OSID 2
2025-04-22 05:47:02 nlp: Port ens160, [0 12 41 223 240 137], true, 1500 add [OK]
2025-04-22 05:47:02 nlp: ADD dev docker0 mac([46 234 67 17 110 75]) attrs(&{3 1500 0 docker0 2e:ea:43:11:6e:4b up|broadcast|multica
st 4099 0 0 <nil>  0xc00274d5c0 0 0 1 0xc0049c78d8 ether <nil> down 0 -1 1 1 65536 65535 [] 0 <nil>}) - info recvd
2025-04-22 05:47:02 nlp: Bridge docker0, 4090, [46 234 67 17 110 75], false, 1500 ADD [OK]
2025-04-22 05:47:02 nlp: ADD dev llb0 mac([0 0 202 254 250 206]) attrs(&{4 1500 1000 llb0 00:00:ca:fe:fa:ce up|broadcast|multicast 696
99 0 0 <nil>  0xc00274d740 0 0 1 0xc0049c78f0 ether <nil> unknown 0 -1 1 1 65536 65535 [] 0 <nil>}) - info recvd
2025-04-22 05:47:02 port add - vlan3804 isPvid false
2025-04-22 05:47:02 port added - vlan3804:5 OSID -1
2025-04-22 05:47:02 vlan 3804 bd created
2025-04-22 05:47:02 port added - llb0:4 OSID 4
2025-04-22 05:47:02 nlp: Port llb0, [0 0 202 254 250 206], true, 1500 add [OK]
```

```
2025-04-22 05:47:02 rt added - 127.0.0.0/8:root mark:
2025-04-22 05:47:02 rt added - 127.0.0.1/32:root mark:
2025-04-22 05:47:02 ifa added 127.0.0.1:lo
2025-04-22 05:47:02 nlp: IPv4 Address 127.0.0.1/8 Port lo added
2025-04-22 05:47:02 rt added - ::1/128:root mark:
2025-04-22 05:47:02 ifa added ::1:lo
2025-04-22 05:47:02 nlp: IPv4 Address ::1/128 Port lo added
2025-04-22 05:47:02 rt added - 192.168.191.0/24:root mark:
2025-04-22 05:47:02 rt added - 192.168.191.137/32:root mark:
2025-04-22 05:47:02 ifa added 192.168.191.137:ens160
2025-04-22 05:47:02 nlp: IPv4 Address 192.168.191.137/24 Port ens160 added
2025-04-22 05:47:02 rt added - fe80::/64:root mark:
2025-04-22 05:47:02 rt added - fe80::/32:root mark:
2025-04-22 05:47:02 ifa added fe80::20c:29ff:fedf:f089:ens160
2025-04-22 05:47:02 nlp: IPv4 Address fe80::20c:29ff:fedf:f089/64 Port ens160 added
2025-04-22 05:47:02 neigh rtpair - 192.168.191.1/32->192.168.191.1
2025-04-22 05:47:02 rt added - 192.168.191.1/32:root mark:1
2025-04-22 05:47:02 added fdb ent, {[0 80 86 192 0 8] 3802} : health(true)
```

2. **BFD**(Bi-directional Forwarding Protocol) **적용 HA 구성 'loxilb' 설치**

'LoxiLB Enterprise'의 BFD(Birectional Forwarding Detection) 기능은 고가용성(HA) 환경에서 1초 이내의 빠른 장애 감지와 자동화한 장애 조치를 지원하여 서비스의 연속성을 보장하는 역할을 합니다.

'LoxiLB Enterprise'는 BFD 지원을 위해 2 대의 'loxilb' 기기(예: Active 'loxilb', Backup 'loxilb')를 구성하여 기기 간에 BFD 세션을 설정하고 서로의 상태를 실시간으로 모니터링하며 장애를 신속하게 감지하고, 장애 발생 시 자동으로 새로운 마스터 'loxilb'를 선출하여 서비스 중단을 최소화합니다. ('loxilb' 실행 시 --cluster, --self, --ka 추가 옵션 사용)

아래는 Master 'loxilb'와 Backup 'loxilb'의 실행 예시입니다.

- In first VM(or Bare metal), run Master 'loxilb' as:

```
docker run -u root --cap-add SYS_ADMIN --restart unless-stopped --privileged -dit -v /dev/log:/dev/log --net=host --name bfd ghcr.io/loxilb-io/loxilb:latest --cluster=192.168.1.91 --self=0 --ka=192.168.1.91:192.168.1.81
```

- In second VM(or Bare metal), run Backup 'loxilb' as:

```
docker run -u root --cap-add SYS_ADMIN --restart unless-stopped --privileged -dit -v /dev/log:/dev/log --net=host --name bfd ghcr.io/loxilb-io/loxilb:latest --cluster=192.168.1.81 --self=1 --ka=192.168.1.81:192.168.1.91
```

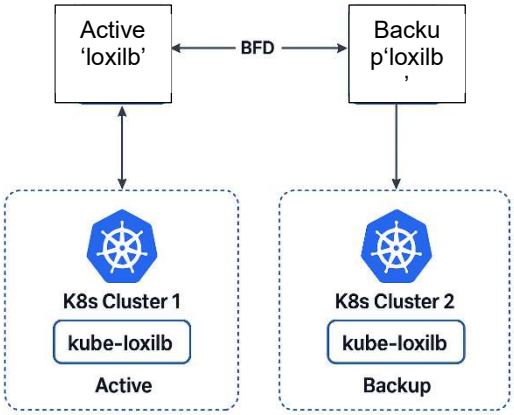

그림 7. HA의 BFD 사용 구성

3. HA 서비스 확인

HA 동작 확인을 위한 구성(예)는 아래와 같이 'Master(192.168.1.70)와 Worker 노드 2개(192.168.1.71, 192.168.1.72) 구성으로 'loxilb'(192.168.1.75)연결하여 확인합니다.

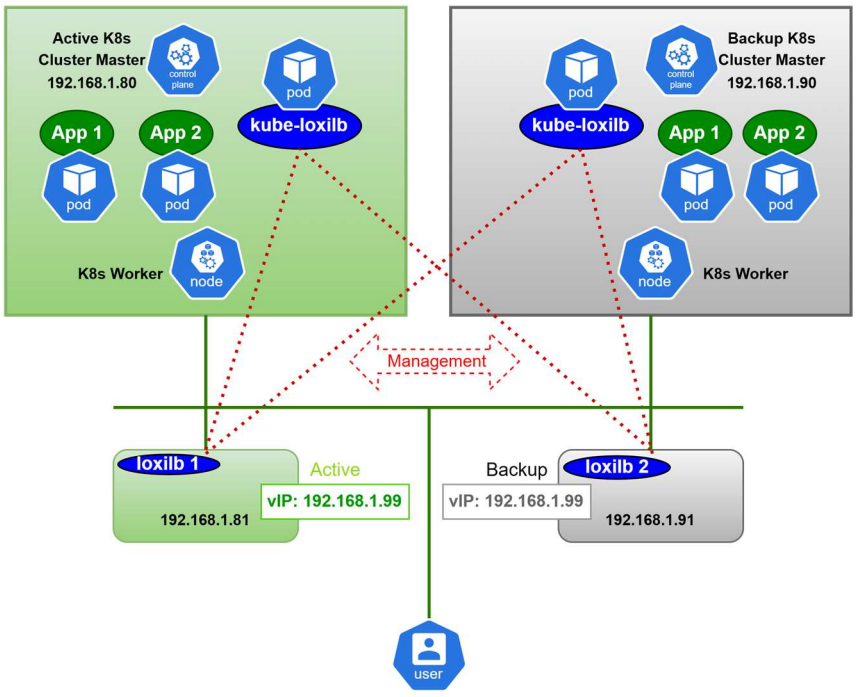

그림 8. 'HA' 기능 확인을 위한 K8s 구성 (가상 IP 주소 vIP를 지정합니다.)

아래는 BFD를적용하여 2개의 서비스 (TCP Port: 55001/55002)를 실행한 Active K8s Master 'loxilb'의 상태입니다. 리눅스 명령어 'ip a'로 가상IP(vIP) 192.168.1.99가 적용된 것을 확인할 수 있습니다. (서비스 실행 시 확인을 위하여 'loxicmd get ct -o wide' 명령으로 vIP 커넥션의 카운트를 확인합니다.)

```
root@activeloxilb:/# loxicmd get lb -o wide
|    EXT IP    | SEC IPS | SOURCES | HOST | PORT  | PROTO |             NAME             | MARK | SEL |  MODE  |   ENDPOINT   | EPORT | WEIGHT | STATE  | COUNTERS |
|--------------|---------|---------|------|-------|-------|------------------------------|------|-----|--------|--------------|-------|--------|--------|----------|
| 192.168.1.99 |         |         |      | 55001 | tcp   | default_web-service1:llb-inst0 |    0 | rr  | onearm | 192.168.1.80 | 30523 |      1 | -      | 0:0      |
| 192.168.1.99 |         |         |      | 55002 | tcp   | default_web-service2:llb-inst0 |    0 | rr  | onearm | 192.168.1.90 | 31088 |      1 | active | 0:0      |
root@activeloxilb:/# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet 192.168.1.99/32 scope global lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:3e:14:72 brd ff:ff:ff:ff:ff:ff
    altname enp2s1
    inet 192.168.1.81/24 brd 192.168.1.255 scope global ens33
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fe3e:1472/64 scope link
       valid_lft forever preferred_lft forever
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 02:42:47:66:cd:4f brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
       valid_lft forever preferred_lft forever
4: llb0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 xdpgeneric/id:296 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:00:ca:fe:fa:ce brd ff:ff:ff:ff:ff:ff
    inet6 fe80::6035:ff:fe0c:d329/64 scope link
       valid_lft forever preferred_lft forever
root@activeloxilb:/# loxicmd get ct -o wide
|          SERVICE NAME          |    DESTIP    |     SRCIP     | DPORT | SPORT | PROTO | IDENT | STATE |                    ACT                    | PACKETS | BYTES |
|--------------------------------|--------------|---------------|-------|-------|-------|-------|-------|-------------------------------------------|---------|-------|
| default_web-service1:llb-inst0 | 192.168.1.81 | 192.168.1.80  | 55873 | 30523 | tcp   | 0:0   | est   | fsnat-192.168.1.99,192.168.1.103:55001:w0 |       1 |    52 |
| default_web-service1:llb-inst0 | 192.168.1.81 | 192.168.1.80  | 55874 | 30523 | tcp   | 0:0   | est   | fsnat-192.168.1.99,192.168.1.103:55001:w0 |       1 |    52 |
| default_web-service1:llb-inst0 | 192.168.1.81 | 192.168.1.80  | 55875 | 30523 | tcp   | 0:0   | est   | fsnat-192.168.1.99,192.168.1.103:55001:w0 |       3 |   346 |
| default_web-service1:llb-inst0 | 192.168.1.99 | 192.168.1.103 | 55001 | 55873 | tcp   | 0:0   | est   | fdnat-192.168.1.81,192.168.1.80:30523:w0  |       2 |    92 |
| default_web-service1:llb-inst0 | 192.168.1.99 | 192.168.1.103 | 55001 | 55874 | tcp   | 0:0   | est   | fdnat-192.168.1.81,192.168.1.80:30523:w0  |       2 |    92 |
| default_web-service1:llb-inst0 | 192.168.1.99 | 192.168.1.103 | 55001 | 55875 | tcp   | 0:0   | est   | fdnat-192.168.1.81,192.168.1.80:30523:w0  |       4 |   769 |
root@activeloxilb:/# loxicmd get bfd -o wide
| INSTANCE  |  REMOTEIP   |  SOURCEIP   | PORT | INTERVAL  | RETRY COUNT | STATE |
|-----------|-------------|-------------|------|-----------|-------------|-------|
| llb-inst0 | 192.168.1.91 | 192.168.1.81 | 3784 | 100000 us |           3 | BFDUp |
root@activeloxilb:/# loxicmd get ha -o wide
| INSTANCE  | HASTATE |
|-----------|---------|
| llb-inst0 | MASTER  |
```

아래는 BFD를적용하여 서비스를 실행한 Backup용 K8s Master 'loxilb'의 상태입니다. (BFD를 사용하며 HA는 Backup 상태이고, vIP는 할당하지 않았습니다.)

```
root@backuploxilb:/# loxicmd get lb -o wide
|    EXT IP    | SEC IPS | SOURCES | HOST | PORT  | PROTO |             NAME             | MARK | SEL |  MODE  |   ENDPOINT   | EPORT | WEIGHT | STATE  | COUNTERS |
|--------------|---------|---------|------|-------|-------|------------------------------|------|-----|--------|--------------|-------|--------|--------|----------|
| 192.168.1.99 |         |         |      | 55001 | tcp   | default_web-service1:llb-inst0 |    0 | rr  | onearm | 192.168.1.80 | 30523 |      1 | -      | 0:0      |
|              |         |         |      |       |       |                              |      |     |        | 192.168.1.90 | 30256 |      1 | -      | 0:0      |
| 192.168.1.99 |         |         |      | 55002 | tcp   | default_web-service2:llb-inst0 |    0 | rr  | onearm | 192.168.1.80 | 31776 |      1 | active | 0:0      |
|              |         |         |      |       |       |                              |      |     |        | 192.168.1.90 | 31088 |      1 | active | 0:0      |
root@backuploxilb:/# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:d1:05:ef brd ff:ff:ff:ff:ff:ff
    altname enp2s1
    inet 192.168.1.91/24 brd 192.168.1.255 scope global ens33
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fed1:5ef/64 scope link
       valid_lft forever preferred_lft forever
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 02:42:d6:87:5c:e8 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
       valid_lft forever preferred_lft forever
4: llb0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 xdpgeneric/id:270 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:00:ca:fe:fa:ce brd ff:ff:ff:ff:ff:ff
    inet6 fe80::a478:d5ff:fef7:b35f/64 scope link
       valid_lft forever preferred_lft forever
root@backuploxilb:/# loxicmd get ct -o wide
|          SERVICE NAME          |    DESTIP    |     SRCIP     | DPORT | SPORT | PROTO | IDENT |  STATE  |                    ACT                    | PACKETS | BYTES |
|--------------------------------|--------------|---------------|-------|-------|-------|-------|---------|-------------------------------------------|---------|-------|
```

```
| -                           | 192.168.1.81 | 192.168.1.91  | 43991 |  3784 | udp  | 0:0  | udp-est |                                                  |      0 |     0 |
| default_web-service2:llb-inst0 | 192.168.1.91 | 192.168.1.80  | 57266 | 31776 | tcp  | 0:0  | est     | fsnat-192.168.1.99,192.168.1.103:55002:w0 |      1 |    52 |
| default_web-service2:llb-inst0 | 192.168.1.91 | 192.168.1.80  | 57267 | 31776 | tcp  | 0:0  | est     | fsnat-192.168.1.99,192.168.1.103:55002:w0 |      1 |    52 |
| -                           | 192.168.1.91 | 192.168.1.81  |  3784 | 43991 | udp  | 0:0  | udp-est |                                                  |    780 | 51354 |
| default_web-service1:llb-inst0 | 192.168.1.91 | 192.168.1.90  | 57263 | 30256 | tcp  | 0:0  | est     | fsnat-192.168.1.99,192.168.1.103:55001:w0 |      1 |    52 |
| default_web-service1:llb-inst0 | 192.168.1.91 | 192.168.1.90  | 57264 | 30256 | tcp  | 0:0  | est     | fsnat-192.168.1.99,192.168.1.103:55001:w0 |      1 |    52 |
| default_web-service1:llb-inst0 | 192.168.1.91 | 192.168.1.90  | 57265 | 30256 | tcp  | 0:0  | est     | fsnat-192.168.1.99,192.168.1.103:55001:w0 |      1 |    52 |
| default_web-service1:llb-inst0 | 192.168.1.99 | 192.168.1.103 | 55001 | 57263 | tcp  | 0:0  | est     | fdnat-192.168.1.91,192.168.1.90:30256:w0  |      2 |    92 |
| default_web-service1:llb-inst0 | 192.168.1.99 | 192.168.1.103 | 55001 | 57264 | tcp  | 0:0  | est     | fdnat-192.168.1.91,192.168.1.90:30256:w0  |      2 |    92 |
| default_web-service1:llb-inst0 | 192.168.1.99 | 192.168.1.103 | 55001 | 57265 | tcp  | 0:0  | est     | fdnat-192.168.1.91,192.168.1.90:30256:w0  |      2 |    92 |
| default_web-service2:llb-inst0 | 192.168.1.99 | 192.168.1.103 | 55002 | 57266 | tcp  | 0:0  | est     | fdnat-192.168.1.91,192.168.1.80:31776:w0  |      2 |    92 |
| default_web-service2:llb-inst0 | 192.168.1.99 | 192.168.1.103 | 55002 | 57267 | tcp  | 0:0  | est     | fdnat-192.168.1.91,192.168.1.80:31776:w0  |      2 |    92 |
root@backuploxilb:/# loxicmd get bfd -o wide
| INSTANCE  |   REMOTEIP   |   SOURCEIP   | PORT | INTERVAL  | RETRY COUNT | STATE |
|-----------|--------------|--------------|------|-----------|-------------|-------|
| llb-inst0 | 192.168.1.81 | 192.168.1.91 | 3784 | 100000 us |           3 | BFDUp |
root@backuploxilb:/# loxicmd get ha -o wide
| INSTANCE  | HASTATE |
|-----------|---------|
| llb-inst0 | BACKUP  |
```

Note

- CSP 등의 일반적인 환경에서 BFD 사용을 위한 예는 오픈소스 'LoxiLB' 기술 블로그에서 확인할 수 있습니다. ('LoxiLB' 기술 블로그 주소: https://buly.kr/31TeNYA)

Backup과 Active 'loxilb'에서 K8s에서 지정한 서비스의 TCP Port 번호 할당을 확인하여 웹브라우저에서 서비스를 연결합니다.

```
docker exec -ti loxilb loxicmd get lb -o wide
```

아래는 Backup'loxilb'의 적용된 LB 정책입니다.

```
root@backuploxilb:/# loxicmd get lb -o wide
|    EXT IP    | SEC IPS | SOURCES | HOST | PORT  | PROTO |             NAME             | MARK | SEL |  MODE   |   ENDPOINT    | EPORT | WEIGHT | STATE  | COUNTERS |
|--------------|---------|---------|------|-------|-------|------------------------------|------|-----|---------|---------------|-------|--------|--------|----------|
| 192.168.1.99 |         |         |      | 55001 | tcp   | default_web-service1:llb-inst0 |    0 | rr  | onearm | 192.168.1.80 | 30523 |      1 | -      | 0:0      |
|              |         |         |      |       |       |                              |      |     |         | 192.168.1.90 | 30256 |      1 | -      | 0:0      |
| 192.168.1.99 |         |         |      | 55002 | tcp   | default_web-service2:llb-inst0 |    0 | rr  | onearm | 192.168.1.80 | 31776 |      1 | active | 0:0      |
|              |         |         |      |       |       |                              |      |     |         | 192.168.1.90 | 31088 |      1 | active | 0:0      |
```

아래는 Active 'loxilb'의 적용된 LB 정책입니다.

```
root@activeloxilb:/# loxicmd get lb -o wide
|    EXT IP    | SEC IPS | SOURCES | HOST | PORT  | PROTO |             NAME             | MARK | SEL |  MODE   |   ENDPOINT    | EPORT | WEIGHT | STATE  | COUNTERS |
|--------------|---------|---------|------|-------|-------|------------------------------|------|-----|---------|---------------|-------|--------|--------|----------|
| 192.168.1.99 |         |         |      | 55001 | tcp   | default_web-service1:llb-inst0 |    0 | rr  | onearm | 192.168.1.90 | 30256 |      1 | active | 25:2267  |
| 192.168.1.99 |         |         |      | 55002 | tcp   | default_web-service2:llb-inst0 |    0 | rr  | onearm | 192.168.1.80 | 31776 |      1 | active | 0:0      |
```

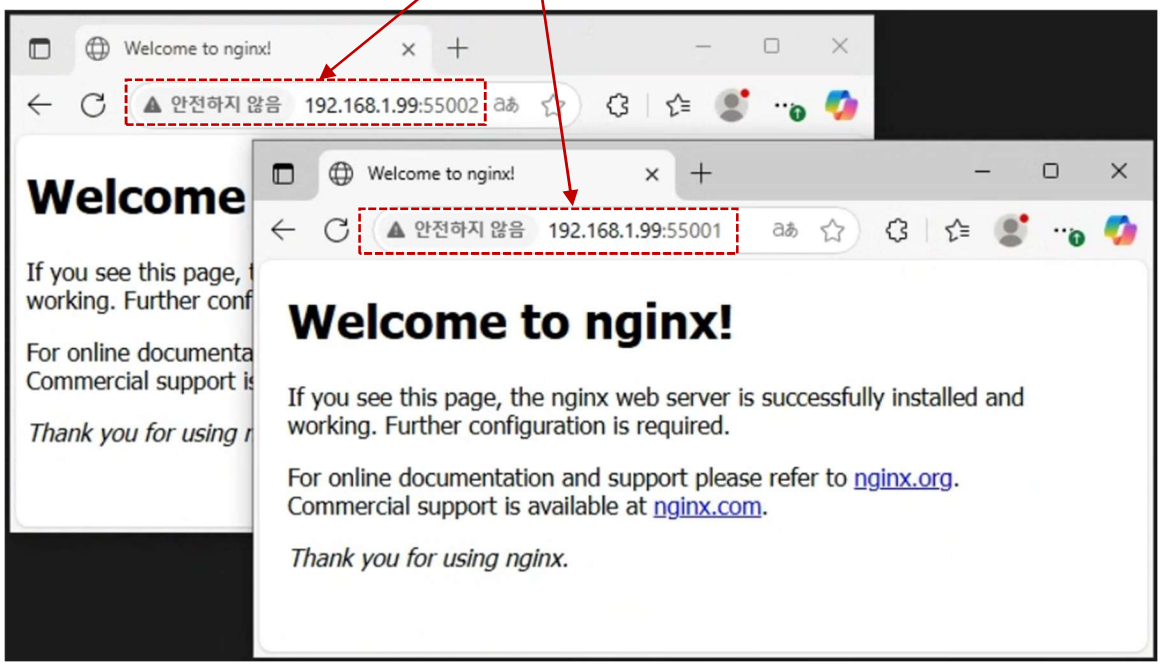

그림 9. 'HA' 환경의 서비스 동작 확인

HA 기능 확인을 위해 Active 'loxilb'를 stop (sudo docker stop loxilb) 하여도 사용자 단말기에서 웹브라우저로 서비스를 전환하며 접속하는 것을 확인합니다. (운영 시에 Backup K8s 클러스터 인프라에 동일한 서비스를 실행하여 단말 접속을 시도하며 정상적인 HA 기능을 확인합니다.)

아래는 Active 'loxilb' 컨테이너를 정지(Stop) 후에 BFD 연결 상태 단절과 Backup 'loxilb' 컨테이너의 상태가 MASTER로 변화된 것을 아래와 같이 확인할 수 있습니다.

```
loxilb@backuploxilb:~$ sudo docker exec -it bfd bash
[sudo] password for loxilb:
root@backuploxilb:/# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet 192.168.1.99/32 scope global lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:d1:05:ef brd ff:ff:ff:ff:ff:ff
    altname enp2s1
    inet 192.168.1.91/24 brd 192.168.1.255 scope global ens33
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fed1:5ef/64 scope link
       valid_lft forever preferred_lft forever
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 02:42:d6:87:5c:e8 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
       valid_lft forever preferred_lft forever
4: llb0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 xdpgeneric/id:270 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:00:ca:fe:fa:ce brd ff:ff:ff:ff:ff:ff
    inet6 fe80::a478:d5ff:fef7:b35f/64 scope link
       valid_lft forever preferred_lft forever
root@backuploxilb:/# loxicmd get ha -o wide
| INSTANCE   | HASTATE |
|------------|---------|
| llb-inst0 | MASTER   |
root@backuploxilb:/# loxicmd get bfd -o wide
| INSTANCE   |    REMOTEIP    |    SOURCEIP    | PORT | INTERVAL   | RETRY COUNT |   STATE   |
|------------|----------------|----------------|------|------------|-------------|-----------|
| llb-inst0 | 192.168.1.81 | 192.168.1.91 | 3784 | 100000 us |           3 | BFDDown |
root@backuploxilb:/# loxicmd get lb -o wide
|     EXT IP     | SEC IPS | SOURCES | HOST | PORT  | PROTO |                 NAME                 | MARK | SEL |  MODE   |    ENDPOINT    | EPORT | WEIGHT | STATE  | COUNTERS |
|----------------|---------|---------|------|-------|-------|--------------------------------------|------|-----|---------|----------------|-------|--------|--------|----------|
| 192.168.1.99 |         |         |      | 55001 | tcp   | default_web-service1:llb-inst0 |    0 | rr  | onearm | 192.168.1.80 | 30523 |      1 | -      | 0:0      |
|                |         |         |      |       |       |                                      |      |     |         | 192.168.1.90 | 30256 |      1 | -      | 0:0      |
| 192.168.1.99 |         |         |      | 55002 | tcp   | default_web-service2:llb-inst0 |    0 | rr  | onearm | 192.168.1.80 | 31776 |      1 | active | 0:0      |
|                |         |         |      |       |       |                                      |      |     |         | 192.168.1.90 | 31088 |      1 | active | 0:0      |
root@backuploxilb:/# loxicmd get ct -o wide
|            SERVICE NAME            |    DESTIP    |    SRCIP     | DPORT | SPORT | PROTO | IDENT | STATE  |                                ACT                                | PACKETS | BYTES |
|------------------------------------|--------------|--------------|-------|-------|-------|-------|--------|-------------------------------------------------------------------|---------|-------|
| default_web-service2:llb-inst0 | 192.168.1.81 | 192.168.1.80 | 61127 | 31776 | tcp   | 0:0   | est    | fsnat-192.168.1.99,192.168.1.103:55002:w0 |       1 |    52 |
| default_web-service2:llb-inst0 | 192.168.1.81 | 192.168.1.80 | 61128 | 31776 | tcp   | 0:0   | est    | fsnat-192.168.1.99,192.168.1.103:55002:w0 |       1 |    52 |
| default_web-service1:llb-inst0 | 192.168.1.81 | 192.168.1.90 | 61118 | 30256 | tcp   | 0:0   | est    | fsnat-192.168.1.99,192.168.1.103:55001:w0 |       1 |    52 |
| default_web-service1:llb-inst0 | 192.168.1.81 | 192.168.1.90 | 61119 | 30256 | tcp   | 0:0   | est    | fsnat-192.168.1.99,192.168.1.103:55001:w0 |       1 |    52 |
| -                                  | 192.168.1.81 | 192.168.1.91 |     0 |     0 | icmp  | 0:0   | closed |                                                                   |       0 |     0 |
| -                                  | 192.168.1.91 | 192.168.1.81 |     0 |     0 | icmp  | 0:0   | closed |                                                                   |      48 |  3840 |
| default_web-service1:llb-inst0 | 192.168.1.99 | 192.168.1.103 | 55001 | 61118 | tcp   | 0:0   | est    | fdnat-192.168.1.81,192.168.1.90:30256:w0 |       2 |    92 |
| default_web-service1:llb-inst0 | 192.168.1.99 | 192.168.1.103 | 55001 | 61119 | tcp   | 0:0   | est    | fdnat-192.168.1.81,192.168.1.90:30256:w0 |       2 |    92 |
| default_web-service2:llb-inst0 | 192.168.1.99 | 192.168.1.103 | 55002 | 61127 | tcp   | 0:0   | est    | fdnat-192.168.1.81,192.168.1.80:31776:w0 |       2 |    92 |
| default_web-service2:llb-inst0 | 192.168.1.99 | 192.168.1.103 | 55002 | 61128 | tcp   | 0:0   | est    | fdnat-192.168.1.81,192.168.1.80:31776:w0 |       2 |    92 |
root@backuploxilb:/#
```

Note

DSR 모드로 loxilb를 사용하고, keepalived/BFD, DNS 등을 통합하는 방법은 이 문서에서는 다루지 않았습니다.

SUMMARY

LoxiLB는 클라우드 네이티브 워크로드를 위한 오픈소스 하이퍼 스케일 소프트웨어 로드밸런싱 솔루션입니다. 핵심 엔진으로 eBPF를 사용하며, Golang을 기반으로 서비스를 제공합니다. 온프레미스, 엣지, 퍼블릭 클라우드 Kubernetes 클러스터 배포를 지원하도록 설계되었습니다.

CHAPTER

05

문제 해결

How to debug and troubleshoot

문제 발생시 해결하는 방법들의 예를 설명합니다.

loxilb 의 Docker컨테이너나 Pod가 Running 상태로 안되나요?

- 해결방법**:**
- 커널버전을 확인해 주세요. loxilb는 최소 **kernel version 5.8 혹은 그 이상**으로 실행해주세요.
- 환경에 맞는 올바른 이미지를 사용했는지 확인해 주세요.

"kubectl get svc" 할때 externalIP 가 pending 이 뜹니다.

1) 외부에서 Loxilb를 실행할때:

- Solution:
  - kube-loxilb.yaml에 적은 loxiURL에 주소가 올바른지 확인해주세요.
  - kube-loxilb(master node)와 loxilb node간 연결을 확인해주세요.

2) in-cluster로 Loxilb를 실행할때:

- Solution: loxilb pod가 올바르게 실행 중 인지 확인해주세요.

3) 노드 설정에 "node.kubernetes.io/exclude-from-external-load-balancers"가 주석되어있는지 확인해주세요.

- Solution: 만약 있다면, Loxilb가 endpoint가 되지않으니"kubectl edit ₩<node-name>"을 통해서 지워주세요

4) 노드 service.yaml에 loadBalancerClass와 type이 맞는지 확인해주세요.

```
spec: loadBalancerClass: loxilb.io/loxilb type: LoadBalancer
```

loxilb log 확인하기

loxilb 는 다양한 이벤트 및 감지한 내용들을 /var/log/loxilb.log에 저장합니다. tail -f 나 원하는 커맨드를 통해서 확인 가능합니다.

```
root@752531364e2c:/# tail -f /var/log/loxilb.log

DBG:  2022/07/10 12:49:27 1:dst-10.10.10.1/32,proto-6,dport-2020,,do-dnat:eip-31.31.31.1,ep-
5001,w-1,alive|eip-32.32.32.1,ep-5001,w-2,alive|eip-100.100.100.1,ep-5001,w-2,alive| pc 0 bc 0

DBG:  2022/07/10 12:49:37 1:dst-10.10.10.1/32,proto-6,dport-2020,,do-dnat:eip-31.31.31.1,ep-
5001,w-1,alive|eip-32.32.32.1,ep-5001,w-2,alive|eip-100.100.100.1,ep-5001,w-2,alive| pc 0 bc 0

DBG:  2022/07/10 12:49:47 1:dst-10.10.10.1/32,proto-6,dport-2020,,do-dnat:eip-31.31.31.1,ep-
5001,w-1,alive|eip-32.32.32.1,ep-5001,w-2,alive|eip-100.100.100.1,ep-5001,w-2,alive| pc 0 bc 0

DBG:  2022/07/10 12:49:57 1:dst-10.10.10.1/32,proto-6,dport-2020,,do-dnat:eip-31.31.31.1,ep-
5001,w-1,alive|eip-32.32.32.1,ep-5001,w-2,alive|eip-100.100.100.1,ep-5001,w-2,alive| pc 0 bc 0

DBG:  2022/07/10 12:50:07 1:dst-10.10.10.1/32,proto-6,dport-2020,,do-dnat:eip-31.31.31.1,ep-
5001,w-1,alive|eip-32.32.32.1,ep-5001,w-2,alive|eip-100.100.100.1,ep-5001,w-2,alive| pc 0 bc 0

DBG:  2022/07/10 12:50:17 1:dst-10.10.10.1/32,proto-6,dport-2020,,do-dnat:eip-31.31.31.1,ep-
5001,w-1,alive|eip-32.32.32.1,ep-5001,w-2,alive|eip-100.100.100.1,ep-5001,w-2,alive| pc 0 bc 0

DBG:  2022/07/10 12:50:27 1:dst-10.10.10.1/32,proto-6,dport-2020,,do-dnat:eip-31.31.31.1,ep-
5001,w-1,alive|eip-32.32.32.1,ep-5001,w-2,alive|eip-100.100.100.1,ep-5001,w-2,alive| pc 0 bc 0

DBG:  2022/07/10 12:50:37 1:dst-10.10.10.1/32,proto-6,dport-2020,,do-dnat:eip-31.31.31.1,ep-
5001,w-1,alive|eip-32.32.32.1,ep-5001,w-2,alive|eip-100.100.100.1,ep-5001,w-2,alive| pc 0 bc 0

DBG:  2022/07/10 12:50:47 1:dst-10.10.10.1/32,proto-6,dport-2020,,do-dnat:eip-31.31.31.1,ep-
5001,w-1,alive|eip-32.32.32.1,ep-5001,w-2,alive|eip-100.100.100.1,ep-5001,w-2,alive| pc 0 bc 0

DBG:  2022/07/10 12:50:57 1:dst-10.10.10.1/32,proto-6,dport-2020,,do-dnat:eip-31.31.31.1,ep-
5001,w-1,alive|eip-32.32.32.1,ep-5001,w-2,alive|eip-100.100.100.1,ep-5001,w-2,alive| pc 0 bc 0
```

.

loxicmd 를 통한 loxilb의 내부 정보 디버깅

```
### Spawn a bash shell of loxilb docker
docker exec -it loxilb bash

root@752531364e2c:/# loxicmd get lb
```

```
| EXTERNALIP | PORT | PROTOCOL | SELECT | # OF ENDPOINTS |
|------------|------|----------|--------|----------------|
| 10.10.10.1 | 2020 | tcp      |      0 |              3 |

root@752531364e2c:/# loxicmd get lb -o wide
| EXTERNALIP | PORT | PROTOCOL | SELECT |  ENDPOINTIP    | TARGETPORT | WEIGHT |
|------------|------|----------|--------|----------------|------------|--------|
| 10.10.10.1 | 2020 | tcp      |      0 | 31.31.31.1     |       5001 |      1 |
|            |      |          |        | 32.32.32.1     |       5001 |      2 |
|            |      |          |        | 100.100.100.1  |       5001 |      2 |

root@0c4f9175c983:/# loxicmd get conntrack
| DESTINATIONIP |  SOURCEIP   | DESTINATIONPORT | SOURCEPORT | PROTOCOL |    STATE    | ACT |
|---------------|-------------|-----------------|------------|----------|-------------|-----|
| 127.0.0.1     | 127.0.0.1   |           11111 |      47180 | tcp      | closed-wait |     |
| 127.0.0.1     | 127.0.0.1   |           11111 |      47182 | tcp      | est         |     |
| 32.32.32.1    | 31.31.31.1  |           35068 |      35068 | icmp     | bidir       |

root@65ad9b2f1b7f:/# loxicmd get port
| INDEX | PORTNAME |        MAC        | LINK/STATE  |    L3INFO     |    L2INFO     |
|-------|----------|-------------------|-------------|---------------|---------------|
|     1 | lo       | 00:00:00:00:00:00 | true/false  | Routed: false | IsPVID: true  |
|       |          |                   |             | IPv4 : []     | VID : 3801    |
|       |          |                   |             | IPv6 : []     |               |
|     2 | vlan3801 | aa:bb:cc:dd:ee:ff | true/true   | Routed: false | IsPVID: false |
|       |          |                   |             | IPv4 : []     | VID : 3801    |
|       |          |                   |             | IPv6 : []     |               |
|     3 | llb0     | 42:6e:9b:7f:ff:36 | true/false  | Routed: false | IsPVID: true  |
|       |          |                   |             | IPv4 : []     | VID : 3803    |
|       |          |                   |             | IPv6 : []     |               |
|     4 | vlan3803 | aa:bb:cc:dd:ee:ff | true/true   | Routed: false | IsPVID: false |
|       |          |                   |             | IPv4 : []     | VID : 3803    |
|       |          |                   |             | IPv6 : []     |               |
|     5 | eth0     | 02:42:ac:1e:01:c1 | true/true   | Routed: false | IsPVID: true  |
|       |          |                   |             | IPv4 : []     | VID : 3805    |
|       |          |                   |             | IPv6 : []     |               |
|     6 | vlan3805 | aa:bb:cc:dd:ee:ff | true/true   | Routed: false | IsPVID: false |
|       |          |                   |             | IPv4 : []     | VID : 3805    |
|       |          |                   |             | IPv6 : []     |               |
|     7 | enp1     | fe:84:23:ac:41:31 | false/false | Routed: false | IsPVID: true  |
|       |          |                   |             | IPv4 : []     | VID : 3807    |
|       |          |                   |             | IPv6 : []     |               |
|     8 | vlan3807 | aa:bb:cc:dd:ee:ff | true/true   | Routed: false | IsPVID: false |
|       |          |                   |             | IPv4 : []     | VID : 3807    |
|       |          |                   |             | IPv6 : []     |               |
|     9 | enp2     | d6:3c:7f:9e:58:5c | false/false | Routed: false | IsPVID: true  |
|       |          |                   |             | IPv4 : []     | VID : 3809    |
|       |          |                   |             | IPv6 : []     |               |
|    10 | vlan3809 | aa:bb:cc:dd:ee:ff | true/true   | Routed: false | IsPVID: false |
```

```
|      |         |                   |             | IPv4 : []     | VID : 3809    |
|      |         |                   |             | IPv6 : []     |               |
|   11 | enp2v15 | 8a:9e:99:aa:f9:c3 | false/false | Routed: false | IsPVID: true  |
|      |         |                   |             | IPv4 : []     | VID : 3811    |
|      |         |                   |             | IPv6 : []     |               |
|   12 | vlan3811| aa:bb:cc:dd:ee:ff | true/true   | Routed: false | IsPVID: false |
|      |         |                   |             | IPv4 : []     | VID : 3811    |
|      |         |                   |             | IPv6 : []     |               |
|   13 | enp3    | f2:c7:4b:ac:fd:3e | false/false | Routed: false | IsPVID: true  |
|      |         |                   |             | IPv4 : []     | VID : 3813    |
|      |         |                   |             | IPv6 : []     |               |
|   14 | vlan3813| aa:bb:cc:dd:ee:ff | true/true   | Routed: false | IsPVID: false |
|      |         |                   |             | IPv4 : []     | VID : 3813    |
|      |         |                   |             | IPv6 : []     |               |
|   15 | enp4    | 12:d2:c3:79:f3:6a | false/false | Routed: false | IsPVID: true  |
|      |         |                   |             | IPv4 : []     | VID : 3815    |
|      |         |                   |             | IPv6 : []     |               |
|   16 | vlan3815| aa:bb:cc:dd:ee:ff | true/true   | Routed: false | IsPVID: false |
|      |         |                   |             | IPv4 : []     | VID : 3815    |
|      |         |                   |             | IPv6 : []     |               |
|   17 | vlan100 | 56:2e:76:b2:71:48 | false/false | Routed: false | IsPVID: false |
|      |         |                   |             | IPv4 : []     | VID : 100     |
|      |         |                   |             | IPv6 : []     |               |
```

loxilb 커널과 eBPF 컴포넌트 디버깅

loxilb 는 다양한 eBPF map을 만들어 DP 실행에 사용합니다. 이 Map은 OS filesystem에 Pin 되어있어 bpftool 과 같은 bpf 관련 어플리케이션으로 디버깅 가능합니다.

```
root@0c4f9175c983:/# ls -lart /opt/loxilb/dp/bpf/
total 0
-rw------- 1 root root 0 Jul 10 11:32 xfis
-rw------- 1 root root 0 Jul 10 11:32 xfck
-rw------- 1 root root 0 Jul 10 11:32 xctk
-rw------- 1 root root 0 Jul 10 11:32 tx_intf_stats_map
-rw------- 1 root root 0 Jul 10 11:32 tx_intf_map
-rw------- 1 root root 0 Jul 10 11:32 tx_bd_stats_map
-rw------- 1 root root 0 Jul 10 11:32 tmac_stats_map
-rw------- 1 root root 0 Jul 10 11:32 tmac_map
-rw------- 1 root root 0 Jul 10 11:32 smac_map
-rw------- 1 root root 0 Jul 10 11:32 sess_v4_stats_map
-rw------- 1 root root 0 Jul 10 11:32 sess_v4_map
```

```
-rw------- 1 root root 0 Jul 10 11:32 rt_v6_stats_map
-rw------- 1 root root 0 Jul 10 11:32 rt_v4_stats_map
-rw------- 1 root root 0 Jul 10 11:32 rt_v4_map
-rw------- 1 root root 0 Jul 10 11:32 polx_map
-rw------- 1 root root 0 Jul 10 11:32 pkts
-rw------- 1 root root 0 Jul 10 11:32 pkt_ring
-rw------- 1 root root 0 Jul 10 11:32 pgm_tbl
-rw------- 1 root root 0 Jul 10 11:32 nat_v4_map
-rw------- 1 root root 0 Jul 10 11:32 mirr_map
-rw------- 1 root root 0 Jul 10 11:32 intf_stats_map
-rw------- 1 root root 0 Jul 10 11:32 intf_map
-rw------- 1 root root 0 Jul 10 11:32 fcas
-rw------- 1 root root 0 Jul 10 11:32 fc_v4_stats_map
-rw------- 1 root root 0 Jul 10 11:32 fc_v4_map
-rw------- 1 root root 0 Jul 10 11:32 dmac_map
-rw------- 1 root root 0 Jul 10 11:32 ct_v4_map
-rw------- 1 root root 0 Jul 10 11:32 bd_stats_map
-rw------- 1 root root 0 Jul 10 11:32 acl_v6_stats_map
-rw------- 1 root root 0 Jul 10 11:32 acl_v4_stats_map
-rw------- 1 root root 0 Jul 10 11:32 acl_v4_map
drwxrwxrwt 3 root root 0 Jul 10 11:32 ..
lrwxrwxrwx 1 root root 0 Jul 10 11:32 xdp -> /opt/loxilb/dp/bpf//tc/
drwx------ 3 root root 0 Jul 10 11:32 tc
lrwxrwxrwx 1 root root 0 Jul 10 11:32 ip -> /opt/loxilb/dp/bpf//tc/
```

```
root@752531364e2c:/# bpftool map dump pinned /opt/loxilb/dp/bpf/intf_map
[{
        "key": {
            "ifindex": 2,
            "ing_vid": 0,
            "pad": 0
        },
        "value": {
            "ca": {
                "act_type": 11,
                "ftrap": 0,
                "oif": 0,
                "cidx": 0
            },
            "": {
                "set_ifi": {
                    "xdp_ifidx": 1,
                    "zone": 0,
                    "bd": 3801,
                    "mirr": 0,
                    "polid": 0,
                    "r": [0,0,0,0,0,0
```

```
                    ]
                }
            }
        }
    },{
        "key": {
            "ifindex": 3,
            "ing_vid": 0,
            "pad": 0
        },
        "value": {
            "ca": {
                "act_type": 11,
                "ftrap": 0,
                "oif": 0,
                "cidx": 0
            },
            "": {
                "set_ifi": {
                    "xdp_ifidx": 3,
                    "zone": 0,
                    "bd": 3803,
                    "mirr": 0,
                    "polid": 0,
                    "r": [0,0,0,0,0,0
                    ]
                }
            }
        }
    }
]
```


```
root@752531364e2c:/# bpftool map dump pinned /opt/loxilb/dp/bpf/nat_v4_map
[{
        "key": {
            "daddr": 17435146,
            "dport": 58375,
            "zone": 0,
            "l4proto": 6
        },
        "value": {
            "ca": {
                "act_type": 5,
                "ftrap": 0,
                "oif": 0,
                "cidx": 1
            },
            "lock": {
                "val": 0
```

```
            },
            "nxfrm": 3,
            "sel_hint": 0,
            "sel_type": 0,
            "nxfrms": [{
                    "nat_flags": 0,
                    "inactive": 0,
                    "wprio": 1,
                    "res": 0,
                    "nat_xport": 35091,
                    "nat_xip": 18816799
                },{
                    "nat_flags": 0,
                    "inactive": 0,
                    "wprio": 2,
                    "res": 0,
                    "nat_xport": 35091,
                    "nat_xip": 18882592
                },{
                    "nat_flags": 0,
                    "inactive": 0,
                    "wprio": 2,
                    "res": 0,
                    "nat_xport": 35091,
                    "nat_xip": 23356516
                },{
                    "nat_flags": 0,
                    "inactive": 1,
                    "wprio": 0,
                    "res": 0,
                    "nat_xport": 0,
                    "nat_xip": 0
                },{
                    "nat_flags": 0,
                    "inactive": 1,
                    "wprio": 0,
                    "res": 0,
                    "nat_xport": 0,
                    "nat_xip": 0
                },{
                    "nat_flags": 0,
                    "inactive": 1,
                    "wprio": 0,
                    "res": 0,
                    "nat_xport": 0,
                    "nat_xip": 0
                },{
                    "nat_flags": 0,
                    "inactive": 1,
                    "wprio": 0,
                    "res": 0,
                    "nat_xport": 0,
                    "nat_xip": 0
                },{
                    "nat_flags": 0,
                    "inactive": 1,
```

```
                    "wprio": 0,
                    "res": 0,
                    "nat_xport": 0,
                    "nat_xip": 0
                }, {
                    "nat_flags": 0,
                    "inactive": 1,
                    "wprio": 0,
                    "res": 0,
                    "nat_xport": 0,
                    "nat_xip": 0
                }, {
                    "nat_flags": 0,
                    "inactive": 1,
                    "wprio": 0,
                    "res": 0,
                    "nat_xport": 0,
                    "nat_xip": 0
                }, {
                    "nat_flags": 0,
                    "inactive": 1,
                    "wprio": 0,
                    "res": 0,
                    "nat_xport": 0,
                    "nat_xip": 0
                }, {
                    "nat_flags": 0,
                    "inactive": 1,
                    "wprio": 0,
                    "res": 0,
                    "nat_xport": 0,
                    "nat_xip": 0
                }, {
                    "nat_flags": 0,
                    "inactive": 1,
                    "wprio": 0,
                    "res": 0,
                    "nat_xport": 0,
                    "nat_xip": 0
                }, {
                    "nat_flags": 0,
                    "inactive": 1,
                    "wprio": 0,
                    "res": 0,
                    "nat_xport": 0,
                    "nat_xip": 0
                }, {
                    "nat_flags": 0,
                    "inactive": 1,
                    "wprio": 0,
                    "res": 0,
                    "nat_xport": 0,
                    "nat_xip": 0
                }, {
                    "nat_flags": 0,
                    "inactive": 1,
```

```
                    "wprio": 0,
                    "res": 0,
                    "nat_xport": 0,
                    "nat_xip": 0
                }
            ]
        }
    }
]
```

eBPF 커널 로그 확인하기

마지막으로, Linux 커널은 일반 eBPF 디버그 로그를 /sys/kernel/debug/tracing/trace_pipe에 출력합니다. loxilb eBPF 모듈은 일반 작동 모드에서 로그를 생성하지 않지만 재컴파일 후에 로그를 활성화할 수 있습니다.

```
root@752531364e2c:/# cat /sys/kernel/debug/tracing/trace_pipe
         loxicmd-30524   [001] d.s1 27870.170790: bpf_trace_printk: out-dir
         loxicmd-30524   [001] d.s1 27870.170791: bpf_trace_printk: smr 4
         loxicmd-30529   [000] d.s1 27871.617467: bpf_trace_printk: [CTRK] start

         loxicmd-30529   [000] d.s1 27871.617484: bpf_trace_printk: new-ct4
         loxicmd-30529   [000] d.s1 27871.617486: bpf_trace_printk: in-dir
         loxicmd-30529   [000] d.s1 27871.617488: bpf_trace_printk: smr 0
         loxicmd-30529   [000] d.s1 27871.617503: bpf_trace_printk: [CTRK] start

         loxicmd-30529   [000] d.s1 27871.617503: bpf_trace_printk: out-dir
         loxicmd-30529   [000] d.s1 27871.617504: bpf_trace_printk: smr 4
            sshd-30790   [000] d.s1 27970.031847: bpf_trace_printk: [CTRK] start

            sshd-30790   [000] d.s1 27970.031866: bpf_trace_printk: new-ct4
            sshd-30790   [000] d.s1 27970.031868: bpf_trace_printk: in-dir
            sshd-30790   [000] d.s1 27970.031870: bpf_trace_printk: smr 0
            sshd-30790   [000] d.s1 27970.031887: bpf_trace_printk: [CTRK] start

            sshd-30790   [000] d.s1 27970.031887: bpf_trace_printk: out-dir
            sshd-30790   [000] d.s1 27970.031888: bpf_trace_printk: smr 0
            sshd-30790   [000] d.s1 27970.031900: bpf_trace_printk: [CTRK] start
```

인터페이스 별 통계 모니터링 하기(diff, peak, gen)

인터페이스 별 통계 정보는 3 개의 값으로 보여줍니다. 1) 변화량 Difference, 2) 정점 Peak, 3) 원본 General 등으로 확인 할 수 있습니다. 보여주는 값으로는

RX,TX 별 Packets, Bytes, Errors 입니다. 더 많은 정보를 보기 위해서는 -o json 옵션을 통해 그 외의 값들을 확인할 수 있습니다. 확인 가능한 특징은 다음과 같습니다.

- 변화량: 기준(초)당 흐르는 데이터의 변화량을 제공하며, 기준을 10 초로 둔다면, 10 초간 인터페이스의 패킷의 수, 패킷의 크기, Error 패킷 등을 제공
- 정점: 기준(초) * 40 의 범위에서 변화량이 가장 높았던 지점을 제공하며, 범위가 지난 이후에는 이전 변화량보다 낮은 수치가 나와도 삭제하고 새로운 정점값을 제공
- 원본: 변화량과 정점을 구하기 위한 원본으로 일반적으로 누적 값을 제공하며, LoxiLB 가 시작한 후의 값을 보여주는 것이 아닌, 시스템의 실행 시점부터 누적된 통계값을 제공

인터페이스 별 통계를 모니터링 하기 위해서는 기준을 등록하는 과정이 필요합니다. 기준을 달리해도 동작에는 영향이 없습니다. 1 개의 시간을 기준으로 등록을 하면 변화량, 정점 둘 다 확인할 수 있습니다.

시작과 함께 등록

```
# --stat-period="시간 1(초),시간 2(초),시간 3(초),..."
loxilb --stat-period="10"
```

- **--stat-period:** 옵션을 통해 모니터링을 시작 시 등록 할 수 있습니다.

커맨드로 등록

아래 커맨드를 사용해서 운영 중 일 때도 등록할 수 있습니다.

```
# loxicmd create statistic {시간(초)}
# loxicmd create stat {시간(초)}
# loxicmd create stats {시간(초)}
loxicmd create statisitic 10 # 10 초 기준의 모니터링 추가
```

조회

아래의 커맨드를 사용하여 변화량, 정점, 원본 등을 확인할 수 있습니다.

```
# 변화량 확인
loxicmd get statistic difference
loxicmd get statistic diff # differnce와 동일
loxicmd get statistic diff -o json # 전체 값 보기
# 정점 확인
loxicmd get statistic peak
# 원본 확인
loxicmd get statistic general
loxicmd get statistic gen  # general과 동일
```

삭제

아래의 커맨드를 사용해 등록된 모니터링을 삭제할 수 있습니다.

```
#loxicmd delete statistic {시간(초)}
#loxicmd delete stat {시간(초)}
#loxicmd delete stats {시간(초)}
loxicmd delete statistic 10
```

LB rule 스탯 모니터링 하기

LB rule을 조회 시 -o wide 옵션을 통해 counter를 확인할 수 있습니다. 각 endpoint별 전송되는 packet과 byte를 보여줍니다.

```
loxicmd get lb -o wide
```

conntrack 스탯 모니터링 하기

conntrack을 조회할 때 packets와 bytes 컬럼을 통해 패킷의 수와 크기를 볼 수 있습니다.

```
loxicmd get conntrack
```

SUMMARY

LoxiLB는 클라우드 네이티브 워크로드를 위한 오픈소스 하이퍼 스케일 소프트웨어 로드밸런싱 솔루션입니다. 핵심 엔진으로 eBPF를 사용하며, Golang을 기반으로 서비스를 제공합니다. 온프레미스, 엣지, 퍼블릭 클라우드 Kubernetes 클러스터 배포를 지원하도록 설계되었습니다.

부록

01

기능 확인

Endpoint

end-point 상태 가져오기

```
loxicmd get ep
```

end-point 설정하기

```
# loxicmd create endpoint IP [--name=<id>] [--probetype=<probetype>] [--
probereq=<probereq>] [--proberesp=<proberesp>] [--probeport=<port>] [--
period=<period>] [--retries=<retries>] loxicmd create endpoint 32.32.32.1 --
probetype=http --probeport=8080 --period=60 --retries=2
```

- IP(string): Endpoint 아이피 주소
- name(string): Endpoint 이름
- probetype(string): ping, http, https, udp, tcp, sctp, none 등 엔드포인트의 서비스 상태
- probereq(string): http/https의 경우 보내는 URI 주소
- proberesp(string): http/https의 경우 받는 String의 값
- probeport(int): http, https, tcp, udp, sctp의 경우 L4 Port번호
- period(int): probing 주기
- retries(int): endPoint가 죽었을때 재 시도 횟수

Note

Endpoint가 처음 생성될 때는 "name" 이 필요하지 않고, Loxilb가 자동으로 이름을 할당합니다.
Endpoint가 처음 생성되면 15초 이내로 서비스의 상태가 결정됩니다. (서비스가 바로 실행되는지 확신 할 수 없으므로, 초기 대기시간을 갖습니다. 해당 값은 변경 불가능합니다.)
이후 기본값(60초) 또는 사용자의 지정값에 따라 모니터링이 진행됩니다.
Endpoint가 비활성화되면 내부적으로 타임아웃을 두어 호출을 최소화합니다. 오직 Endpoint가 활성화 되어있을 때만 Probe주기가 작동합니다.

UDP end-point는 두가지의 방법으로 상태를 확인합니다:

만약 UDP가 미리 정해 둔 응답이 있는 경우 아래와 같이 사용합니다.

```
loxicmd create endpoint 172.1.217.133 --name="udpep1" --probetype=udp --
probeport=32031 --period=60 --retries=2 --probereq="probe" --proberesp="hello"
```

아래와 같이 'loxicmd get ep' 명령어로 상태를 확인 할 수 있습니다.

```
root@92bcacbbf6b8:/# loxicmd create endpoint 172.1.217.133 --name="udpep1" --probetype=udp --probeport=32031 --
period=60 --retries=2 --probereq="probe" --proberesp="hello"
Debug: response.StatusCode: 200
Success
root@92bcacbbf6b8:/# loxicmd get ep
|     HOST      |  NAME  |   PTYPE   | PORT  | DURATION | RETRIES | MINDELAY | AVGDELAY | MAXDELAY | STATE |
|---------------|--------|-----------|-------|----------|---------|----------|----------|----------|-------|
| 172.1.217.133 | udpep1 | udp:probe | 32031 |       60 |       2 |          |          |          | nok   |
```

- Host : 엔드포인트 IP주소
- Name : 엔드포인트 이름
- ptype : 상태 확인에 필요한 endpoint타입
- port : 엔드포인트 Port
- duration :엔트포인트 확인 주기
- retries : 재시도 횟수
- minDelay : ICMP패킷의 최소 지연값
- avgDelay : ICMP패킷의 평균 지연값
- maxDelay : ICMP패킷의 최대 지연값
- State : 엔드포인트 상태

만약 미리 정해 둔 응답이 없는 경우에는 "ICMP Port unreachable" 메세지를 통해 UDP가 동작 중 인지 아닌지 확인합니다. "ICMP Port unreachable" 메세지가 오면 Endpoint가 비활성화 되어있는 상태를 뜻합니다.

```
loxicmd create endpoint 32.32.32.1 --probetype=http --probeport=8080 --period=60 --
retries=2
```

UDP가 동작 중 인지 아닌지 확인합니다. "ICMP Port unreachable" 메세지가 오면 Endpoint가 비활성화 되어있는 상태를 뜻합니다.

```
loxicmd get ep
|    HOST    |        NAME         | PTYPE | PORT | DURATION | RETRIES | MINDELAY | AVGDELAY | MAXDELAY | STATE |
|------------|---------------------|-------|------|----------|---------|----------|----------|----------|-------|
| 32.32.32.1 | 32.32.32.1_http_8080 | http: | 8080 |       60 |       2 | 0s       | 0s       | 0s       | ok    |
```

```
# Modify duration and retry count using name
loxicmd get ep
|    HOST    |        NAME         | PTYPE | PORT | DURATION | RETRIES | MINDELAY | AVGDELAY  | MAXDELAY  | STATE |
|------------|---------------------|-------|------|----------|---------|----------|-----------|-----------|-------|
```

```
| 32.32.32.1 | 32.32.32.1_http_8080 | http: | 8080 |       40 |       4 | 0s       | 0s        | 0s        | ok    |
```

https 프로빙 정보로 "endpoint" 만들기

```
# loxicmd create endpoint IP [--name=<id>] [--probetype=<probetype>] [--
probereq=<probereq>] [--proberesp=<proberesp>] [--probeport=<port>] [--
period=<period>] [--retries=<retries>]
loxicmd create endpoint 32.32.32.1 --probetype=https --probeport=8080 --
probereq="health" --proberesp="OK" --period=60 --retries=2
```

Note

loxilb에서 TLS 연결을 위해 CA certificate이 필요합니다. mTLS 연결을 위해서 개인키 및 개인 인증서등이 필요합니다. 관리자는 연결에 필요한 모든 인증서들을 "/opt/loxilb/cert/rootCA.crt" 또는 endpoint 당 인증서를 "/opt/loxilb/cert/₩<IP>/rootCA.crt"에 위치시켜야 합니다.
개인키는 "/opt/loxilb/cert/server.key" 및 개인 인증서는 "/opt/loxilb/cert/server.crt"의 이름으로 있어야 합니다. Minica (https://github.com/jsha/minica), Certstrap {https://github.com/square/certstrap} , CICD (https://github.com/loxilb-io/loxilb/tree/main/cicd/httpsep) 테스트 케이스를 참조해 주세요.

Endpoint YAML 예시

```
apiVersion: netlox/v1
kind: Endpoint
metadata:
  name: test
spec:
  hostName: "Test"
  description: string
  inactiveReTries: 0
  probeType: string
  probeReqUrl: string
  probeDuration: 0
  probePort: 0
```

endpoint 삭제

```
loxicmd delete endpoint 31.31.31.31
```

Firewall

LoxLB는 명령어로 방화벽 정책을 설정 할 수 있습니다.

Firewall 상태 가져오기

```
loxicmd get firewall
```

Firewall 생성하기

```
#loxicmd create firewall --firewallRule=<ruleKey>:<ruleValue>, [--allow] [--drop] [--trap] [--redirect=<PortName>] [--setmark=<FwMark>] [--snat=<IPAddress>]

loxicmd create firewall --firewallRule="sourceIP:1.2.3.2/32,destinationIP:2.3.1.2/32,preference:200" -allow

loxicmd create firewall --firewallRule="sourceIP:1.2.3.2/32,destinationIP:2.3.1.2/32,preference:200" --allow --setmark=10

loxicmd create firewall --firewallRule="sourceIP:1.2.3.2/32,destinationIP:2.3.1.2/32,preference:200" -drop

loxicmd create firewall --firewallRule="sourceIP:1.2.3.2/32,destinationIP:2.3.1.2/32,preference:200" -trap

loxicmd create firewall --firewallRule="sourceIP:1.2.3.2/32,destinationIP:2.3.1.2/32,preference:200" --redirect=ens0

loxicmd create firewall --firewallRule="sourceIP:1.2.3.2/32,minDestinationPort:8805,maxDestinationPort:8805" --snat=10.66.1.21
```

- sourceIP(string) - CIDR포맷의 출발지 아이피 및 마스크
- destinationIP(string) - CIDR포맷의 목적지 아이피 및 마스크
- minSourcePort(int) - 출발지 port 의 최소값
- maxSourcePort(int) - 출발지 port 의 최대값
- minDestinationPort(int) - 목적지 port의 최소값
- maxDestinationPort(int) - 목적지 port의 최대값
- protocol(int) - 프로토콜 번호
- portName(string) - Ingress인터페이스 이름
- preference(int) - 우선순위

아래는 firewall 명령어로 정책을 적용 후 'loxicmd get firewall' 명령어로 정책을 확인하는(예) 입니다.

```
root@92bcacbbf6b8:/# loxicmd create firewall --firewallRule="sourceIP:1.2.3.2/32,destinationIP:2.3.1.2/32,preference:200" -allow
Debug: response.StatusCode: 200eate firewall --firewallRule="sourceIP:1.2.3.2/32,destinationIP:2.3.1.2/32,preference:200" -allow
root@92bcacbbf6b8:/# loxicmd get firewall
| SOURCE IP  | DESTINATION IP | MIN SPORT | MAX SPORT | MIN DPORT | MAX DPORT | PROTOCOL | PORT NAME | PREFERENCE | OPTION |
|------------|----------------|-----------|-----------|-----------|-----------|----------|-----------|------------|--------|
| 1.2.3.2/32 | 2.3.1.2/32     |         0 |         0 |         0 |         0 |        0 |           |        200 | Drop   |
root@92bcacbbf6b8:/#
```

- Source IP : Firewall을 적용할 출발지 IP 주소 및 대역
- destination IP :Firewall을 적용할 목적지 IP 주소 및 대역
- min SPort : Firewall을 적용할 출발지 Port 번호 대역의 최소값
- max SPort : Firewall을 적용할 출발지 Port 번호 대역의 최대값
- min DPort : Firewall을 적용할 목적지 Port 번호 대역의 최소값
- max DPort : Firewall을 적용할 목적지 Port 번호 대역의 최대값
- protocol : Firewall을 적용할 프로토콜 번호
- port Name : Firewall을 적용할 인터페이스 이름
- preference : Firewall의 우선순위
- Option : Firewall에 적용된 action
- Counters : Firewall에 매치된 패킷 스탯

SNAT 생성하기

SNAT는 우선 LB Rule을 생성 하신 뒤, firewall 명령어를 통해 적용합니다. SNAT를 생성하고 *loxicmd get fw* 명령어를 통해 실제 결과를 보면 Option에 SNAT가 들어간 것을 확인 할 수 있습니다.

Endpoint인 10.80.200.22, 10.80.200.23 Port 8805를 가진 트래픽에 대해서 SNAT로 10.66.1.21을 하는 경우의 예시를 보면 다음과 같습니다.

```
loxicmd create firewall --
firewallRule="sourceIP:10.80.200.22/32,minDestinationPort:8805,maxDestinationPort:8805" --snat=10.66.1.21
```

Firewall YAML 예시

```
apiVersion: netlox/v1
kind: Firewall
metadata:
  name: test
spec:
  ruleArguments:
```

```
    sourceIP: 192.169.1.2/24
    destinationIP: 192.169.2.1/24
    preference: 200
  opts:
    allow: true
```

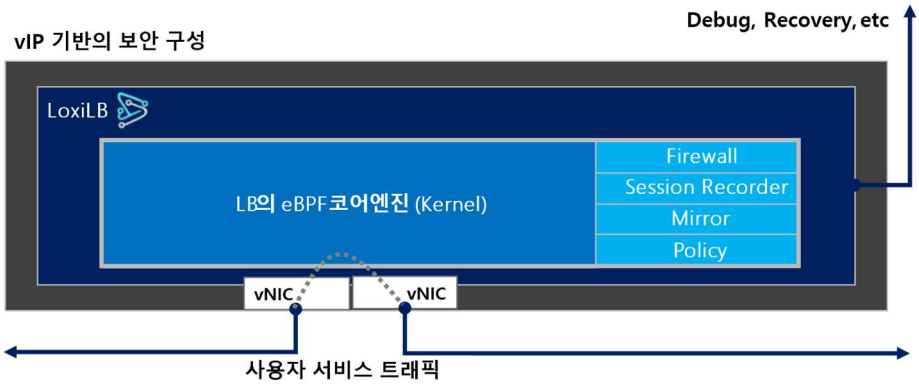

그림 10. 방화벽 기능을 제공하는 보안 강화 구성

Firewall 삭제

```
#loxicmd delete firewall --firewallRule=<ruleKey>:<ruleValue>
loxicmd delete firewall --
firewallRule="sourceIP:1.2.3.2/32,destinationIP:2.3.1.2/32,preference:200
```

IPaddress

IPaddress 가져오기

```
loxicmd get ip
```

IPaddress 추가하기

```
#loxicmd create ip <DeviceIPNet> <device>
loxicmd create ip 192.168.0.1/24 eno7
```

- DeviceIPNet(string): CIDR구조의 IP address와 mask
- device(string): IP주소를 추가할 인터페이스 이름

IPaddress YAML 예시

```
apiVersion: netlox/v1
kind: IPaddress
metadata:
  name: test
spec:
  dev: eno8
ipAddress: 192.168.23.1/32
```

IPaddress 삭제

```
#loxicmd delete ip <DeviceIPNet> <device> loxicmd delete ip 192.168.0.1/24 eno7
```

Load Balancer

아래와 같이 'loxicmd create lb --help' 명령어를 실행하여 Load Balancer 명령어 체계를 확인할 수 있습니다.

```
root@92bcacbbf6b8:/# loxicmd create lb —help
Create a LoadBalancer

—select value options
        rr - select the lb end-points based on round-robin
        hash - select the lb end-points based on hashing
        priority - select the lb based on weighted round-robin
—mode value options
        onearm - LB put LB-IP as srcIP
        fullnat - LB put Service IP as srcIP

ex) loxicmd create lb 192.168.0.200 —tcp=80:32015 —endpoints=10.212.0.1:1,10.212.0.2:1,10.212.0.3:1
    loxicmd create lb 192.168.0.200 —tcp=80:32015 —name="http-service" —
endpoints=10.212.0.1:1,10.212.0.2:1,10.212.0.3:1
    loxicmd create lb 192.168.0.200 —udp=80:32015 —endpoints=10.212.0.1:1,10.212.0.2:1,10.212.0.3:1 —mark=10
    loxicmd create lb 192.168.0.200 —tcp=80:32015 —udp=80:32015 —endpoints=10.212.0.1:1,10.212.0.2:1,10.212.0.3:1
    loxicmd create lb 192.168.0.200 —select=hash —tcp=80:32015 —endpoints=10.212.0.1:1,10.212.0.2:1,10.212.0.3:1
    loxicmd create lb 192.168.0.200 —tcp=80:32015 —endpoints=10.212.0.1:1,10.212.0.2:1,10.212.0.3:1 —mode=dsr
    loxicmd create lb 192.168.0.200 —sctp=37412:38412 —secips=192.168.0.201,192.168.0.202 —
endpoints=10.212.0.1:1,10.212.0.2:1,10.212.0.3:1
    loxicmd create lb  2001::1 —tcp=2020:8080 —endpoints=4ffe::1:1,5ffe::1:1,6ffe::1:1
    loxicmd create lb  2001::1 —tcp=2020:8080 —endpoints=31.31.31.1:1,32.32.32.1:1,33.33.33.1:1

Usage:
  loxicmd create lb IP [—select=<rr|hash|priority>] [—tcp=<port>:<targetPort>] [—udp=<port>:<targetPort>] [—
sctp=<port>:<targetPort>] [—icmp] [—mark=<val>] [—secips=<ip>,][—endpoints=<ip>:<weight>,] [—mode=<onearm|fullnat>]
[—bgp] [—monitor] [—inatimeout=<to>] [—name=<service-name>] [flags]

Flags:
      —bgp                Enable BGP in the load balancer
      —endpoints strings  Endpoints is pairs that can be specified as '<endpointIP>:<Weight>'
  -h, —help               help for lb
      —icmp               ICMP Ping packet Load balancer
      —inatimeout uint32  Specify the timeout (in seconds) after which a LB session will be reset for inactivity
      —mark uint16        Specify the mark num to segregate a load-balancer VIP service
      —mode string        NAT mode for load balancer rule
      —monitor            Enable monitoring end-points of this rule
      —name string        Name for load balancer rule
      —sctp strings       Port pairs can be specified as '<port>:<targetPort>'
      —secips strings     Secondary IPs for SCTP multihoming rule specified as '<secondaryIP>'
      —select string      Select the hash algorithm for the load balance.(ex) rr, hash, priority (default "rr")
      —tcp strings        Port pairs can be specified as '<port>:<targetPort>'
      —udp strings        Port pairs can be specified as '<port>:<targetPort>'

Global Flags:
  -s, —apiserver string   Set API server IP address (default "127.0.0.1")
  -o, —output string      Set output layer (ex.) wide, json)
  -p, —port int16         Set API server port number (default 11111)
      —protocol string    Set API server http/https (default "http")
  -t, —timeout int16      Set timeout (default 10)
root@92bcacbbf6b8:/#
```

Load-balancer rule 가져오기 (대략적인 LB rule 정보 가져오기)

```
loxicmd get lb
```

Load-balancer rule 가져오기 (자세한 LB rule 정보 가져오기)

```
loxicmd get lb -o wide
```

json 포맷으로 LB rule 정보 가져오기

```
loxicmd get lb -o json
```

load-balancer rule 설정하는 법

NAT44, tcp, round-robin의 load-balancer rule 설정 예시

```
loxicmd create lb 1.1.1.1 --tcp=1828:1920 --endpoints=2.2.3.4:1
```

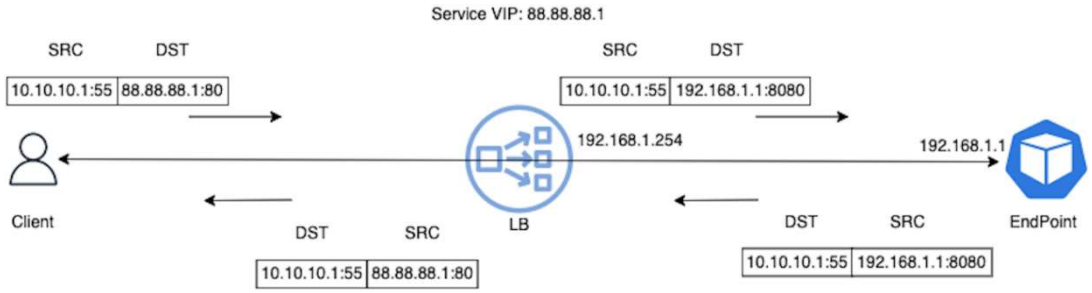

그림 11. Normal NAT (예)

NAT66, tcp, round-robin, load-balancer rule 설정 예시

```
loxicmd create lb 2001::1 --tcp=2020:8080 --endpoints=4ffe::1:1,5ffe::1:1,6ffe::1:1
```

NAT64, tcp, round-robin load-balancer rule 설정 예시

```
loxicmd create lb 2001::1 --tcp=2020:8080 --
endpoints=31.31.31.1:1,32.32.32.1:1,33.33.33.1:1
```

Note

loxilb는 기본적으로 Round-robin 알고리즘을 사용합니다.
Endpoint는 <IPAddress:weight>의 구조로 구성되어 있습니다.
round-robin알고리즘을 사용하는 경우 weight는 큰 의미가 없습니다.

WRR (Weighted round-robin) load-balancer rule 설정 예시(4:4:2비율로 endpoint에 트래픽 전송)

```
loxicmd create lb 20.20.20.1 --select=priority --tcp=2020:8080 --
endpoints=31.31.31.1:40,32.32.32.1:40,33.33.33.1:20
```

hash 모드 load-balancer rule 설정 예시

```
loxicmd create lb 20.20.20.1 --select=hash --tcp=2020:8080 --
endpoints=31.31.31.1:1,32.32.32.1:1,33.33.33.1:1
```

30초 뒤 강제로 tcp session timeout 하는 설정 예시 예시

```
loxicmd create lb 20.20.20.1 --tcp=2020:8080 --
endpoints=31.31.31.1:1,32.32.32.1:1,33.33.33.1:1 --inatimeout=30
```

one-arm 모드 예시

```
loxicmd create lb 20.20.20.1 --tcp=2020:8080 --
endpoints=100.100.100.2:1,100.100.100.3:1,100.100.100.4:1 --mode=onearm
```

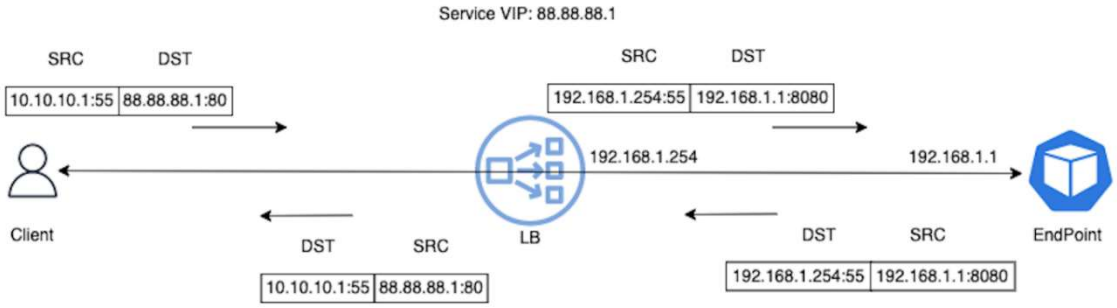

그림 12. One-ARM (예)

Full NAT 모드 예시

```
loxicmd create lb 88.88.88.1 --sctp=38412:38412 --endpoints=192.168.70.3:1 --
mode=fullnat
```

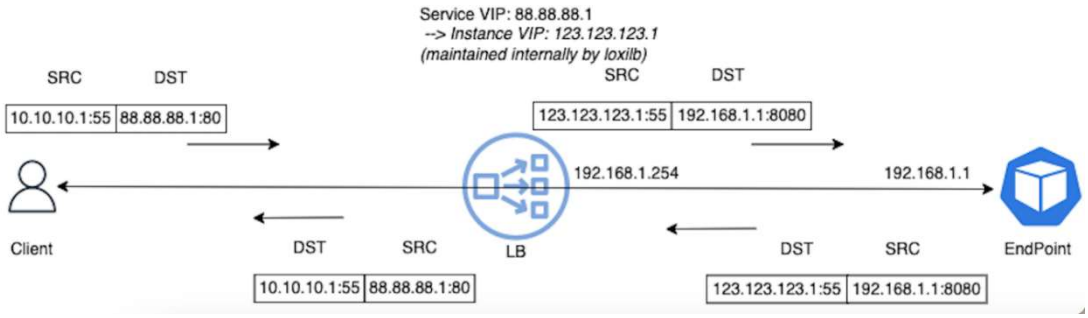

그림 *13.* Full-NAT (예)

Note

Full-NAT 모드에서 loxilb는 들어오는 요청의 소스 IP를 특수 인스턴스 IP로 바꿉니다. 이 인스턴스 IP는 클러스터 배포의 각 인스턴스와 연결되며 loxilb에 의해 내부적으로 유지 관리됩니다. 이 모드에서는 loxilb 클러스터의 다양한 인스턴스가 고유한 인스턴스 IP를 가지며, 각 인스턴스는 엔드포인트를 향해 BGP에 의해 광고되어 그에 따라 반환 경로가 설정됩니다. 이는 액티브-액티브 클러스터링 모드가 필요한 경우 트래픽을 최적으로 분산하고 분산하는 데 도움이 됩니다.

DSR(direct-server return) 모드 예시

```
loxicmd create lb 20.20.20.1 --tcp=2020:8080 --endpoints=31.31.31.1:1,32.32.32.1:1 --
mode=dsr
```

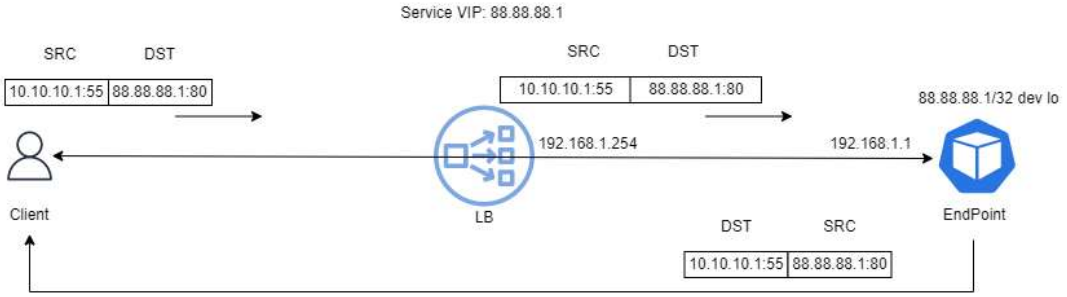

그림 6. L2-DSR mode (예)

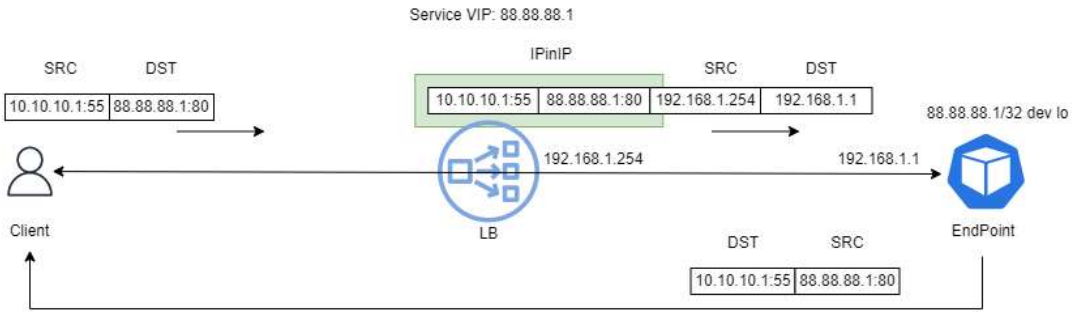

그림 5. L3-DSR mode (예)

Note

L2-DSR(Direct Server Return) 모드에서 loxilb는 로드 밸런싱 작업을 수행하지만 IP 주소는 변경하지 않습니다. 선택한 엔드포인트에 따라 레이어2 헤더만 업데이트합니다. 또한 DSR 모드에서 loxilb는 상태 저장성이 필요하지 않으며 엔드포인트는 loxilb를 포함하지 않는 다른 반환 경로를 선택할 수 있습니다. 이는 리턴 트래픽이 LB를 우회하도록 허용하여 LB 노드의 부하를 줄여야 하는 특정 시나리오에 유리할 수 있습니다..

Note

L3-DSR(Direct Server Return) 모드에서 loxilb는 로드 밸런싱 작업을 수행하지만 원본 페이로드를 엔드포인트로 향하는 IPinIP 터널로 캡슐화합니다. 또한 L2-DSR 모드와 마찬가지로 loxilb는 상태 저장성이 필요하지 않으며 엔드포인트는 loxilb를 포함하지 않는 다른/직접 반환 경로를 선택할 수 있습니다.

endpoint의 health 체크를 같이 하는 예시

```
loxicmd create lb 20.20.20.1 --tcp=2020:8080 --endpoints=31.31.31.1:1,32.32.32.1:1 --monitor
```

Note

Endpoint의 모니터링은 기본이 아닙니다. 위의 예시와 같이 "--monitor" 옵션을 넣어서 Endpoint의 모니터링을 활성화할 수 있습니다. loxilb는 다양한 endpoint 모니터링 방법을 제공합니다. 비활성 Endpoint에 트래픽이 흐를 수 있습니다. 쿠버네티스(K8s)는 보유 서비스 모니터링 매커니즘이 있어 LoxiLB에 해당 endpoint상태를 알려주기 때문입니다.
endpoint section (https://github.com/loxilb-io/loxilbdocs/blob/main/docs/cmd.md#endpoint)에서 자세한 내용을 참조하세요..

Load-balancer YAML 예시

```
apiVersion: netlox/v1
kind: Loadbalancer
metadata:
  name: test
spec:
  serviceArguments:
    externalIP: 1.2.3.1
    port: 80
    protocol: tcp
    sel: 0
  endpoints:
  - endpointIP: 4.3.2.1
    weight: 1
    targetPort: 8080
  - endpointIP: 4.3.2.2
    weight: 1
    targetPort: 8080
  - endpointIP: 4.3.2.3
    weight: 1
    targetPort: 8080
```

Note

1. 일반 NAT

loxilb에서 사용하는 기본 NAT 모드입니다. 이 모드에서 loxilb는 들어오는 요청에 대해 단순 DNAT를 사용합니다. 즉, 대상 IP(서비스 IP이기도 함)가 선택한 엔드포인트 IP로 변경됩니다. 발신 응답의 경우 정반대의 방식(SNAT)을 사용합니다. 이 모드에서는 상태 저장에 의존하기 때문에 리턴 패킷도 loxilb를 통과해야 합니다. 다음 그림은 이 작업을 보여줍니다.

이 모드에서는 원본 소스 IP가 엔드포인트까지 보존되며, 이를 필요로 하는 모든 사용자에게 최상의 가시성을 제공합니다. 마지막으로, 이는 엔드포인트가 소스에 도달하는 방법을 알고 있어야 한다는 의미이기도 합니다.

Note

2. One-arm

기존의 원암 NAT 모드는 LB 노드가 별도의 인그레스 및 이그레스 네트워크 대신 LAN에 하나의 암(또는 연결)을 갖는 것을 의미했습니다. loxilb의 원암 NAT 모드는 기존 원암 모드의 약간 확장된 버전입니다. 원암 모드에서는 엔드포인트 노드로 들어오는 요청을 전송할 때 loxilb가 LAN IP를 소스-IP로 선택합니다. 원본 소스가 동일한 LAN에 있지 않더라도, 이는 원암 모드에 대한 loxilb의 기본 동작입니다.

Note

3. Full-NAT

Full-NAT 모드에서 loxilb는 들어오는 요청의 소스 IP를 특수 인스턴스 IP로 바꿉니다. 이 인스턴스 IP는 클러스터 배포의 각 인스턴스와 연결되며 loxilb에 의해 내부적으로 유지 관리됩니다. 이 모드에서는 loxilb 클러스터의 다양한 인스턴스가 고유한 인스턴스 IP를 가지며, 각 인스턴스는 엔드포인트를 향해 BGP에 의해 광고되어 그에 따라 반환 경로가 설정됩니다. 이는 액티브-액티브 클러스터링 모드가 필요한 경우 트래픽을 최적으로 분산하고 분산하는 데 도움이 됩니다.

Note

4. L2-DSR 모드

L2-DSR(직접 서버 리턴) 모드에서 loxilb는 로드 밸런싱 작업을 수행하지만 IP 주소는 변경하지 않습니다. 선택한 엔드포인트에 따라 레이어2 헤더만 업데이트합니다. 또한 DSR 모드에서 loxilb는 상태 저장성이 필요하지 않으며 엔드포인트는 loxilb를 포함하지 않는 다른 반환 경로를 선택할 수 있습니다. 이는 리턴 트래픽이 LB를 우회하도록 허용하여 LB 노드의 부하를 줄여야 하는 특정 시나리오에 유리할 수 있습니다.

Note

5. L3-DSR 모드

L3-DSR(직접 서버 리턴) 모드에서 loxilb는 로드 밸런싱 작업을 수행하지만 원본 페이로드를 엔드포인트로 향하는 IPinIP 터널로 캡슐화합니다. 또한 L2-DSR 모드와 마찬가지로 loxilb는 상태 저장성이 필요하지 않으며 엔드포인트는 loxilb를 포함하지 않는 다른/직접 반환 경로를 선택할 수 있습니다.

Mirror

Mirror 가져오기

```
loxicmd get mirror
```

Mirror 추가하기

```
#loxicmd create mirror <mirrorIdent> --mirrorInfo=<InfoOption>:<InfoValue>,... --
targetObject=attachement:<port1,rule2>,mirrObjName:<ObjectName>

loxicmd create mirror mirr-1 --mirrorInfo="type:0,port:ensp0" --
targetObject="attachement:1,mirrObjName:ensp1
```

- **mirrorIdent(string)**: Mirror ID
- **type(int)** : 미러링 타입(0=SPAN, 1= RSPAN, 2 = ERSPAN)
- **port(string)** : 미러된 트래픽을 보낼 포트
- **vlan(int)** : RSPAN에 추가할 Tag 번호
- **remoteIP(string)** : ERSPAN 터널에 필요한 Remote IP주소
- **sourceIP(string)** : ERSPAN 터널에 필요한 출발지 IP주소
- **tunnelID(int)** : ERSPAN 에 필요할 ID

Mirror YAML 예시

```
apiVersion: netlox/v1
kind: Mirror
metadata:
  name: test
spec:
  mirrorIdent: mirr-1
  mirrorInfo:
    type: 0
    port: eno1
  targetObject:
    attachment: 1
    mirrObjName: eno2
```

Mirror 삭제

```
#loxicmd delete mirror <mirrorIdent>
loxicmd delete mirror mirr-1
```

Policy

Policy 가져오기

```
loxicmd get policy
```

Policy 추가하기

```
#loxicmd create policy IDENT --rate=<Peak>:<Commited> --
target=<ObjectName>:<Attachment> [--block-size=<Excess>:<Committed>] [--color] [--pol-
type=<policy type>]
loxicmd create policy pol-0 --rate=100:100 --target=ensp0:1
loxicmd create policy pol-1 --rate=100:100 --target=ensp0:1 --block-size=12000:6000
loxicmd create policy pol-1 --rate=100:100 --target=ensp0:1 --color
loxicmd create policy pol-1 --rate=100:100 --target=ensp0:1 --color --pol-type 0
```

- rate(string): Mbps기반의 트래픽 속도 제어량 'Peak:Commited의 쌍으로 표시
- block-size(string): bps기반의 Block 허용량 'Excess:Committed'으로 표시.
- target(string): 'ObjectName:Attachment'로 구성된 타겟 인터페이스
- color(boolean): Color의 활성화
- pol-type(int): Policy 트래픽 제어 타입. 0 : TrTCM, 1 : SrTCM

아래와 같이 'policy' 명령어를 실행하여 결과를 확인하는 (예)를 볼 수 있습니다.

```
root@92bcacbbf6b8:/# loxicmd create policy pol-1 --rate=100:100 --target=ensp0:1 --block-size=12000:6000
Debug: response.StatusCode: 200
root@92bcacbbf6b8:/# loxicmd get policy
| IDENT | PEAKINFORATE | COMMITTEDINFORATE |
|-------|--------------|-------------------|
| pol-1 |          100 |               100 |
root@92bcacbbf6b8:/# loxicmd get policy -o wide
| IDENT | PEAKINFORATE | COMMITTEDINFORATE | EXCESSBLKSIZE | COMMITTEDBLKSIZE | POLICYTYPE | COLORAWARE | POLOBJNAME | ATTACHMENT |
|-------|--------------|-------------------|---------------|------------------|------------|------------|------------|------------|
| pol-1 |          100 |               100 |         12000 |             6000 |          0 | false      | ensp0      |          1 |
root@92bcacbbf6b8:/#
```

- **Ident** : Policy 아이디
- **peakInfoRate** : Policy에 적용된 Peak 값
- **committedInfoRate** : Policy에 적용된 Commit값
- **excessBlkSize** :블록 사이즈
- **committedBlkSize** : Commit블록 사이즈
- **policyType** : Policy 적용 타입

- **ColorAware** : Color 적용 유무
- **polObjName** : Policy적용 인터페이스
- **attachment** : Port/Rule attachment정보(1이 포트)

Policy YAML 예시

```
apiVersion: netlox/v1
kind: Policy
metadata:
  name: test
spec:
  policyIdent: pol-eno8
  policyInfo:
    type: 0
    colorAware: false
    committedInfoRate: 100
    peakInfoRate: 100
  targetObject:
    attachment: 1
    polObjName: eno8
```

Policy 삭제

```
#loxicmd delete policy <Polident>
loxicmd delete policy pol-1
```

Route

Route 가져오기

```
loxicmd get route
```

Route 추가하기

```
#loxicmd create route <DestinationIPNet> <gateway>
loxicmd create route 192.168.212.0/24 172.17.0.254
```

- DestinationIPNet(string): CIDR구조의 route 정보
- gateway(string): 게이트웨이 및 넥스트 홉

아래와 같이 'route' 명령어를 실행하여 결과를 확인하는 (예)를 볼 수 있습니다.

```
root@92bcacbbf6b8:/# loxicmd create route 192.168.212.0/24 172.17.0.254
Success
root@92bcacbbf6b8:/# loxicmd get route
| DESTINATIONIPNET |   GATEWAY    |   FLAG   |
|------------------|--------------|----------|
| 0.0.0.0/0        | 172.17.0.1   | Ind      |
| 127.0.0.0/8      |              | Self     |
| 172.17.0.0/16    |              | Self     |
| 172.17.0.1/32    | 172.17.0.1   | Ind Host |
| 172.17.0.254/32  | 172.17.0.254 | Ind Host |
| 192.168.0.1/32   | 192.168.0.1  | Ind Host |
| 192.168.212.0/24 | 172.17.0.254 | Ind      |
root@92bcacbbf6b8:/# loxicmd get route -o wide
| DESTINATIONIPNET |   GATEWAY    |   FLAG   | HARDWAREMARK | PACKETS | BYTES |
|------------------|--------------|----------|--------------|---------|-------|
| 0.0.0.0/0        | 172.17.0.1   | Ind      |            4 |       0 |     0 |
| 127.0.0.0/8      |              | Self     |            1 |       0 |     0 |
| 172.17.0.0/16    |              | Self     |            2 |       0 |     0 |
| 172.17.0.1/32    | 172.17.0.1   | Ind Host |            3 |       0 |     0 |
| 172.17.0.254/32  | 172.17.0.254 | Ind Host |            6 |       0 |     0 |
| 192.168.0.1/32   | 192.168.0.1  | Ind Host |            5 |       0 |     0 |
| 192.168.212.0/24 | 172.17.0.254 | Ind      |            7 |       0 |     0
|root@92bcacbbf6b8:/#
```

- **destinationIPNet** : 라우팅할 IP/netmask 정보

- **gateway** : nexthop IP주소
- **flag** : 라우팅 테이블 Flag
- **HardwareMark** : 인터페이스 마크
- **packets** : 라우팅 테이블 패킷 수(enable의 경우에만 적용)
- **bytes** :라우팅 테이블 byte 수(enable의 경우에만 적용)

Route YAML 예시

```
apiVersion: netlox/v1
kind: Route
metadata:
  name: test
spec:
  destinationIPNet: 192.168.30.0/24
gateway: 172.17.0.1
```

Route 삭제

```
#loxicmd delete route <DestinationIPNet>
loxicmd delete route 192.168.212.0/24
```

VLAN

VLAN, VLAN Member 가져오기

```
loxicmd get vlan
```

VLAN, VLAN Member 만들기

```
#loxicmd create vlan <Vid>
loxicmd create vlan 100
```

Vid(int): vlanID 번호

```
#loxicmd create vlanmember <Vid> <DeviceName> --tagged=<Tagged>
loxicmd create vlanmember 100 eno7 --tagged=true
loxicmd create vlanmember 100 eno7
```

- Vid(int): vlanID번호
- DeviceName(string): 인터페이스 이름
- tagged(boolean): Vlan Tag여부 (기본값은 False)

아래와 같이 명령어 실행(예)를 볼 수 있습니다.

```
root@92bcacbbf6b8:/# loxicmd create vlan 100
Success
root@92bcacbbf6b8:/# loxicmd create vlanmember 100 eth0 --tagged=true
Success
root@92bcacbbf6b8:/# loxicmd get vlan
| DEVICE NAME | VLAN ID |      MEMBER      |
|-------------|---------|------------------|
| vlan100     |     100 | Device: eth0.100 |
|             |         | tagged: true     |
|             |         | Device: eth0     |
|             |         | tagged: false    |
| vlan3801    |    3801 |                  |
| vlan3803    |    3803 |                  |
```

VLAN YAML 예시

```
apiVersion: netlox/v1
kind: Vlan
metadata:
  name: test
spec:
vid: 100
```

VLAN Member YAML 예시

```
apiVersion: netlox/v1
kind: VlanMember
metadata:
  name: test
  vid: 100
spec:
  dev: eno8
Tagged: true
```

VLAN, VLAN Member 삭제

```
#loxicmd delete vlan <Vid>
loxicmd delete vlan 100

#loxicmd delete vlanmember <Vid> <DeviceName> --tagged=<Tagged>
loxicmd delete vlanmember 100 eno7 --tagged=true
loxicmd delete vlanmember 100 eno7
```

VxLAN

VxLAN, VxLAN Peer 가져오기

```
loxicmd get vxlan
```

VxLAN, VxLAN Peer 추가하기

```
#loxicmd create vxlan <VxlanID> <EndpointDeviceName>
loxicmd create vxlan 100 eno7
```

- EndpointDeviceName(string): VTEP 인터페이스 이름

VxLANID(int): VxLAN ID 번호

```
#loxicmd create vxlanpeer <VxlanID> <PeerIP>
loxicmd create vxlan-peer 100 30.1.3.1
```

- VxlanID(int): Vxlan ID 번호
- PeerIP(string): Vxlan peer의 IP 주소

아래와 같이 'vxlan' 명령어를 실행하여 결과를 확인하는 (예)를 볼 수 있습니다.

```
root@92bcacbbf6b8:/# loxicmd create vxlan 100 eth0
Success
root@92bcacbbf6b8:/# loxicmd get vxlan
| DEVICE NAME | VXLAN ID | ENDPOINT INTERFACE | PEER IP |
|-------------|----------|--------------------|---------|
| vxlan100    |      100 | eth0               |         |
root@92bcacbbf6b8:/# loxicmd create vxlan-peer 100 30.1.3.1
Success
root@92bcacbbf6b8:/# loxicmd get vxlan -o wide
| DEVICE NAME | VXLAN ID | ENDPOINT INTERFACE | PEER IP  |
|-------------|----------|--------------------|----------|
| vxlan100    |      100 | eth0               | 30.1.3.1 |
root@92bcacbbf6b8:/#
```

VxLAN yaml 예시

```
apiVersion: netlox/v1
kind: Vxlan
metadata:
  name: test
spec:
  epIntf: eno8
vxlanID: 100
```

VxLAN Peer yaml 예시

```
apiVersion: netlox/v1
kind: VxlanPeer
metadata:
  name: test
  vxlanID: 100
spec:
  peerIP: 21.21.21.1
```

VxLAN, VxLAN Peer 삭제하기

```
#loxicmd delete vxlan <VxlanID>
loxicmd delete vxlan 100

#loxicmd delete vxlanpeer <VxlanID> <PeerIP>
loxicmd delete vxlan-peer 100 30.1.3.1   peerIP: 21.21.21.1
```

SUMMARY

loxicmd 명령어 기능별 사용 레퍼런스 입니다.

부록

02

UI

UI: LoxiLB Enterprise의 GUI(Graphical User Interface)

LoxiLB Enterprise의 GUI(Graphical User Interface)를 'LoxiLB Enterprise' 사용자들에게 제공하며, 'loxilb' 구성의 기기(가상머신/베어메탈) 정보들을 직관적으로 관리할 수 있어 운영에 도움을 줄 수 있습니다.

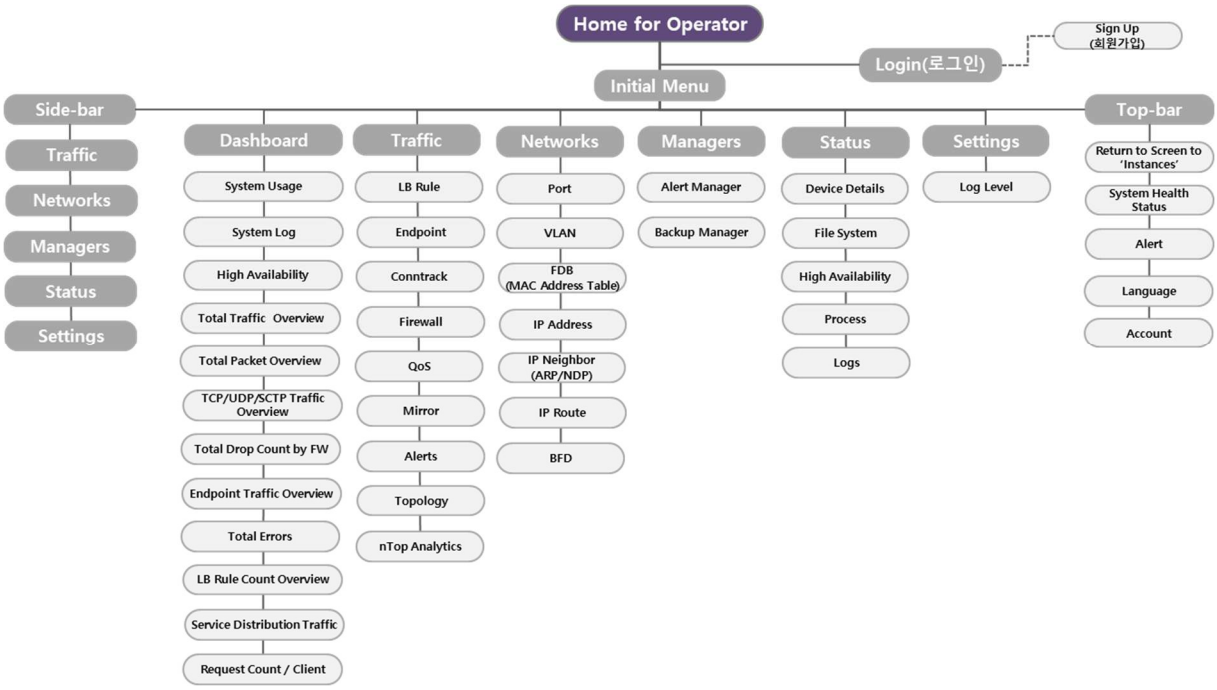

그림 167. UI Architecture

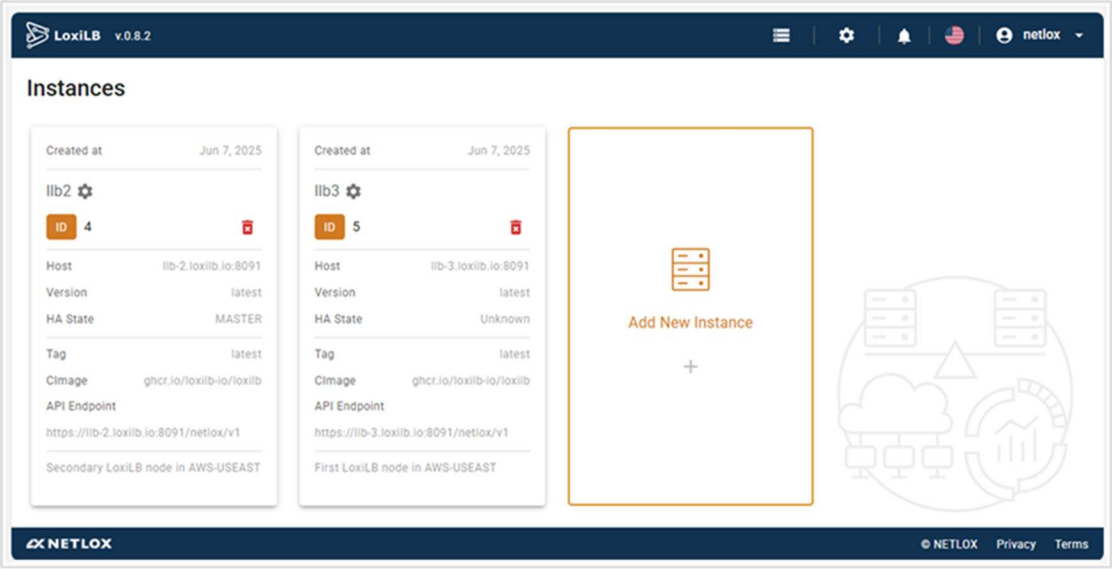

그림 8. LoxiLB GUI의 loxilb 선택 화면

'loxilb' 기기별 박스의 제공 정보는 아래와 같은 내용을 확인할 수 있습니다.

- Host 정보와 버전(Version)
- HA State: 고가용성(High Availability) 동작을 Master/Backup 으로 표시
- 사용 컨테이너 이미지(Image)와 태그(Tag)
- 'loxilb' 등록 삭제와 수정 기능
- API 정보

아래 ‘그림 17. 대시보드(Dashboard) ’와 같이 LoxiLB GUI(Graphical User Interface)는 각각의 ‘loxilb’ 기기의 자원 사용량을 모니터링 할 수 있습니다.

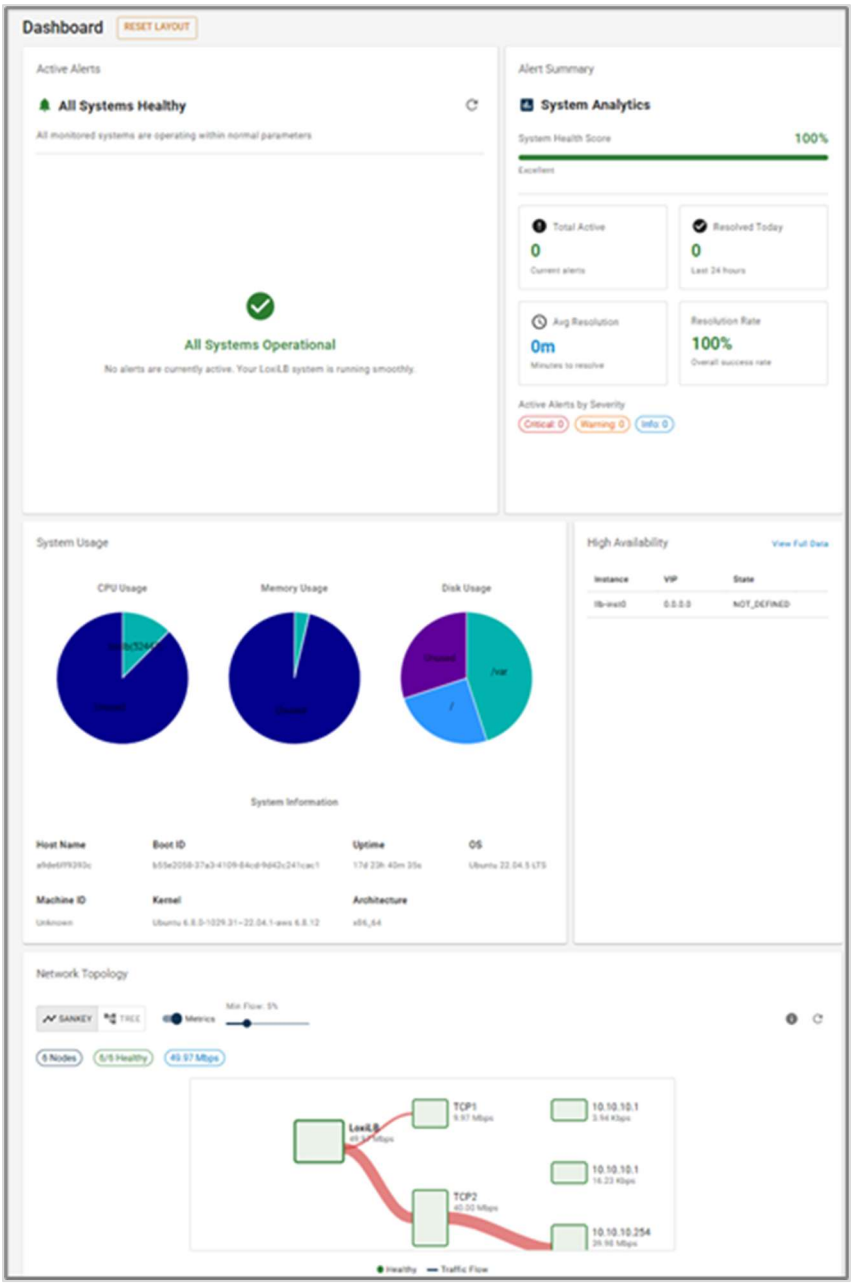

그림18. 대시보드(Dashboard)

대사보드에서 제공하는 사이드 탭은 각 ‘loxilb’ 기기의 트래픽(Traffic), 네트워크(Networks), 상태(Status) 등의 상세 사항을 선택하여 확인할 수 있습니다.

아래 ‘그림 18. 트래픽(Traffic)의 로드밸런싱(LB) 룰(Rule) ’에서와 같이 LB Rule 별 설정 값들을 확인할 수 있습니다.

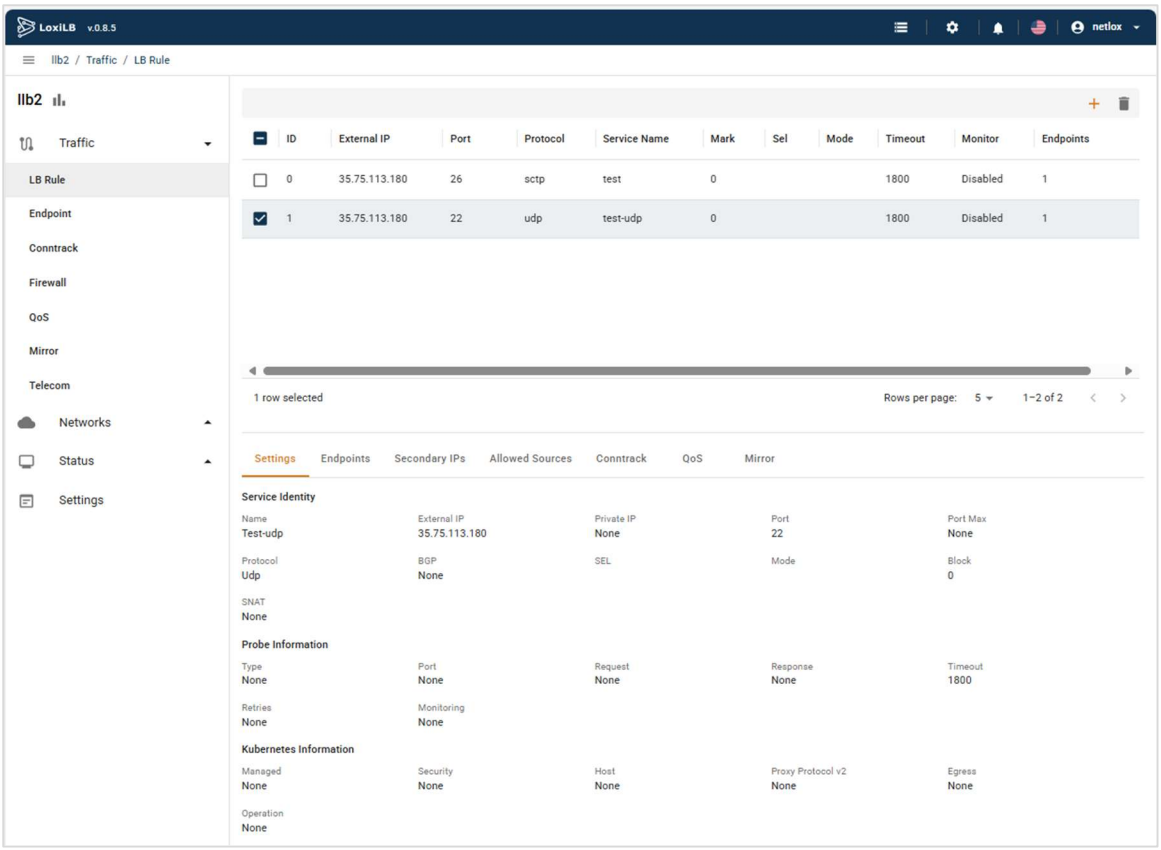

그림 19. 트래픽(Traffic)의 로드밸런싱(LB) 룰(Rule)

LB Rule의 Endpoint를 선택하면 아래 '그림 19. 엔드포인트(Endpoint)'와 같이 Active 상태와 사용량 Counter를 확인할 수 있습니다.

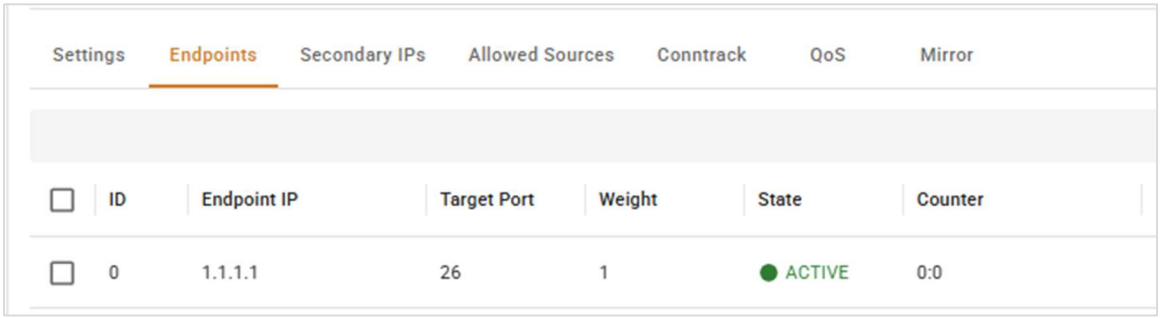

그림 20. 엔드포인트(Endpoint)

사이드탭 Conntrack을 선택하면 아래 '그림 20. 콘트랙(Conntrack)'과 같이 선택한 Conntrack의 설정값과 사용량을 확인할 수 있습니다.

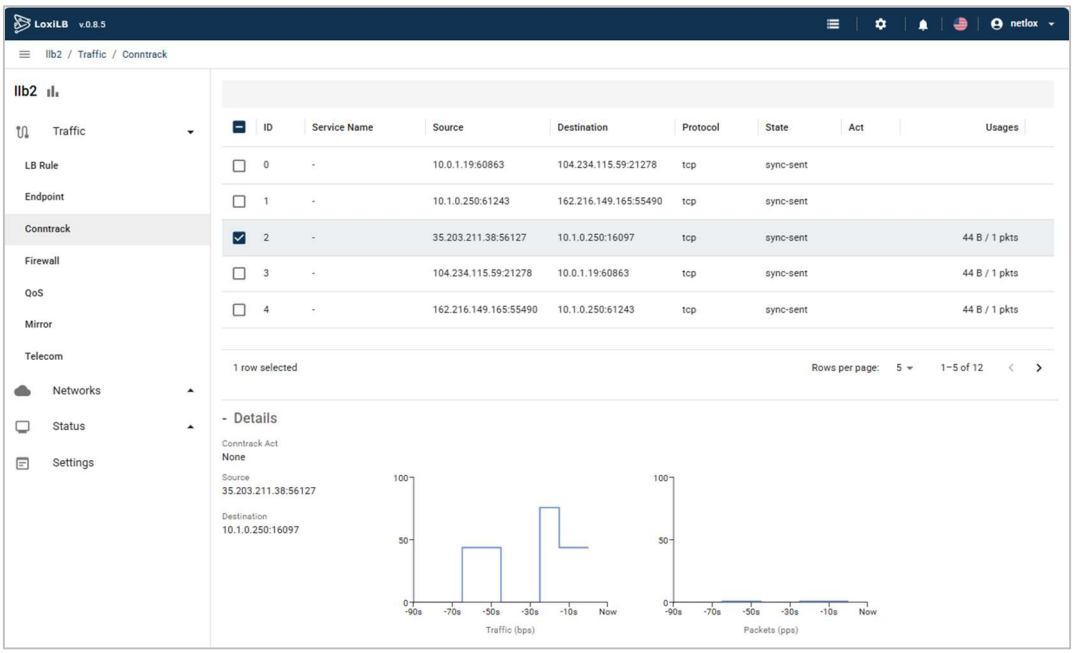

그림 21. 콘트랙(Conntrack)

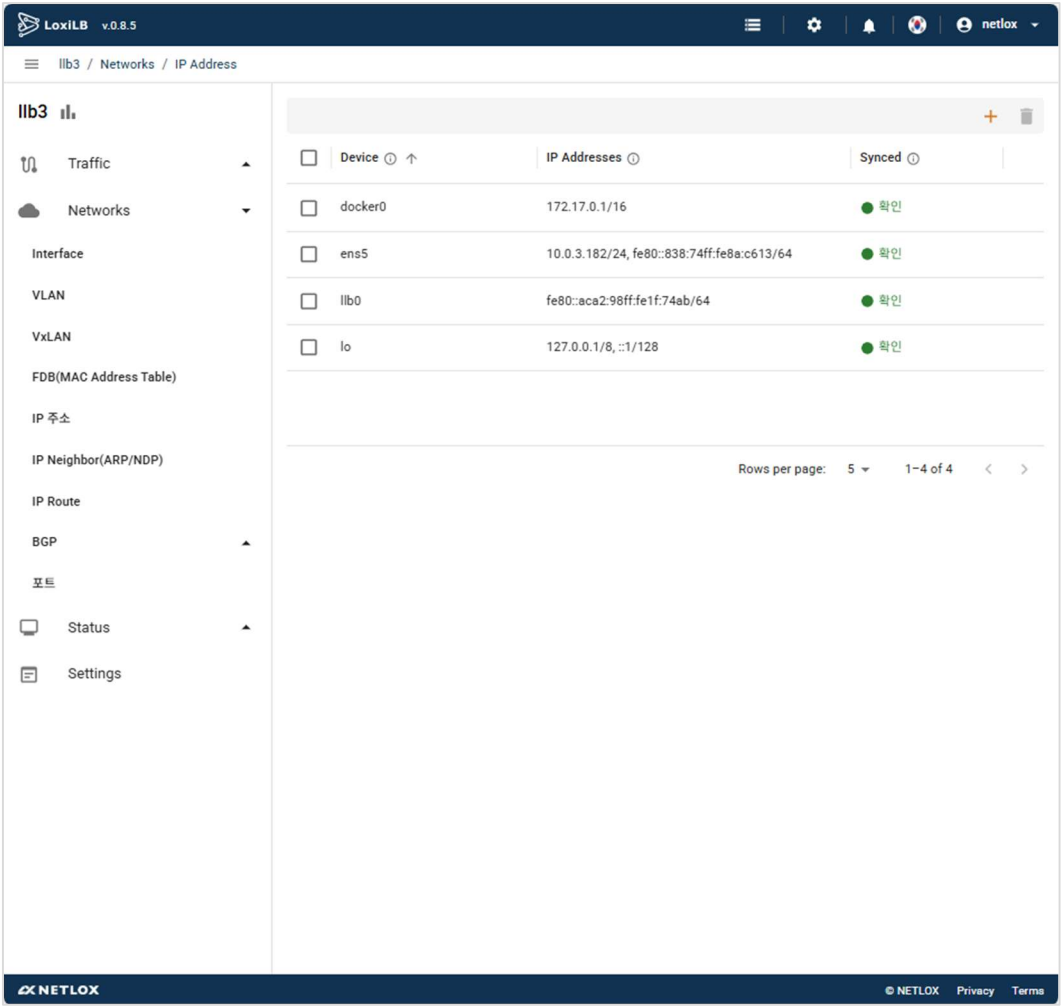

그림 229. IP Address

SUMMARY

LoxiLB Enterprise의 GUI(Graphical User Interface)를 'LoxiLB Enterprise' 사용자들에게 제공하며, 'loxilb' 구성의 기기(가상 머신/베어메탈) 정보들을 직관적으로 관리할 수 있어 운영에 도움을 줄 수 있습니다.

부록

03

Error 메시지

Error 메세지

서비스 구성 오류

에러 메시지	심각도	대응방안
no-realport error	MEDIUM	실제 포트 설정 확인, 포트 매핑 검토
malformed-service-pport error	HIGH	서비스 포트 형식 검증, 포트 번호 범위 확인
malformed-service-ptype error	HIGH	서비스 프로토콜 타입 확인, 지원 프로토콜 사용
proxy-proto-v2 not tcp service error	MEDIUM	TCP 서비스에서만 프록시 프로토콜 v2 사용 가능
malformed-service error	HIGH	서비스 설정 전체 검증, JSON/YAML 형식 확인
port exists	MEDIUM	포트 중복 사용 확인, 다른 포트 번호 사용
no such port	MEDIUM	포트 존재 확인, 포트 생성 후 사용
no-portimap error	MEDIUM	포트 내부 매핑 설정 확인
malformed-service privateIP error	HIGH	프라이빗 IP 형식 검증, RFC1918 준수 확인
fdb port error	MEDIUM	FDB 포트 설정 확인
fdb v6 dst unsupported	MEDIUM	IPv6 목적지 FDB 미지원, IPv4 사용 권장
malformed service proto	HIGH	서비스 프로토콜 형식 검증
Phy port not created	MEDIUM	물리 포트 생성 확인, 포트 초기화
No such tag port	MEDIUM	태그 포트 존재 확인
no-realport sif error	MEDIUM	SIF 실제 포트 설정 확인
malformed-service nat46 error	HIGH	NAT46 서비스 설정 형식 검증
no-portomap error	MEDIUM	포트 출력 매핑 설정 확인
service-args error	MEDIUM	서비스 인자 유효성 검증
malformed-proto error	HIGH	프로토콜 형식 검증, 표준 프로토콜 사용
malformed-service dsr-port error	HIGH	DSR 포트 설정 형식 검증
proto error	MEDIUM	프로토콜 설정 확인
no-port error	MEDIUM	포트 설정 필수
host-args unknown probe port	MEDIUM	프로브 포트 설정 확인, 알려진 포트 사용

로드밸런싱 규칙 오류

에러 메시지	심각도	대응방안
lbrule-exists error	HIGH	기존 규칙 확인, 중복 규칙 수정 또는 삭제
lbrule not-exists error	HIGH	규칙 생성 확인, 존재하지 않는 규칙 참조 해제
rule-allowed-src error	HIGH	허용된 소스 IP 범위 확인, 규칙 정정
lbrule-exist error: cant modify rule security mode	CRITICAL	보안 모드 규칙 수정 불가, 규칙 재생성 필요
rule-mark error	HIGH	규칙 마킹 값 범위 확인, 유효한 마크 사용
lbrule-exist error: cant modify fullproxy rule mode	CRITICAL	풀프록시 모드 규칙 수정 불가, 규칙 재생성 필요
malformed-rule dst error	HIGH	목적지 규칙 형식 검증, IP 주소/포트 확인
lbrule-exist error: cant modify rule egress mode	CRITICAL	이그레스 모드 규칙 수정 불가, 규칙 재생성 필요
fwrule-exists error	HIGH	방화벽 규칙 중복 확인, 기존 규칙 수정
rule-snat error	HIGH	SNAT 규칙 설정 확인, NAT 테이블 검토
malformed-rule src error	HIGH	소스 규칙 형식 검증, IP 주소/포트 확인
no-rule error	HIGH	규칙 설정 필수, 최소 하나의 규칙 생성
rule-hwm error	HIGH	규칙 하이 워터마크 초과, 규칙 수 조정
LB Rule-referred	HIGH	참조된 LB 규칙, 종속성 확인 후 삭제

네트워크 인터페이스 오류

에러 메시지	심각도	대응방안
not ipv4 address	MEDIUM	IPv4 주소 형식 검증, 올바른 형식 사용
source address malformed	HIGH	소스 주소 형식 검증, IP 주소 정정
myip address parse error	HIGH	로컬 IP 주소 파싱 오류, 주소 형식 확인
malformed-secIP nat46 error	HIGH	NAT46 보조 IP 형식 검증
ip address parse error	HIGH	IP 주소 파싱 오류, 형식 검증
no such ip address	MEDIUM	IP 주소 존재 확인, 할당된 주소 사용
secondaryIP-args len error	MEDIUM	보조 IP 인자

인증 보안 오류

에러 메시지	심각도	대응방안
password must not contain the same character more than twice in a row	MEDIUM	패스워드 정책 준수: 동일 문자 3회 연속 사용 금지
password must not be the same as the username	MEDIUM	패스워드와 사용자명 다르게 설정
invalid refresh token in cache	MEDIUM	토큰 캐시 정리, 새로운 토큰 발급
user not found	HIGH	사용자 계정 존재 확인, 계정 생성 필요
invalid token format in cache	MEDIUM	토큰 형식 검증, 캐시 무효화
password must contain at least one uppercase letter	MEDIUM	대문자 최소 1개 포함 필수
token is expired	HIGH	토큰 갱신, 자동 갱신 메커니즘 확인
Token not found") // Token not fou	HIGH	토큰 존재 확인, 새로운 토큰 발급
User not found") // User not fou	HIGH	사용자 계정 확인, 계정 복구 절차
invalid token	MEDIUM	토큰 유효성 검증, 재인증 수행
password must contain at least one lowercase letter	MEDIUM	소문자 최소 1개 포함 필수
password must contain at least one special character	MEDIUM	특수문자 최소 1개 포함 필수
password must be at least 9 characters long	MEDIUM	패스워드 최소 9자리 이상 설정
password must not be the same as the previous password	MEDIUM	이전 패스워드와 다른 새 패스워드 사용
username already exists	MEDIUM	고유한 사용자명 사용, 중복 확인
no-user error	MEDIUM	사용자 설정 필수
token not found	HIGH	토큰 생성 및 할당 확인
password must contain at least one number	MEDIUM	숫자 최소 1개 포함 필수

일반 오류

에러 메시지	심각도	대응방안
unknown work type	MEDIUM	작업 타입 설정 확인, 지원되는 타입 리스트 검토
my discriminator not found	MEDIUM	식별자 설정 확인, 중복 여부 검토
existing cnode	MEDIUM	클러스터 노드 상태 확인, 중복 등록 해제
malformed-args error	MEDIUM	API 파라미터 형식 검증, 스키마 확인
Overflow	MEDIUM	버퍼 크기 조정, 메모리 사용량 최적화
netrpc call timeout	CRITICAL	네트워크 연결 상태 확인, RPC 서버 응답성 점검
rootca cert load failed	CRITICAL	인증서 파일 경로 및 권한 확인, 만료일 검토
ulcl-ulhwm error	MEDIUM	상위 워터마크 설정 조정
zone number err	MEDIUM	존 번호 범위 확인, 설정 정정
prop-exists error	MEDIUM	속성 중복 확인, 기존 속성 삭제 후 재설정
no such zone	MEDIUM	존 생성 상태 확인, 존 설정 파일 검토
RT Trie Err	MEDIUM	라우팅 트리 구조 검증, 메모리 상태 확인
same fdb	MEDIUM	FDB 엔트리 중복 확인, 중복 제거
rt mod error	MEDIUM	라우팅 테이블 수정 권한 확인
Not-Ready	CRITICAL	시스템 초기화 완료 대기, 의존성 서비스 확인
send-err	MEDIUM	네트워크 송신 상태 확인, 버퍼 용량 검토
no such cnode	MEDIUM	클러스터 노드 등록 상태 확인
malformed-args fwmark !=0 for snat-error	MEDIUM	SNAT 설정 시 방화벽 마크 값 검증
invalid parameters	MEDIUM	파라미터 유효성 검사, 허용 범위 확인
af-packet-err	MEDIUM	AF_PACKET 소켓 설정 확인
existing zone	MEDIUM	존 중복 생성 방지, 기존 존 활용
fdb attr error	MEDIUM	FDB 속성 설정 확인
malformed-weight error	MEDIUM	가중치 값 범위 및 형식 검증
unknown log level	MEDIUM	로그 레벨 설정값 확인, 유효한 레벨로 변경
no-ulcl error	MEDIUM	ULCL 설정 상태 확인
need-realdev error	MEDIUM	실제 장치 필요, 가상 장치 대신 물리 장치 사용
not such ifname	MEDIUM	인터페이스 이름 확인, 존재하는 인터페이스로 변경
non-udp-n3-args error	MEDIUM	N3 인터페이스 설정 시 UDP 프로토콜 사용 확인
rt exists	MEDIUM	라우트 중복 확인, 기존 라우트 수정 또는 삭제
intfv6-err	MEDIUM	IPv6 인터페이스 설정 확인
no loxi-eni found	MEDIUM	LoxiLB ENI 설정 확인, ENI 생성
invalid gws	MEDIUM	게이트웨이 설정 검증
zone exists	MEDIUM	존 중복 확인, 고유한 존 이름 사용
not found	MEDIUM	요청한 리소스 존재 확인
not such addrs	MEDIUM	주소 설정 확인, 유효한 주소 사용
sess-mod error	MEDIUM	세션 수정 권한 및 상태 확인

intf-err	MEDIUM	인터페이스 설정 전반 검토
ulcl-dlhwm error	MEDIUM	하위 워터마크 설정 조정
no such fdb	MEDIUM	FDB 엔트리 존재 확인
no-master error	MEDIUM	마스터 노드 설정 확인
fdb zone error	MEDIUM	FDB 존 설정 확인
Mask format is wrong	MEDIUM	서브넷 마스크 형식 검증
vxlan can not be tagged	MEDIUM	VXLAN 설정 시 태깅 불가 확인
unexpected HTTP response	MEDIUM	HTTP 응답 코드 확인, API 엔드포인트 검증
bindv6-err	MEDIUM	IPv6 바인딩 설정 확인
bind-err	MEDIUM	포트 바인딩 상태 확인, 포트 사용 중 여부 점검
malformed-lbep error	MEDIUM	로드밸런서 엔드포인트 형식 검증
zone is not set	MEDIUM	존 설정 필수, 기본 존 지정
no such allowed src prefix	MEDIUM	허용된 소스 프리픽스 설정 확인
no such policer error	MEDIUM	폴리서 설정 존재 확인
no neigh found	MEDIUM	네이버 테이블 확인, ARP 엔트리 점검
no-zone error	MEDIUM	존 설정 필수
Range	MEDIUM	범위 설정 값 확인
nh exists	MEDIUM	넥스트홉 중복 확인
prop-noexist error	MEDIUM	속성 존재 확인, 필수 속성 설정
sess-exists error	MEDIUM	세션 중복 확인, 기존 세션 정리
no such ifa	MEDIUM	인터페이스 어드레스 확인
no-nh error	MEDIUM	넥스트홉 설정 필수
ulcl-exists error	MEDIUM	ULCL 중복 확인

SUMMARY

Error 메시지

LoxiLB Enterprise의 유형별 에러 메시지입니다.

LoxiLB Enterprise User Guide

초판일 2025년 9월 22일

회사 NetLOX

이메일 contact@netlox.io

블록그 https://www.loxilb.io/blog

GitHub https://github.com/loxilb-io/loxilb